# Oracle9*i*

User-Managed Backup and Recovery Guide

Release 1 (9.0.1)

June 2001

Part No. A90134-01

ORACLE®

Oracle9*i* User-Managed Backup and Recovery Guide, Release 1 (9.0.1)

Part No. A90134-01



Primary Author: Lance Ashdown

Contributors: Tammy Bednar, Wei Hu, Vikram Joshi, Bill Lee, Yunrui Li, Gary Ngai, Ron Obermarck, Alok Pareek, Vinay Srihari, Janet Stern, Mike Stewart, Kothanda Umamageswaran

# Contents

## 3 Performing User-Managed Restore Operations

## 4 Performing User-Managed Media Recovery



## 5 Troubleshooting User-Managed Media Recovery

## 6 User-Managed Media Recovery Scenarios



## 7 Performing User-Managed TSPITR

## Index

# Send Us Your Comments

**Oracle9*i* User-Managed Backup and Recovery Guide, Release 1 (9.0.1)**

**Part No. A90134-01**

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

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

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

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

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

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

# Preface

This preface contains these topics:

## Audience

This manual is intended for database administrators who perform the following tasks:

- Perform backups and restore database files with operating system utilities, and perform recovery with the SQL*Plus RECOVER command.
- Use Recovery Manager as a backup and recovery solution but choose to back up some files with user-managed methods.

To use this document, you need to be familiar with relational database concepts and basic database administration as described in these manuals:

- *Oracle9i Database Concepts*
- *Oracle9i Backup and Recovery Concepts*
- *Oracle9i Database Administrator's Guide*

You should also be familiar with the operating system environment under which you are running Oracle.

## Organization

This document contains:

**"What's New in User-Managed Backup and Recovery?"**

This preface describes the new features and enhancements to user-managed backup and recovery.

**Chapter 1, "Introduction to User-Managed Backup and Recovery"**

This chapter explains the purpose and basic functionality of user-managed backup and recovery methods.

**Chapter 2, "Making User-Managed Backups"**

This chapter describes how to back up control files, datafiles, and archived redo logs with operating system commands.

**Chapter 3, "Performing User-Managed Restore Operations"**

This chapter describes how to restore control files, database file, and archived redo logs with operating system commands.

### Chapter 4, "Performing User-Managed Media Recovery"

This chapter describes how to use the SQL*Plus RECOVER command to perform media recovery on restored datafiles.

### Chapter 5, "Troubleshooting User-Managed Media Recovery"

This chapter describes how to troubleshoot problems that can occur when performing user-managed media recovery.

### Chapter 6, "User-Managed Media Recovery Scenarios"

This chapter describes basic scenarios involving user-managed restore and recovery.

### Chapter 7, "Performing User-Managed TSPITR"

This chapter describes how recover a tablespace to a time that is different from the rest of the database.

## Related Documentation

For more information, see these Oracle resources:

- *Oracle9i Backup and Recovery Concepts* to gain a conceptual overview of backup and recovery
- *Oracle9i Recovery Manager User's Guide* to learn about Recovery Manager
- `http://www.oracle.com/database/recovery`

In North America, printed documentation is available for sale in the Oracle Store at

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

Customers in Europe, the Middle East, and Africa (EMEA) can purchase documentation from

`http://www.oraclebookshop.com/`

Other customers can contact their Oracle representative to purchase printed documentation.

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

`http://technet.oracle.com/membership/index.htm`

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

`http://technet.oracle.com/docs/index.htm`

# Conventions

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

- Conventions in Text
- Conventions in Code Examples

## Conventions in Text

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

| Convention | Meaning | Example |
|---|---|---|
| **Bold** | Bold typeface indicates terms that are defined in the text or terms that appear in a glossary, or both. | The C datatypes such as **ub4**, **sword**, or **OCINumber** are valid.<br>When you specify this clause, you create an **index-organized table**. |
| *Italics* | Italic typeface indicates book titles, emphasis, syntax clauses, or placeholders. | *Oracle9i Database Concepts*<br>Ensure that the recovery catalog and target database do *not* reside on the same disk. |
| UPPERCASE monospace (fixed-width font) | Uppercase monospace typeface indicates elements supplied by the system. Such elements include parameters, privileges, datatypes, RMAN keywords, SQL keywords, SQL*Plus or utility commands, packages and methods, as well as system-supplied column names, database objects and structures, user names, and roles. | You can specify this clause only for a `NUMBER` column.<br>You can back up the database using the `BACKUP` command.<br>Query the `TABLE_NAME` column in the `USER_TABLES` data dictionary view.<br>Specify the `ROLLBACK_SEGMENTS` parameter.<br>Use the `DBMS_STATS.GENERATE_STATS` procedure. |

| Convention | Meaning | Example |
|---|---|---|
| lowercase monospace (fixed-width font) | Lowercase monospace typeface indicates executables and sample user-supplied elements. Such elements include computer and database names, net service names, and connect identifiers, as well as user-supplied database objects and structures, column names, packages and classes, user names and roles, program units, and parameter values. | Enter `sqlplus` to open SQL*Plus.<br>The `department_id`, `department_name`, and `location_id` columns are in the `hr.departments` table.<br>Set the `QUERY_REWRITE_ENABLED` initialization parameter to `true`.<br>Connect as `oe` user. |

## Conventions in Code Examples

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

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

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

| Convention | Meaning | Example |
|---|---|---|
| [ ] | Brackets enclose one or more optional items. Do not enter the brackets. | `DECIMAL (digits [ , precision ])` |
| { } | Braces enclose two or more items, one of which is required. Do not enter the braces. | `{ENABLE \| DISABLE}` |
| \| | A vertical bar represents a choice of two or more options within brackets or braces. Enter one of the options. Do not enter the vertical bar. | `{ENABLE \| DISABLE}`<br>`[COMPRESS \| NOCOMPRESS]` |
| ... | Horizontal ellipsis points indicate either:<br>■ That we have omitted parts of the code that are not directly related to the example<br>■ That you can repeat a portion of the code | `CREATE TABLE ... AS subquery;`<br>`SELECT col1, col2, ... , coln FROM employees;` |
| .<br>.<br>. | Vertical ellipsis points indicate that we have omitted several lines of code not directly related to the example. | |

| Convention | Meaning | Example |
|---|---|---|
| Other notation | You must enter symbols other than brackets, braces, vertical bars, and ellipsis points as it is shown. | `acctbal NUMBER(11,2);`<br>`acct CONSTANT NUMBER(4) := 3;` |
| *Italics* | Italicized text indicates variables for which you must supply particular values. | `CONNECT SYSTEM/system_password` |
| UPPERCASE | Uppercase typeface indicates elements supplied by the system. We show these terms in uppercase in order to distinguish them from terms you define. Unless terms appear in brackets, enter them in the order and with the spelling shown. However, because these terms are not case sensitive, you can enter them in lowercase. | `SELECT last_name, employee_id FROM employees;`<br>`SELECT * FROM USER_TABLES;`<br>`DROP TABLE hr.employees;` |
| lowercase | Lowercase typeface indicates programmatic elements that you supply. For example, lowercase indicates names of tables, columns, or files. | `SELECT last_name, employee_id FROM employees;`<br>`sqlplus hr/hr` |

## Documentation Accessibility

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

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

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

# What's New in User-Managed Backup and Recovery?

This chapter describes the new user-managed backup and recovery features of Oracle9*i* Release 1 (9.0.1) and provides pointers to additional information. New features information from previous releases is also retained to help those users migrating to the current release.

The following sections describe the new features in user-managed backup and recovery:

- Oracle9i Release 1 (9.0.1) New Features in User-Managed Backup and Recovery
- Oracle8i New Features in User-Managed Backup and Recovery

# Oracle9*i* Release 1 (9.0.1) New Features in User-Managed Backup and Recovery

Oracle9*i* Release 1 (9.0.1) includes the following new features for backup and recovery that improve database availability and manageability.

- **Batch Termination of Backup Mode**

  The ALTER DATABASE END BACKUP statement takes all datafiles currently in backup mode out of backup mode. The purpose of this feature is to allow a crash recovery script to restart a database without intervention even though the failure occurred during an online backup. Previously, if the database crashed during an online backup, then each tablespace has to be taken out of backup mode individually, or media recovery had to be performed on the database.

  **See Also:** "Ending a Backup After an Instance Failure or SHUTDOWN ABORT" on page 2-11

- **Trial Recovery**

  The SQL*Plus RECOVER . . . TEST statement can perform a trial recovery in memory without affecting the physical database. This feature enables you to test backup and recovery strategies without actually applying changes to the files on disk. Also, if you are troubleshooting media recovery problems, trial recovery lets you foresee what problems might occur if you were to continue with normal recovery.

  **See Also:** "Performing Trial Recovery" on page 5-8

- **Recovery Through Media Recovery Problems**

  Use the RECOVER statement with the ALLOW . . . CORRUPTION clause to permit recovery to corrupt blocks during datafile media recovery. After recovery completes, you can use RMAN to perform block media recovery on the corrupted blocks. Hence, this feature can shorten recovery time and increase database availability.

  **See Also:** Chapter 5, "Troubleshooting User-Managed Media Recovery"

- **Multiple Conversion Pairs for *_FILE_NAME_CONVERT Parameters**

  You can specify multiple conversion pairs in the DB_FILE_NAME_CONVERT and LOG_FILE_NAME_CONVERT initialization parameters.

- **LOG_ARCHIVE_DEST_n Supports Up to 10 Locations**

  The LOG_ARCHIVE_DEST_*n* initialization parameter can archive to up to 10 locations.

  **See Also:** "Listing Database Files Before a Backup" on page 2-2, and "About User-Managed Restore Operations" on page 3-2

# Oracle8*i* New Features in User-Managed Backup and Recovery

This section contains these topics:



## Oracle8*i* Release 2 (8.1.6) New Features in User-Managed Backup and Recovery

Oracle8*i* Release 2 (8.1.6) contains a number of internal improvements that provide more robust protection against data corruption.

- **Protection Against Logical Corruption**

  Logical data corruptions are typically caused by application errors and are difficult to repair because these corruptions are in the redo logs. You can prevent most logical corruptions by enabling block checking, which can detect and roll back changes that corrupt the database. Block checking is improved in these ways:

  – Oracle checks more block types, such as rollback segment blocks, transaction table blocks, and segment header blocks.

  – Block checking is more efficient, checking more blocks without increasing system overhead.

  – Block checking is always turned on for the SYSTEM tablespace, regardless of the setting of the DB_BLOCK_CHECKING initialization parameter.

- **Protection Against Memory Corruptions**

  If block checking is turned on, then the database writer process performs block checking immediately before writing a block to disk. This check enables Oracle to catch some corruptions when they are still in memory and automatically repair corrupted blocks before they are written to disk.

- **Protection Against Physical Data Corruption**

  Typically, Oracle detects physical I/O corruptions by storing a checksum in each data block. Oracle release 8.1.6 always performs checksum calculations in the SYSTEM tablespace, regardless of the DB_BLOCK_CHECKSUM parameter setting.

  If the calculated checksum does not match the stored checksum when Oracle reads the control file or redo logs, then Oracle rereads the data from either a different log or the same member in more situations than previous Oracle releases. Hence, Oracle has a second chance to find a good copy of the data and repair any physical data corruption.

## Oracle8*i* Release 1 (8.1.5) New Features in User-Managed Backup and Recovery

The following backup and recovery features are new in release 8.1.5:

- **Backups with the SUSPEND/RESUME Feature**

  You can temporarily suspend and then resume database operations without shutting down the database. While the database is suspended, you can make online backups of split mirrors.

  **See Also:** see "Making User-Managed Backups in SUSPEND Mode" on page 2-16

- **TSPITR Supports Transportable Tablespaces**

  You can use transportable tablespaces to perform tablespace point-in-time recovery (TSPITR).

  **See Also:** "Performing TSPITR with Transportable Tablespaces" on page 7-14

- **LOG_ARCHIVE_DEST_n Initialization Parameter**

  The LOG_ARCHIVE_DEST_*n* (where *n* is an integer from 1 to 5) initialization parameter can archive to up to 5 locations.

# 1

# Introduction to User-Managed Backup and Recovery

This chapter introduces database concepts that are fundamental to user-managed backup and recovery.

This chapter includes the following topics:

- About User-Managed Backup and Recovery
- Why Use User-Managed Backup and Recovery Methods?
- Overview of User-Managed Backup and Recovery

> **See Also:** *Oracle9i Backup and Recovery Concepts* for a conceptual overview of essential backup and recovery concepts

# About User-Managed Backup and Recovery

User-managed backup and recovery is any strategy in which Recovery Manager (RMAN) is not used as the principal backup and recovery tool. The basic user-managed backup strategy is to make periodic backups of datafiles and archived logs with operating system commands.

The basic user-managed media recovery procedure is as follows:

1. In the event of a media failure, restore database file backups with operating system commands.
2. Recover restored datafiles with the SQL*Plus RECOVER statement.
3. If the database is closed, then open it for normal use; if it is open, then bring the recovered tablespaces back online.

# Why Use User-Managed Backup and Recovery Methods?

Oracle Corporation recommends using RMAN as the foundation of an enterprise backup and recovery strategy, but user-managed methods (that is, methods that do not involve RMAN) are also just as effective. Note that some features such as block media recovery can only be performed with RMAN.

The following are possible circumstances in which you may choose to employ user-managed methods rather than use RMAN:

- You are migrating from an older version of the database to the current version and do not immediately want to update your legacy backup scripts.
- You maintain a network containing Oracle7 and later databases and want a single backup and recovery method to handle all databases in the same way. RMAN only supports Oracle databases of release 8.0 or greater.
- All your RMAN backup sets are destroyed and you are forced to restore user-managed backups and perform recovery with the SQL*Plus RECOVER command.

Oracle Corporation supports user-managed backup and recovery as a viable alternative to using RMAN.

# Overview of User-Managed Backup and Recovery

This section contains these topics:

- About User-Managed Restore and Recovery

## About User-Managed Backups

User-managed backups can be either logical or physical. You can use the Export utility to make backups of logical objects such as tables, views, and stored procedures, and use the Import utility to restore these objects.

If you do not use RMAN, then you can use operating system utilities to make physical backups. A physical backup is a backup of an Oracle database file or archived redo log located on the operating system. Note that these files can either be manually-managed database files or **Oracle-managed files**. If you use the Oracle Managed Files feature, then Oracle names the files for you and also deletes them for you when you drop a tablespace. From the point of view of backup and recovery, Oracle managed files are no different from user-managed files.

The following table illustrates the main types of physical backups and the non-RMAN methods for performing these backups.

| Backup Object | Backup Method | Example |
|---|---|---|
| Datafiles | Operating system utility | `% cp df3.f df3.bak` |
| Archived logs | Operating system utility | `% cp log_1_23.arc log_1_23.bak` |
| Control files | SQL statement | `SQL> ALTER DATABASE BACKUP CONTROLFILE TO cf1.bak` |
| Initialization parameter file | SQL statement | `SQL> CREATE PFILE = init.ora.bak FROM SPFILE;` |
| Network and password files | Operating system utility | `% cp tnsnames.ora tnsnames.bak`<br>`C:\> copy tnsnames.ora tnsnames.bak` |
| Logical objects (tables, indexes, PL/SQL units) | Export utility | `% export SYSTEM/manager TABLE=hr.emp FILE=emp.dmp` |

**See Also:**

- *Oracle9i Database Utilities* to learn how to use the Export and Import utilities
- *Oracle9i Database Administrator's Guide* to learn about Oracle Managed Files

### Basic Backup Methodology

The basic method for taking user-managed backups of the whole database is as follows:

1. Identify the datafiles, control files, and archived redo logs to be backed up by querying dynamic performance views or data dictionary tables (refer to "Querying V$ Views to Obtain Backup Information" on page 2-2 for procedures).
2. Use an operating system command such as the UNIX `cp` command to back up datafiles and archived redo logs (refer to "Making User-Managed Backups of the Whole Database" on page 2-4 for procedures).
3. Use a SQL statement to back up the control file (refer to "Making User-Managed Backups of the Control File" on page 2-19 for procedures).
4. Use an operating system command such as the UNIX `cp` command to back up configuration files (refer to"Making User-Managed Backups of Miscellaneous Oracle Files" on page 2-29 for procedures).

> **Caution:** Do not back up online redo logs. If you reset the online logs after media recovery, and then accidentally apply the backed up logs to the database, then you can corrupt the database.

### Consistent and Inconsistent User-Managed Backups

You can use RMAN or operating system commands to make an **inconsistent backup** or a **consistent backup**. An inconsistent backup is a backup of one or more database files made while the database is open or after the database has not been shut down normally. A consistent backup is a backup of one or more database files that you make after the database has been shut down normally. Unlike an inconsistent backup, a consistent backup does not require recovery after it is restored.

A consistent whole database backup is the only valid backup option for databases running in `NOARCHIVELOG` mode, because otherwise redo needed for recovery is not available. In `NOARCHIVELOG` mode, Oracle overwrites redo records without archiving them first.

If you run the database in `ARCHIVELOG` mode, then you can back up database files while the database is open. These backups are inconsistent, but as long as you have the necessary archived redo logs you can recover these backups. You can either take a tablespace offline and back up its datafiles, or perform an **online backup**. An

online backup occurs when the tablespace is still online. To perform an online backup, you must begin and end the backup with SQL statements that place the tablespace in and take the tablespace out of **backup mode**.

**See Also:**

- "Making Consistent Whole Database Backups" on page 2-4
- "Making User-Managed Backups of Offline Tablespaces and Datafiles" on page 2-6
- "Making User-Managed Backups of Online Tablespaces and Datafiles" on page 2-7

### Backups in SUSPEND Mode

Some third-party tools allow you to mirror a set of disks or logical devices, that is, maintain an exact duplicate of the primary data in another location, and then **split the mirror**. Splitting the mirror involves separating the copies so that you can use them independently.

Using the SUSPEND/RESUME functionality, you can suspend I/O to the database, then split the mirror and make a backup of the split mirror. By using this feature, which complements the online backup functionality, you can quiesce the database so that no new I/O can be performed. You can then access the suspended database to make backups without I/O interference.

**See Also:** "About the Suspend/Resume Feature" on page 2-16

### Verification of Backups

The best method for backup verification is to perform a test restore and recover of the database to another location. If you successfully perform this operation, then you know that the backup is valid.

You can also use the DBVERIFY utility to test backups for corruption. DBVERIFY is an external command-line utility that performs a physical data structure integrity check on offline datafiles. Use DBVERIFY primarily when you need to ensure that a backup datafile is valid before it is restored or as a diagnostic aid when you have encountered data corruption problems. The name and location of DBVERIFY is dependent on your operating system (for example, dbv on Sun/Sequent systems).

**See Also:** "Verifying User-Managed Backups" on page 2-27

## About User-Managed Restore and Recovery

When a media failure occurs that damages datafiles, you must restore backups of the affected datafiles using operating system commands and then perform recovery with the SQL*Plus RECOVER command. You can either restore only some datafiles and perform recovery of the tablespaces containing the restored datafiles, or restore and recover the entire database. You should keep careful records of your backups so that you know the original locations of the datafiles as well as the locations of the backups.

To begin media recovery operations when your database is running in ARCHIVELOG mode, use the SQL*Plus RECOVER command. The two basic types of media recovery are **complete recovery**, in which all redo generated on the database is applied, and **incomplete recovery**, in which not all the existing redo is applied. Incomplete recovery is only valid for restore and recovery of the entire database. A special procedure for performing incomplete recovery of an individual tablespace is called **tablespace point-in-time recovery (TSPITR)**.

### Basic Restore and Recovery Methodology

The basic user-managed restore and recovery strategy is as follows:

1. Determine what you need to restore and recover (refer to "Determining Which Datafiles Require Recovery" on page 3-5 for procedures).
2. Restore backups of files permanently damaged by media failure by using an operating system utility. If you cannot restore a datafile to its original location, then relocate the restored datafile and change the location in the control file (refer to "Restoring Datafiles" on page 3-6 for procedures).
3. Restore any necessary archived redo log files with an operating system utility (refer to "Restoring Archived Redo Logs" on page 3-15 for procedures).
4. Use the SQL*Plus RECOVER command to recover the files, as described in "Performing User-Managed Media Recovery: Overview" on page 4-2.

### Implications of the Archiving Mode for Media Recovery

The archiving mode of the database determines the type of recovery that you can perform. For example, if a database is in NOARCHIVELOG mode and a media failure damages some or all of the datafiles, then usually the only option for recovery is to restore the most recent consistent, whole database backup and open it.

The disadvantage of NOARCHIVELOG mode is that to recover the database from the time of the most recent full backup up to the time of the media failure, you have to reenter manually all of the changes executed in that interval. If your database is in

ARCHIVELOG mode, and the redo logs covering this interval are available as archived log files or online log files, then you can use complete or incomplete recovery to reconstruct your database, thereby minimizing the number of lost changes.

### User-Managed Tablespace Point-in-Time Recovery (TSPITR)

User-managed tablespace point-in-time recovery (TSPITR) enables you to quickly recover one or more tablespaces (other than the SYSTEM tablespace) to a time that is different from that of the rest of the database.

User-managed TSPITR is most useful for recovering:

- An erroneous DROP TABLE or TRUNCATE TABLE operation.
- A table that is logically corrupted.
- An incorrect batch job or other DML statement that has affected only a subset of the database.
- A logical schema to a point different from the rest of the physical database when multiple schemas exist in separate tablespaces of one physical database.
- A tablespace in a VLDB (very large database) when TSPITR is more efficient than restoring the whole database from a backup and rolling it forward (see "Preparing for Tablespace Point-in-Time Recovery: Basic Steps" on page 7-4 before making any decisions).

**See Also:** Chapter 7, "Performing User-Managed TSPITR"

# 2

# Making User-Managed Backups

If you do not use Recovery Manager (RMAN), then you can make backups of your database files using user-managed methods.

This chapter contains the following sections:

# Querying V$ Views to Obtain Backup Information

Before making a backup, identify all the files in your database. Then, ascertain what you need to back up.

This section contains these topics:

- Listing Database Files Before a Backup
- Determining Datafile Status for Online Tablespace Backups

## Listing Database Files Before a Backup

Before beginning a backup, query the database to determine which files you should back up. Note that backups of Oracle Managed Files are not different from backups of database files that you name manually.

**To list datafiles, online redo logs, and control files:**

1. Start SQL*Plus and query V$DATAFILE to obtain a list of datafiles. For example, enter:

   ```
   SQL> SELECT NAME FROM V$DATAFILE;
   ```

   You can also join the V$TABLESPACE and V$DATAFILE views to obtain a listing of datafiles along with their associated tablespaces:

   ```
   SELECT t.NAME "Tablespace", f.NAME "Datafile"
     FROM V$TABLESPACE t, V$DATAFILE f
     WHERE t.TS# = f.TS#
     ORDER BY t.NAME;
   ```

2. Obtain the filenames of online redo log files by querying the V$LOGFILE view. For example, issue the following query:

   ```
   SQL> SELECT MEMBER FROM V$LOGFILE;
   ```

3. Obtain the filenames of the current control files by querying the V$CONTROLFILE view. For example, issue the following query:

   ```
   SQL> SELECT NAME FROM V$CONTROLFILE;
   ```

   Note that you only need to back up one copy of a multiplexed control file.

4. If you plan to take a control file backup with the ALTER DATABASE BACKUP CONTROLFILE TO '*filename*' statement, then save a list of all datafiles and online redo log files with the control file backup. Because the current database structure may not match the database structure at the time a given control file

backup was created, saving a list of files recorded in the backup control file can aid the recovery procedure.

## Determining Datafile Status for Online Tablespace Backups

To check whether a datafile is part of a current online tablespace backup, query the V$BACKUP view. This view is useful only for user-managed online tablespace backups, not offline tablespace backups or RMAN backups.

The V$BACKUP view 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. Use this information to determine whether you have left any tablespaces in backup mode.

V$BACKUP is not useful if the control file currently in use is a restored backup or a new control file created after the media failure occurred. A restored or re-created control file does not contain the information Oracle needs to fill V$BACKUP accurately. Also, if you have restored a backup of a file, this 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 can contain misleading data about restored files.

For example, the following query displays which datafiles are currently included in a tablespace that has been placed in backup mode:

```
SELECT t.name AS "TB_NAME", d.file# as "DF#", d.name AS "DF_NAME", b.status
FROM V$DATAFILE d, V$TABLESPACE t, V$BACKUP b
WHERE d.TS#=t.TS#
AND b.FILE#=d.FILE#
AND b.STATUS='ACTIVE'
/
```

Sample output follows:

```
TB_NAME              DF#        DF_NAME                STATUS
-------------------- ---------- ---------------------  ------------------
TBS_1                3          /oracle/dbs/tbs_11.f       ACTIVE
TBS_1                4          /oracle/dbs/tbs_12.f       ACTIVE
```

In the STATUS column, NOT ACTIVE indicates that the file is not currently in backup mode (that is, ALTER TABLESPACE . . . BEGIN BACKUP), whereas ACTIVE indicates that the file is currently in backup mode.

# Making User-Managed Backups of the Whole Database

You can make a whole database backup of all files in a database after the database has been shut down with the NORMAL, IMMEDIATE, or TRANSACTIONAL options. A whole database backup taken while the database is open or after an instance crash or SHUTDOWN ABORT is inconsistent. In such cases, the files are inconsistent with respect to the checkpoint SCN.

You can make a whole database backup if a database is operating in either ARCHIVELOG or NOARCHIVELOG mode. If you run the database in NOARCHIVELOG mode, however, the backup must be consistent; that is, you must shut down the database cleanly before the backup.

The set of backup files that results from a consistent whole database backup is consistent because all files are checkpointed to the same SCN. You can restore the consistent database backup without performing recovery. After restoring the backup files, you can 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.

Control files play a crucial role in database restore and recovery. For databases running in ARCHIVELOG mode, Oracle recommends that you back up control files with the ALTER DATABASE BACKUP CONTROLFILE TO '*filename*' statement. If you back up the control file with an operating system utility during a closed, consistent whole database backup, then you should only use this control file when restoring the other datafiles taken in the backup. Although a control file backed up with an operating system utility during a consistent backup can sometimes be used for recovery (but only if you specify the USING BACKUP CONTROLFILE clause of the RECOVER statement), Oracle does not recommend this practice because neglecting to specify the USING BACKUP CONTROLFILE clause can cause recovery problems.

> **See Also:** "Making User-Managed Backups of the Control File" on page 2-19 for more information about backing up control files.

## Making Consistent Whole Database Backups

To guarantee that a database's datafiles are consistent, shut down the database with the NORMAL, IMMEDIATE, or TRANSACTIONAL options before making a whole database backup.

> **Caution:** If the database is in NOARCHIVELOG mode, then never perform a whole database backup after an instance fails or is aborted. This backup is inconsistent and requires recovery to be made consistent, so unless the needed redo exists in the online redo logs and these logs are intact, the backup is unusable.

**To make a consistent whole database backup:**

1. If the database is open, use SQL*Plus to shut down the database with the NORMAL, IMMEDIATE, or TRANSACTIONAL options. For example, do one of the following:

   ```
   SQL> SHUTDOWN NORMAL
   SQL> SHUTDOWN IMMEDIATE
   SQL> SHUTDOWN TRANSACTIONAL
   ```

   Do not make a whole database backup when the instance is aborted or stopped because of a failure. If possible, reopen the database and shut it down cleanly.

2. Use an operating system utility to make backups of all datafiles as well as all control files specified by the CONTROL_FILES parameter of the initialization parameter file. Also, back up the initialization parameter file and other Oracle product initialization files. To find these files, do a search for *.ora starting in your Oracle home directory and recursively search all of its subdirectories.

   > **Note:** If you are forced to perform a restore operation, you must restore the control files to all locations specified in the initialization parameter file. Hence, it is better to make copies of each multiplexed control file—even if the control files are identical—to avoid problems at restore time.

   For example, you can back up the datafiles and control files in the /disk1/oracle/dbs directory to /disk2/backup as follows:

   ```
   % cp /disk1/oracle/dbs/*.dbf /disk2/backup
   % cp /disk1/oracle/dbs/*.cf /disk2/backup
   % cp /disk1/oracle/network/admin/*.ora /disk2/backup
   % cp /disk1/oracle/rdbms/admin/*.ora /disk2/backup
   ```

3. Restart the database. For example, enter:

   ```
   SQL> STARTUP
   ```

**See Also:** *Oracle9i Database Administrator's Guide* for more information on starting up and shutting down a database.

# Making User-Managed Backups of Offline Tablespaces and Datafiles

You can back up all or some of the datafiles of an individual tablespace while the tablespace is offline. All other tablespaces of the database can remain open and available for systemwide use. You must have the DBA privilege or have the MANAGE TABLESPACE system privilege to take tablespaces offline and online.

Note these guidelines when backing up offline tablespaces:

- You cannot offline the SYSTEM tablespace or a tablespace with active rollback segments. The following procedure cannot be used for such tablespaces.
- Assume that a table is in tablespace `Primary` and its index is in tablespace `Index`. Taking tablespace `Index` offline while leaving tablespace `Primary` online can cause errors when DML is issued against the indexed tables located in `Primary`. The problem only manifests when the access method chosen by the optimizer needs to access the indexes in the `Index` tablespace.

**To back up offline tablespaces:**

1. Before beginning a backup of a tablespace, identify the tablespace's datafiles by querying the DBA_DATA_FILES view. For example, assume that you want to back up the `users` tablespace. Enter the following in SQL*Plus:

```
SELECT TABLESPACE_NAME, FILE_NAME
  FROM SYS.DBA_DATA_FILES
  WHERE TABLESPACE_NAME = 'users';

TABLESPACE_NAME                FILE_NAME
------------------------------ ------------------
users                          /oracle/dbs/users.f
```

   In this example, `/oracle/dbs/users.f` is a fully specified filename corresponding to the datafile in the `users` tablespace.

2. Take the tablespace offline using normal priority if possible. Normal priority is recommended because it guarantees that you can subsequently bring the tablespace online without the requirement for tablespace recovery. For example, the following statement takes a tablespace named `users` offline normally:

```
SQL> ALTER TABLESPACE users OFFLINE NORMAL;
```

   After you take a tablespace offline with normal priority, all datafiles of the tablespace are closed.

3. Back up the offline datafiles. For example, a UNIX user might enter the following to back up the datafile `users.f`:

```
% cp /disk1/oracle/dbs/users.f /disk2/backup/users.backup
```

4. Bring the tablespace online. For example, the following statement brings tablespace `users` back online:

```
ALTER TABLESPACE users ONLINE;
```

> **Note:** If you took the tablespace offline using temporary or immediate priority, then you cannot bring the tablespace online unless you perform tablespace recovery.

After you bring a tablespace online, it is open and available for use.

5. Archive the unarchived redo logs so that the redo required to recover the tablespace backup is archived. For example, enter:

```
ALTER SYSTEM ARCHIVE LOG CURRENT;
```

## Making User-Managed Backups of Online Tablespaces and Datafiles

You can back up all or only specific datafiles of an online tablespace while the database is open. The procedure differs depending on whether the online tablespace is read / write or read-only.

> **Note:** You should not back up temporary tablespaces.

This section contains these topics:

- Making User-Managed Backups of Online Read / Write Tablespaces
- Making Multiple User-Managed Backups of Online Read / Write Tablespaces
- Ending a Backup After an Instance Failure or SHUTDOWN ABORT
- Making User-Managed Backups of Read-Only Tablespaces
- Making User-Managed Backups of Undo Tablespaces

## Making User-Managed Backups of Online Read/Write Tablespaces

You must put a read/write tablespace in backup mode to make user-managed datafile backups when the tablespace is online and the database is open. The ALTER TABLESPACE BEGIN BACKUP statement places a tablespace in backup mode.

Oracle stops recording checkpoints to the datafiles in the tablespace when a tablespace is in backup mode. Because a block can be partially updated at the very moment that the operating system backup utility is copying it, Oracle copies whole changed data blocks into the redo stream while in backup mode. After you take the tablespace out of backup mode with the ALTER TABLESPACE . . . END BACKUP or ALTER DATABASE END BACKUP statement, Oracle advances the datafile header to the current database checkpoint.

When you restore a datafile backed up in this way, the datafile header has a record 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. The redo logs contain all changes required to recover the datafiles and make them consistent.

**To back up online read/write tablespaces in an open database:**

1. Before beginning a backup of a tablespace, identify all of the datafiles in the tablespace with the DBA_DATA_FILES data dictionary view. For example, assume that you want to back up the `users` tablespace. Enter the following:

```
SELECT TABLESPACE_NAME, FILE_NAME
FROM SYS.DBA_DATA_FILES
WHERE TABLESPACE_NAME = 'users';

TABLESPACE_NAME                FILE_NAME
-------------------------------  --------------------
USERS                          /oracle/dbs/tbs_21.f
USERS                          /oracle/dbs/tbs_22.f
```

   In this example, `/oracle/dbs/tbs_21.f` and `/oracle/dbs/tbs_22.f` are fully specified filenames corresponding to the datafiles of the `users` tablespace.

2. Mark the beginning of the online tablespace backup. For example, the following statement marks the start of an online backup for the tablespace `users`:

```
SQL> ALTER TABLESPACE users BEGIN BACKUP;
```

**Caution:** If you forget to mark the beginning of an online tablespace backup, or neglect to assure that the BEGIN BACKUP statement has completed before backing up an online tablespace, then the backup datafiles are not useful for subsequent recovery operations. Attempting to recover such a backup is risky and can return errors that result in inconsistent data. For example, the attempted recovery operation can issue a "fuzzy files" warning, and can lead to an inconsistent database that you cannot open.

3. Back up the online datafiles of the online tablespace with operating system commands. For example, UNIX users might enter:

```
% cp /oracle/dbs/tbs_21.f /oracle/backup/tbs_21.backup
% cp /oracle/dbs/tbs_22.f /oracle/backup/tbs_22.backup
```

4. After backing up the datafiles of the online tablespace, indicate the end of the online backup by using the SQL statement ALTER TABLESPACE with the END BACKUP option. For example, the following statement ends the online backup of the tablespace `users`:

```
SQL> ALTER TABLESPACE users END BACKUP;
```

5. Archive the unarchived redo logs so that the redo required to recover the tablespace backup is archived. For example, enter:

```
SQL> ALTER SYSTEM ARCHIVE LOG CURRENT;
```

**Caution:** If you forget to take the tablespace out of backup mode, then Oracle continues to write entire copies of data blocks in this tablespace to the online redo logs, possibly causing performance problems. Also, you will receive an `ORA-01149` error if you attempt to shut down the database with the tablespaces still in backup mode.

## Making Multiple User-Managed Backups of Online Read/Write Tablespaces

When backing up several online tablespaces, you can back them up either serially or in parallel. Use either of the following procedures depending on your needs.

### Backing Up Online Tablespaces in Parallel

You can simultaneously put all tablespaces requiring backups in backup mode. Note that online redo logs can grow large if multiple users are updating these tablespaces because the redo must contain a copy of each changed data block.

**To back up online tablespaces in parallel:**

1. Prepare all online tablespaces for backup by issuing all necessary ALTER TABLESPACE statements at once. For example, put tablespaces `ts1`, `ts2`, and `ts3` in backup mode as follows:

   ```
   SQL> ALTER TABLESPACE ts1 BEGIN BACKUP;
   SQL> ALTER TABLESPACE ts2 BEGIN BACKUP;
   SQL> ALTER TABLESPACE ts3 BEGIN BACKUP;
   ```

2. Back up all files of the online tablespaces. For example, a UNIX user might back up datafiles with the `tbs_` prefix as follows:

   ```
   % cp /oracle/dbs/tbs_* /oracle/backup
   ```

3. Take the tablespaces out of backup mode as in the following example:

   ```
   SQL> ALTER TABLESPACE ts1 END BACKUP;
   SQL> ALTER TABLESPACE ts2 END BACKUP;
   SQL> ALTER TABLESPACE ts3 END BACKUP;
   ```

4. Archive the unarchived redo logs so that the redo required to recover the tablespace backups is archived. For example, enter:

   ```
   SQL> ALTER SYSTEM ARCHIVE LOG CURRENT;
   ```

### Backing Up Online Tablespaces Serially

You can place all tablespaces requiring online backups in backup mode one at a time. Oracle Corporation recommends the serial backup option because it minimizes the time between ALTER TABLESPACE . . . BEGIN/END BACKUP statements. During online backups, more redo information is generated for the tablespace because whole data blocks are copied into the redo log.

**To back up online tablespaces serially:**

1. Prepare a tablespace for online backup. For example, to put tablespace `tbs_1` in backup mode enter the following:

   ```
   SQL> ALTER TABLESPACE tbs_1 BEGIN BACKUP;
   ```

2. Back up the datafiles in the tablespace. For example, enter:

   ```
   % cp /oracle/dbs/tbs_1.f /oracle/backup/tbs_1.bak
   ```

3. Take the tablespace out of backup mode. For example, enter:

```
SQL> ALTER TABLESPACE tbs_1 END BACKUP;
```

4. Repeat this procedure for each remaining tablespace until you have backed up all the desired tablespaces.
5. Archive the unarchived redo logs so that the redo required to recover the tablespace backups is archived. For example, enter:

```
SQL> ALTER SYSTEM ARCHIVE LOG CURRENT;
```

## Ending a Backup After an Instance Failure or SHUTDOWN ABORT

This section contains these topics:



### About Instance Failures When Tablespaces are in Backup Mode

The following situations can cause a tablespace backup to fail and be incomplete:

- The backup completed, but you did not indicate the end of the online tablespace backup operation with the ALTER TABLESPACE . . . END BACKUP statement.
- An instance failure or SHUTDOWN ABORT interrupted the backup before you could complete it.

Whenever crash recovery is required (not instance recovery, because in this case the datafiles are open already), if a datafile is in backup mode when an attempt is made to open it, then the system assumes that the file is a restored backup. Oracle will not open the database until either a recovery command is issued, or the datafile is taken out of backup mode.

For example, Oracle may display a message such as the following when you run the STARTUP statement:

```
ORA-01113: file 12 needs media recovery
ORA-01110: data file 12: '/oracle/dbs/tbs_41.f'
```

If Oracle indicates that the datafiles for multiple tablespaces require media recovery because you forgot to end the online backups for these tablespaces, then so long as

the database is mounted, running the ALTER DATABASE END BACKUP statement takes all the datafiles out of backup mode simultaneously.

In high availability situations, and in situations when no DBA is monitoring the database (for example, in the early morning hours), the requirement for user intervention is intolerable. Hence, you can write a crash recovery script that does the following:

1. Mounts the database
2. Runs the ALTER DATABASE END BACKUP statement
3. Runs ALTER DATABASE OPEN, allowing the system to come up automatically

An automated crash recovery script containing ALTER DATABASE END BACKUP is especially useful in the following situations:

- All nodes in an Oracle Real Application Clusters configuration crash.
- One node crashes in a **cold failover cluster** (that is, a cluster that is *not* an Oracle Real Application Cluster in which the secondary node must mount and recover the database when the first node crashes).

Alternatively, you can take the following manual measures after the system crashes with tablespaces in backup mode:

- Recover the database and avoid issuing END BACKUP statements altogether.
- Mount the database, then run ALTER TABLESPACE . . . END BACKUP for each tablespace still in backup mode.

### Ending Backup Mode with the ALTER DATABASE END BACKUP Statement

You can run the ALTER DATABASE END BACKUP statement when you have multiple tablespaces still in backup mode. The primary purpose of this command is to allow a crash recovery script to restart a failed system without DBA intervention. You can also perform the following procedure manually.

**To take tablespaces out of backup mode simultaneously:**

1. Mount but do not open the database. For example, enter:

   ```
   SQL> STARTUP MOUNT
   ```

2. If performing this procedure manually (that is, not as part of a crash recovery script), query the V$BACKUP view to list the datafiles of the tablespaces that were being backed up before the database was restarted:

   ```
   SQL>  SELECT * FROM V$BACKUP WHERE STATUS = 'ACTIVE';
   ```

```
FILE#      STATUS             CHANGE#    TIME
---------- ------------------ ---------- ---------
        12 ACTIVE                  20863 25-NOV-00
        13 ACTIVE                  20863 25-NOV-00
        20 ACTIVE                  20863 25-NOV-00
 3 rows selected.
```

3. Issue the ALTER DATABASE END BACKUP statement to take all datafiles currently in backup mode out of backup mode. For example, enter:

   ```
   SQL> ALTER DATABASE END BACKUP;
   ```

   You can use this statement only when the database is mounted but not open. If the database is open, use ALTER TABLESPACE . . . END BACKUP or ALTER DATABASE DATAFILE . . . END BACKUP for each affected tablespace or datafile.

> **Caution:** Do not use ALTER DATABASE END BACKUP if you have restored any of the affected files from a backup.

### Ending Backup Mode with the RECOVER Command

The ALTER DATABASE END BACKUP statement is not the only way to respond to a failed online backup: you can also run the RECOVER command. This method is useful when you are not sure whether someone has restored a backup, because if someone has indeed restored a backup, then the RECOVER command brings the backup up to date. Only run the ALTER DATABASE END BACKUP or ALTER TABLESPACE . . . END BACKUP statement if you are sure that the files are current.

> **Note:** The RECOVER command method is slow because Oracle must scan redo generated from the beginning of the online backup.

**To take tablespaces out of backup mode with the RECOVER command:**

1. Mount the database. For example, enter:

   ```
   SQL> STARTUP MOUNT
   ```

2. Recover the database as normal. For example, enter:

   ```
   SQL> RECOVER DATABASE
   ```

3. Use the V$BACKUP view to confirm that there are no active datafiles:

   ```
   SQL>  SELECT * FROM V$BACKUP WHERE STATUS = 'ACTIVE';
   ```

```
FILE#      STATUS            CHANGE#    TIME
---------- ----------------- ---------- ---------
0 rows selected.
```

> **See Also:** Chapter 4, "Performing User-Managed Media Recovery" for information on recovering a database.

## Making User-Managed Backups of Read-Only Tablespaces

When backing up an online read-only tablespace, you can simply back up the online datafiles. You do not have to place the tablespace in backup mode because the system is permitting changes to the datafiles.

If the set of read-only tablespaces is self-contained, then in addition to backing up the tablespaces with operating system commands, you can also export the tablespace metadata by using the transportable tablespace functionality. In the event of a media error or a user error (such as accidentally dropping a table in the read-only tablespace), you can transport the tablespace back into the database.

> **See Also:** *Oracle9i Database Administrator's Guide* to learn how to transport tablespaces

**To back up online read-only tablespaces in an open database:**

1. Query the DBA_TABLESPACES view to determine which tablespaces are read-only. For example, run this query:

```
SELECT TABLESPACE_NAME, STATUS
FROM DBA_TABLESPACES
WHERE STATUS = 'READ ONLY';
```

2. Before beginning a backup of a read-only tablespace, identify all of the tablespace's datafiles by querying the DBA_DATA_FILES data dictionary view. For example, assume that you want to back up the `history` tablespace. Enter the following:

```
SELECT TABLESPACE_NAME, FILE_NAME
FROM SYS.DBA_DATA_FILES
WHERE TABLESPACE_NAME = 'HISTORY';

TABLESPACE_NAME                FILE_NAME
------------------------------ --------------------
HISTORY                        /oracle/dbs/tbs_hist1.f
HISTORY                        /oracle/dbs/tbs_hist2.f
```

In this example, `/oracle/dbs/tbs_hist1.f` and `/oracle/dbs/tbs_hist2.f` are fully specified filenames corresponding to the datafiles of the `history` tablespace.

3. Back up the online datafiles of the read-only tablespace with operating system commands. You do not have to take the tablespace offline or put the tablespace in backup mode because users are automatically prevented from making changes to the read-only tablespace. For example, UNIX users can enter:

```
% cp /oracle/dbs/tbs_hist*.f /backup
```

> **Note:** When restoring a backup of a read-only tablespace, take the tablespace offline, restore the datafiles, then bring the tablespace online. A backup of a read-only tablespace is still usable if the read-only tablespace is made read / write after the backup, but the restored backup will require recovery.

4. Optionally, export the metadata in the read-only tablespace. By using the transportable tablespace feature, you can quickly restore the datafiles and import the metadata in case of media failure or user error. For example, export the metadata for tablespace `history` as follows:

```
% exp TRANSPORT_TABLESPACE=y TABLESPACES=(history) FILE=/oracle/backup/tbs_hist.dmp
```

> **See Also:** *Oracle9i Database Reference* for more information about the `DBA_DATA_FILES` and `DBA_TABLESPACES` views

## Making User-Managed Backups of Undo Tablespaces

In releases prior to Oracle9*i*, undo space management was based on rollback segments. This method is called **manual undo management mode**. In Oracle9*i*, you have the option of placing the database in **automatic undo management mode**. With this design, you allocate undo space in a single undo tablespace instead of distributing space into a set of statically allocated rollback segments.

The procedures for backing up undo tablespaces are exactly the same as for backing up any other read / write tablespace. Because the automatic undo tablespace is so important for recovery and for read consistency, you should back it up frequently as you would for tablespaces containing rollback segments when running in manual undo management mode.

If the datafiles in the undo tablespace were lost while the database was open, and you did not have a backup, you could receive error messages when querying

objects containing uncommitted changes. Also, if an instance failure occurred, you would not be able to roll back uncommitted transactions to their original values.

> **See Also:** *Oracle9i Database Administrator's Guide* to learn how to manage undo space

# Making User-Managed Backups in SUSPEND Mode

This section contains the following topics:

- About the Suspend/Resume Feature
- Making Backups in a Suspended Database

## About the Suspend/Resume Feature

Some third-party tools allow you to mirror a set of disks or logical devices, that is, maintain an exact duplicate of the primary data in another location, and then **split the mirror**. Splitting the mirror involves separating the copies so that you can use them independently.

With the SUSPEND/RESUME functionality, you can suspend I/O to the database, then split the mirror and make a backup of the split mirror. By using this feature, which complements the backup mode functionality, you can suspend database I/Os so that no new I/O can be performed. You can then access the suspended database to make backups without I/O interference.

You do not need to use SUSPEND/RESUME to make split mirror backups in most cases, although it is necessary if your system requires the database cache to be free of **dirty buffers** before a volume can be split.

> **Note:** Some RAID devices benefit from suspending writes while the split operation is occurring; your RAID vendor can advise you on whether your system would benefit from this feature.

The ALTER SYSTEM SUSPEND statement suspends the database by halting I/Os to datafile headers, datafiles, and control files. When the database is suspended, all pre-existing I/O operations can complete; however, any new database I/O access attempts are queued.

The ALTER SYSTEM SUSPEND and ALTER SYSTEM RESUME statements operate on the database and not just the instance. If the ALTER SYSTEM SUSPEND statement is

entered on one system in an Oracle Real Application Clusters configuration, then the internal locking mechanisms propagate the halt request across instances, thereby suspending I/O operations for all active instances in a given cluster.

## Making Backups in a Suspended Database

After a successful database suspension, you can back up the database to disk or break the mirrors. Because suspending a database does not guarantee immediate termination of I/O, Oracle recommends that you precede the ALTER SYSTEM SUSPEND statement with a BEGIN BACKUP statement so that the tablespaces are placed in backup mode.

You must use conventional user-managed backup methods to back up split mirrors. RMAN cannot make database backups or copies because these operations require reading the datafile headers. After the database backup is finished or the mirrors are re-silvered, then you can resume normal database operations using the ALTER SYSTEM RESUME statement.

Backing up a suspended database without splitting mirrors can cause an extended database outage because the database is inaccessible during this time. If backups are taken by splitting mirrors, however, then the outage is nominal. The outage time depends on the size of cache to flush, the number of datafiles, and the time required to break the mirror.

Note the following restrictions for the SUSPEND/RESUME feature:

- In an Oracle Real Application Clusters configuration, you should not start a new instance while the original nodes are suspended.
- No checkpoint is initiated by the ALTER SYSTEM SUSPEND or ALTER SYSTEM RESUME statements.
- You cannot issue SHUTDOWN with IMMEDIATE or NORMAL options while the database is suspended.
- Issuing SHUTDOWN ABORT on a database that was already suspended reactivates the database. This operation prevents media recovery or crash recovery from hanging.

**To make a split mirror backup in SUSPEND mode:**

1. Place the database tablespaces in backup mode. For example, to place tablespace users in backup mode enter:

```
ALTER TABLESPACE users BEGIN BACKUP;
```

2. If your mirror system has problems with splitting a mirror while disk writes are occurring, then suspend the database. For example, issue the following:

```
ALTER SYSTEM SUSPEND;
```

3. Check to make sure that the database is suspended by querying V$INSTANCE. For example:

```
SELECT DATABASE_STATUS FROM V$INSTANCE;

DATABASE_STATUS
-----------------
SUSPENDED
```

4. Split the mirrors at the operating system or hardware level.
5. End the database suspension. For example, issue the following statement:

```
ALTER SYSTEM RESUME;
```

6. Check to make sure that the database is active by querying V$INSTANCE. For example, enter:

```
SELECT DATABASE_STATUS FROM V$INSTANCE;

DATABASE_STATUS
-----------------
ACTIVE
```

7. Take the specified tablespaces out of backup mode. For example, enter the following to take tablespace users out of backup mode:

```
ALTER TABLESPACE users END BACKUP;
```

8. Copy the control file and archive the online redo logs as usual for a backup.

---

**Caution:** Do not use the ALTER SYSTEM SUSPEND statement as a substitute for placing a tablespace in backup mode.

---

**See Also:** *Oracle9i Database Administrator's Guide* for more information about the SUSPEND / RESUME feature, and *Oracle9i SQL Reference* for more information about the ALTER SYSTEM statement

# Making User-Managed Backups of the Control File

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 have these options when backing up the control file:

- Backing Up the Control File to a Binary File
- Backing Up the Control File to a Trace File

## Backing Up the Control File to a Binary File

The primary method for backing up the control file is to use a SQL statement to generate a binary file. A binary backup is preferable to a trace file backup because it contains additional information such as the archived log history, offline range for read-only and offline tablespaces, and backup sets and copies (if you use RMAN). Note that binary control file backups do not include tempfile entries.

**To back up the control file after a structural change:**

1. Make the desired change to the database. For example, you may create a new tablespace:

   ```
   CREATE TABLESPACE tbs_1 DATAFILE 'file_1.f' SIZE 10M;
   ```

2. Back up the database's control file, specifying a filename for the output binary file. The following SQL statement backs up a database's control file to /oracle/backup/cf.bak:

   ```
   ALTER DATABASE BACKUP CONTROLFILE TO '/oracle/backup/cf.bak' REUSE;
   ```

   You can specify the REUSE option to make the new control file overwrite a control file that currently exists.

## Backing Up the Control File to a Trace File

The TRACE option of the ALTER DATABASE BACKUP CONTROLFILE statement helps you manage and recover the control file. The TRACE option prompts Oracle to write SQL statements to the database's trace file rather than generate a binary backup. The statements in the trace file start the database, re-create the control file, and recover and open the database appropriately.

To back up the control file to a trace file, mount or open the database and issue the following SQL statement:

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

If you specify neither the RESETLOGS nor NORESETLOGS option in the SQL statement, then the output is a trace file containing a CREATE CONTROLFILE ... NORESETLOGS statement. As in the case of binary control file backups, tempfile entries are not included in the trace output.

> **See Also:** "Recovery of Read-Only Files with a Re-Created Control File" on page 4-36 for special issues relating to read-only, offline normal, and temporary files included in CREATE CONTROLFILE statements

### Backing Up the Control File to a Trace File: Example

Assume that you want to generate a script that re-creates the control file for the sales database. The database has these characteristics:

- Three threads are enabled, of which thread 2 is public and thread 3 is private.
- The redo logs are multiplexed into three groups of two members each.
- The database has the following datafiles:
  - /diska/prod/sales/db/filea.dbf (offline datafile in online tablespace)
  - /diska/prod/sales/db/database1.dbf (online in SYSTEM tablespace)
  - /diska/prod/sales/db/fileb.dbf (only file in read-only tablespace)

You issue the following statement to create a trace file containing a CREATE CONTROLFILE ... NORESETLOGS statement:

```
ALTER DATABASE BACKUP CONTROLFILE TO TRACE NORESETLOGS;
```

You then edit the trace file to create a script that creates a new control file for the sales database based on the control file that was current when you generated the trace file. To avoid recovering offline normal or read-only tablespaces, edit them out of the CREATE CONTROLFILE statement in the trace file. When you open the database with the re-created control file, the dictionary check code will mark these omitted files as MISSING. You can run an ALTER DATABASE RENAME FILE statement renames them back to their original filenames.

For example, you can edit the CREATE CONTROLFILE . . . NORESETLOGS script in the trace file as follows, renaming files labeled MISSING:

```
# The following statements 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 datafiles. 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'
;

# This datafile is offline, but its tablespace is online. Take the datafile offline
# manually.
ALTER DATABASE DATAFILE '/diska/prod/sales/db/filea.dbf' OFFLINE;

# Recovery is required if any datafiles are restored backups,
# or if the most recent shutdown was not normal or immediate.
RECOVER DATABASE;

# All redo logs need archiving and a log switch is needed.
ALTER SYSTEM ARCHIVE LOG ALL;

# The database can now be opened normally.
ALTER DATABASE OPEN;

# The backup control file does not list read-only and normal offline tablespaces so that
# Oracle can avoid performing recovery on them. Oracle checks the data dictionary and
```

```
# finds information on these absent files and marks them 'MISSINGxxxx'. It then renames
# the missing files to acknowledge them without having to recover them.
ALTER DATABASE RENAME FILE 'MISSING0002'
     TO '/diska/prod/sales/db/fileb.dbf';
```

# Making User-Managed Backups of Archived Redo Logs

To save disk space in your primary archiving location, you may want to back up archived logs to tape or to an alternative disk location. If you archive to multiple locations, then only back up one copy of each log sequence number.

**To back up archived redo logs:**

1. To determine which archived redo log files that the database has generated, query V$ARCHIVED_LOG. For example, run the following query:

   ```
   SELECT THREAD#,SEQUENCE#,NAME
   FROM V$ARCHIVED_LOG;
   ```

2. Back up one copy of each log sequence number by using an operating system utility. This example backs up all logs in the primary archiving location to a disk devoted to log backups:

   ```
   % cp /oracle/dbs/arc_dest/* /disk7/log_backups
   ```

   **See Also:** *Oracle9i Database Reference* for more information about the data dictionary views.

# Making User-Managed Backups to Raw Devices

A **raw device** is a disk or partition that does not have a file system. In other words, a raw device can contain only a single file. Backing up files on raw devices poses operating system specific issues. The following sections discuss some of these issues on two of the most common Oracle operating systems: UNIX and Windows NT.

**See Also:** *Oracle9i Real Application Clusters Installation and Configuration* for a general overview of raw devices as they related to Oracle Real Application Clusters

## Backing Up to Raw Devices on UNIX

When backing up to or from raw devices, the UNIX dd command is the most common backup utility. See your operating system specific documentation for complete details about this utility.

The most important aspect of using dd is determining which options to specify. You need to know the following information:

| | |
|---|---|
| Block size | You can specify the size of the buffer that dd uses to copy data. For example, you can specify that dd should copy data in units of 8 KB or 64 KB. Note that the block size for dd need not correspond to either the Oracle block size or the operating system block size: it is merely the size of the buffer used by dd when making the copy. |
| Raw offset | On some systems, the beginning of the file on the raw device is reserved for use by the operating system. This storage space is called the **raw offset**. Oracle should not back up or restore these bytes. |
| Size of Oracle block `0` | At the beginning of every Oracle file, the system-specific code places an Oracle block called **block 0**. The generic Oracle code does not recognize this block, but the block is included in the size of the file on the operating system. Typically, this block is the same size as the other Oracle blocks in the file. |

The preceding information enables you to set the dd options specified in Table 2–1.

***Table 2–1 Options for dd Command***

| This option ... | Specifies ... |
|---|---|
| `if` | The name of the input file, that is, the file that you are reading. |
| `of` | The name of the output file, that is, the file to which you are writing. |
| `bs` | The buffer size used by dd to copy data. |
| `skip` | The number of dd buffers to skip on the input raw device if a raw offset exists. For example, if you are backing up a file on a raw device with a 64 KB raw offset, and the dd buffer size is 8 KB, then you can specify `skip=8` so that the copy starts at offset 64 KB. |
| `seek` | The number of dd buffers to skip on the output raw device if a raw offset exists. For example, if you are backing up a file onto a raw device with a 64 KB raw offset, and the dd buffer size is 8 KB, then you can specify `skip=8` so that the copy starts at offset 64 KB. |

*Table 2–1 Options for dd Command*

| This option ... | Specifies ... |
|---|---|
| `count` | The number of blocks on the input raw device for `dd` to copy. It is best to specify the exact number of blocks to copy when copying from raw device to file system, otherwise any extra space at the end of the raw volume that is not used by the oracle datafile is copied to the file system.<br>Remember to include block `0` in the total size of the input file. For example, if the `dd` block size is 8 KB, and you are backing up a 30720 KB datafile, then you can set `count=3841`. This value for `count` actually backs up 30728 bytes: the extra 8 bytes are for Oracle block `0`. |

Because a raw device can be the input or output device for a backup, you have four possible scenarios for the backup. The possible options for `dd` depend on which scenario you choose, as illustrated in Table 2–2.

*Table 2–2 Scenarios Involving dd Backups*

| Backing Up from ... | Backing Up to ... | Options Specified for dd Command |
|---|---|---|
| Raw device | Raw device | `if`, `of`, `bs`, `skip`, `seek`, `count` |
| Raw device | File system | `if`, `of`, `bs`, `skip`, `count` |
| File system | Raw device | `if`, `of`, `bs`, `seek` |
| File system | File system | `if`, `of`, `bs` |

## Backing Up with the dd utility on UNIX: Examples

For these examples of `dd` utility usage, assume the following:

- You are backing up a 30720 KB datafile.
- The beginning of the datafile has a block 0 of 8 KB.
- The raw offset is 64 KB.
- You set the `dd` block size to 8 KB when a raw device is involved in the copy.

In this example, you back up from one raw device to another raw device:

```
% dd if=/dev/rsd1b of=/dev/rsd2b bs=8k skip=8 seek=8 count=3841
```

In this example, you back up from a raw device to a file system:

```
% dd if=/dev/rsd1b of=/backup/df1.dbf bs=8k skip=8 count=3841
```

In this example, you back up from a file system to a raw device:

```
% dd if=/backup/df1.dbf of=/dev/rsd2b bs=8k seek=8
```

In this example, you back up from a file system to a file system, and so can set the block size to a high value to boost I/O performance:

```
% dd if=/oracle/dbs/df1.dbf of=/backup/df1.dbf bs=1024k
```

## Backing Up to Raw Devices on NT

Like UNIX, Windows NT supports raw disk partitions in which Oracle can store datafiles, online logs, and control files. Each raw partition is assigned either a drive letter or physical drive number and does not contain a file system. As in UNIX, each raw partition on NT is mapped to a single file.

NT differs from UNIX in the naming convention for Oracle files. On NT, raw datafile names are formatted as follows:

```
\\.\drive_letter:
\\.\PHYSICALDRIVEdrive_number
```

For example, the following are possible raw filenames:

```
\\.\G:
\\.\PHYSICALDRIVE3
```

Note that you can also create aliases to raw filenames. The standard Oracle installation provides a `SETLINKS` utility that can create aliases such as `\\.\Datafile12` that point to filenames such as `\\.\PHYSICALDRIVE3`.

The procedure for making user-managed backups of raw datafiles is basically the same as for copying files on an NT file system, except that you should use the Oracle `OCOPY` utility rather than the NT-supplied `copy.exe` or `ntbackup.exe` utilities. Alternatively, if you have MKS utilities, then you can use the `dd` utility. `OCOPY` supports 64-bit file I/O, physical raw drives, and raw files. Note that `OCOPY` cannot back up directly to tape.

To display online documentation for `OCOPY`, enter `OCOPY` by itself at the NT prompt. Sample output follows:

```
Usage of OCOPY:
     ocopy from_file [to_file [a | size_1 [size_n]]]
     ocopy -b from_file to_drive
     ocopy -r from_drive to_dir
```

Note the important `OCOPY` options described in the following table.

| This option ... | Specifies ... |
|---|---|
| b | Splits the input file into multiple output files. This option is useful for backing up to devices that are smaller than the input file. |
| r | Combines multiple input files and writes to a single output file. This option is useful for restoring backups created with the -b option. |

### Backing Up with OCOPY: Example

In this example, assume the following:

- Datafile 12 is mounted on the \\.\G: raw partition.
- The C: drive mounts a file system.
- The database is open.

To back up the datafile on the raw partition \\.\G: to a local file system, you can execute the following command at the NT prompt after placing datafile 12 in backup mode:

```
OCOPY "\\.G:" C:\backup\datafile12.bak
```

### Specifying the -b and -r Options for OCOPY: Example

In this example, assume the following:

- \\.\G: is a raw partition containing datafile 7
- The A: drive is a removable disk drive.
- The database is open.

To back up the datafile onto drive A:, you can execute the following command at the NT prompt after placing datafile 7 in backup mode:

```
# first argument is filename, second argument is drive
OCOPY -b "\\.\G:" A:\
```

When drive A: fills up, you can use another disk. In this way, you can divide the backup of datafile 1 into multiple files.

Similarly, to restore the backup, take the tablespace containing datafile 7 offline and run this command:

```
# first argument is drive, second argument is directory
OCOPY -r A:\ "\\.\G:"
```

# Verifying User-Managed Backups

You should periodically verify your backups to ensure that they are usable for recovery. This section contains the following topics:

- Testing the Restore of Backups
- Using the DBVERIFY Utility

## Testing the Restore of Backups

The best way to test the usability of backups is to restore them to a separate host and attempt to open the database, performing media recovery if necessary. This option requires that you have a separate host available for the restore procedure.

**See Also:**

- "Restoring Datafiles" on page 3-6 to learn how to restore datafiles
- "Restoring and Re-Creating Control Files" on page 3-8 to learn how to restore datafiles
- "Restoring Archived Redo Logs" on page 3-15 to learn how to restore datafiles
- "Performing Complete User-Managed Media Recovery" on page 4-9 to learn how to recover files.

## Using the DBVERIFY Utility

The DBVERIFY program is an external command-line utility that performs a physical data structure integrity check on an offline datafile. Use DBVERIFY primarily when you need to ensure that a user-managed backup of a datafile is valid before it is restored or as a diagnostic aid when you have encountered data corruption problems.

The name and location of DBVERIFY is dependent on your operating system. For example, to perform an integrity check on datafile `tbs_52.f` on UNIX, you can run the `dbv` command as follows:

```
% dbv file=tbs_52.f
```

Sample `dbv` output follows:

```
DBVERIFY: Release 9.0.1.0.0
```

```
(c) Copyright 2000 Oracle Corporation.  All rights reserved.

DBVERIFY - Verification starting : FILE = tbs_52.f

DBVERIFY - Verification complete

Total Pages Examined         : 250
Total Pages Processed (Data) : 4
Total Pages Failing   (Data) : 0
Total Pages Processed (Index): 15
Total Pages Failing   (Index): 0
Total Pages Processed (Other): 29
Total Pages Empty            : 202
Total Pages Marked Corrupt   : 0
Total Pages Influx           : 0
```

**See Also:** *Oracle9i Database Utilities* for information about DBVERIFY

# Making Logical Backups with 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 binary 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 use them to supplement physical database backups.

This section describes the Import and Export utilities, and includes the following topics:

- Using Export
- Using Import

**See Also:** *Oracle9i Database Utilities* for complete documentation describing the Export and Import utilities

## Using Export

The Export utility can back up logical database objects while the database 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.

**Caution:** If you use Export to perform a logical backup, then you must export all data 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 `ALTER SYSTEM QUIESCE RESTRICTED` mode while you export the data, so no regular users can access the data. Alternatively, you can quiesce the database before you export the data, and unquiesce the database afterward.

Table 2–3 lists available export modes.

***Table 2–3 Export Modes***

| Mode | Description |
|---|---|
| User (Owner) | Exports all objects owned by a user. |
| Tablespace | Exports all objects contained in the tablespace. |
| Table | Exports all or specific tables owned by a user and objects defined on these tables such as privileges, triggers, views, and indexes. |
| Full Database | Exports all objects of the database. |

## Using Import

The Import utility can 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:**

1. Re-create the database structure, including all tablespaces and users. These re-created structures should not have objects in them.
2. 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.

# Making User-Managed Backups of Miscellaneous Oracle Files

You should always back up initialization parameter files, networking and configuration files, and password files. If a media failure destroys these files, then you may have difficulty re-creating your original environment. For example, if you

back up the database and server parameter file but do not back up the networking files (for example, `tnsnames.ora` and `listener.ora`), then you can restore and recover the database but will not be able to authenticate users through Oracle Net until you re-create the networking files.

As a general rule, you should back up miscellaneous Oracle files after changing them. For example, if you add or change the net service names that can be used to access the database, then create a new backup of the `tnsnames.ora` file.

The easiest way to find configuration files is to start in the Oracle home directory and do a recursive search for all files ending in the `.ora` extension. For example, on UNIX you can run this command:

```
% find $ORACLE_HOME -name "*.ora" -print
```

You must use third-party utilities to back up the configuration files. For example, you can use the UNIX cp command to back up the `tnsnames.ora` and `listener.ora` files as follows:

```
% cp $ORACLE_HOME/network/admin/tnsnames.ora /disk2/bkups/tnsnames01-22-01.ora
% cp $ORACLE_HOME/network/adminlistener.ora /disk2/bkups/listener01-22-01.ora
```

You can also use an operating system utility to back up the server parameter file. Although the database does not depend on the existence of a particular version of the server parameter file to be started, you should keep relatively current backups of this file so that you do not lose changes made to the file. Note that if you lose the server parameter file, you can always create a new one or start the instance with a client-side initialization parameter file (`PFILE`).

> **See Also:** *Oracle9i Database Administrator's Guide* to learn how to manage and export server parameter files

# 3

# Performing User-Managed Restore Operations

This chapter describes how to recover a database, and includes the following topics:

- About User-Managed Restore Operations
- Keeping Records For Use in a Restore Scenario
- Determining Which Datafiles Require Recovery
- Restoring Datafiles
- Re-Creating Datafiles When Backups Are Unavailable
- Restoring and Re-Creating Control Files
- Restoring Archived Redo Logs

# About User-Managed Restore Operations

To restore a file is to replace it with a backup file. Typically, you restore a file when a media failure or user error has damaged or deleted the original file. The following files are candidates for restore operations:

- Datafiles
- Control files
- Archived redo logs
- Server parameter file

In each case, the loss of a primary file and the restore of a backup has the following implications for media recovery.

| If you lose . . . | Then . . . |
|---|---|
| One or more datafiles | You must restore them from a backup and perform media recovery. Recovery is required whenever the checkpoint SCN in the datafile header does not match the checkpoint SCN for the datafile that is recorded in the control file. |
| All copies of the current control file | You must restore a backup control file and then open the database with the `RESETLOGS` option.<br>If you do not have a backup, then you can attempt to re-create the control file. If possible, use the script included in the `ALTER DATABASE BACKUP CONTROLFILE TO TRACE` output. Additional work may be required to match the control file structure with the current database structure. |
| One copy of a multiplexed control file | Copy one of the intact multiplexed control files into the location of the damaged or missing control file and open the database. If you cannot copy the control file to its original location (for example, because the disk drive cannot be salvaged), then edit the initialization parameter file to reflect a new location. Then, open the database. |
| One or more archived logs required for media recovery | You must restore backups of these archived logs for media recovery to proceed. You can restore either to the default or to a nondefault location. If you do not have backups, then you must performing incomplete recovery up to a point before the first missing log and open `RESETLOGS`. |
| The server parameter file | If you have a backup of the server parameter file, then restore it. Alternatively, if you have a backup of the client-side initialization parameter file, then you can restore a backup of this file, start the instance, and then re-create the server parameter file. |

**Note:** Restore and recovery of Oracle Managed Files is no different from restore and recovery of user-named files.

# Keeping Records For Use in a Restore Scenario

One of the most important aspects of user-managed backup and recovery is keeping records of all current database files as well as the backups of these files. For example, you should have records for the location of the following files:

- Datafiles
- Control files
- Online redo logs (note that online logs are never backed up)
- Archived redo logs
- Initialization parameter files
- Password files
- Networking-related files

## Recording the Locations of Datafiles, Control Files, and Online Redo Logs

The following useful SQL script displays the location of all control files, datafiles, and online redo log files for the database:

```
SELECT NAME FROM V$DATAFILE
UNION ALL
SELECT MEMBER FROM V$LOGFILE
UNION ALL
SELECT NAME FROM V$CONTROLFILE
/
```

Sample output follows:

```
NAME
--------------------------------------------------------------------------------
/oracle/dbs/tbs_01.f
/oracle/dbs/tbs_02.f
/oracle/dbs/tbs_11.f
/oracle/dbs/tbs_12.f
/oracle/dbs/t1_log1.f
/oracle/dbs/t1_log2.f
/oracle/dbs/cf1.f
/oracle/dbs/cf2.f
```

```
8 rows selected.
```

> **See Also:** *Oracle9i Database Reference* for more information on the V$ views.

## Recording the Locations of Archived Redo Logs

You can determine the location of the default archived log destinations by executing the following SQL script:

```
SELECT NAME, VALUE
FROM V$PARAMETER
WHERE NAME LIKE log_archive_dest%
AND VALUE IS NOT NULL
/

NAME                             VALUE
-------------------------------- -----------------------------------------
log_archive_dest_1               LOCATION=/oracle/work/arc_dest/arc
log_archive_dest_state_1         enable
```

Determine the format for archived logs by running SHOW as follows:

```
SHOW PARAMETER LOG_ARCHIVE_FORMAT

NAME                                 TYPE    VALUE
------------------------------------ ------- ------------------------------
log_archive_format                   string  r_%t_%s.arc
```

To see a list of all the archived logs recorded in the control file, issue this query:

```
SELECT NAME FROM V$ARCHIVED_LOG;

NAME
--------------------------------------------------------------------------------
/oracle/work/arc_dest/arcr_1_110.a
/oracle/work/arc_dest/arcr_1_111.a
/oracle/work/arc_dest/arcr_1_112.a
/oracle/work/arc_dest/arcr_1_113.a
```

## Recording the Locations of Backup Files

It is not enough to merely record the location of backup files: you must correlate the backups with the original files. If possible, name the backups with the same relative filename as the primary file. Whatever naming system you use, keep a table

containing the relevant information. For example, you could keep the following table as a record of database file locations in case of a restore emergency.

| Datafile Number | Tablespace | Backup Filename |
|---|---|---|
| 0 (control file) | 0 (control file) | /dsk3/backup/cf.f |
| 1 | SYSTEM | /dsk3/backup/tbs_01.f |
| 2 | undo | /dsk3/backup/tbs_02.f |
| 3 | temp | /dsk3/backup/tbs_11.f |
| 4 | users | /dsk3/backup/tbs_12.f |

## Determining Which Datafiles Require Recovery

You can use the dynamic performance view V$RECOVER_FILE to determine which files to restore in preparation for media recovery. This view lists all files that need to be recovered, and explains why they need to be recovered.

The following query displays the file ID numbers of datafiles that require media recovery as well as the reason for recovery (if known) and the SCN and time when recovery needs to begin:

```
SELECT * FROM V$RECOVER_FILE;

FILE#      ONLINE  ERROR              CHANGE#    TIME
---------- ------- ------------------ ---------- ---------
        14 ONLINE                             0
        15 ONLINE  FILE NOT FOUND             0
        21 OFFLINE OFFLINE NORMAL             0
```

**Note:** The view is not useful if the control file currently in use is a restored backup or a new control file created after the media failure occurred. A restored or re-created control file does not contain the information Oracle needs to fill V$RECOVER_FILE accurately.

Query V$DATAFILE and V$TABLESPACE to obtain filenames and tablespace names for datafiles requiring recovery. For example, enter:

```
SELECT d.NAME, t.NAME AS tablespace_name
FROM V$DATAFILE d, V$TABLESPACE t
WHERE t.TS# = d.TS#
AND d.FILE# IN (14,15,21);  # use values obtained from V$RECOVER_FILE query
```

```
NAME                               TABLESPACE_NAME
---------------------------------- ----------------
/oracle/dbs/tbs_14.f               TBS_1
/oracle/dbs/tbs_15.f               TBS_2
/oracle/dbs/tbs_21.f               TBS_3
```

You can combine these queries in the following SQL*Plus script (sample output shown below):

```
COL df# FORMAT 999
COL df_name FORMAT a20
COL tbsp_name FORMAT a10
COL status FORMAT a7
COL error FORMAT a10

SELECT r.FILE# AS df#, d.NAME AS df_name, t.NAME AS tbsp_name,
       d.STATUS, r.ERROR, r.CHANGE#, r.TIME
FROM V$RECOVER_FILE r, V$DATAFILE d, V$TABLESPACE t
WHERE t.TS# = d.TS#
AND d.FILE# = r.FILE#
/
```

Sample output follows:

```
 DF# DF_NAME              TBSP_NAME  STATUS  ERROR          CHANGE# TIME
---- -------------------- ---------- ------- ---------- ---------- ----------
  14 /oracle/dbs/tbs_14.f TBS_1      OFFLINE OFFLINE             0
                                             NORMAL

  15 /oracle/dbs/tbs_15.f TBS_2      OFFLINE OFFLINE             0
                                             NORMAL

  21 /oracle/dbs/tbs_21.f TBS_3      OFFLINE OFFLINE             0
                                             NORMAL
```

# Restoring Datafiles

If a media failure permanently damages one or more datafiles of a database, then you must restore backups of these datafiles before you can recover the damaged files. If you cannot restore a damaged datafile to its original location (for example, you must replace a disk, so you restore the files to an alternate disk), then you must indicate the new locations of these files to the control file.

If you are restoring an Oracle file on a raw disk or partition, then the procedure is basically the same as when restoring to a file on a file system. However, you must be aware of the naming conventions for files on raw devices (which differ

depending on the operating system), and use an operating system utility that supports raw devices.

> **See Also:** "Making User-Managed Backups to Raw Devices" on page 2-22 for an overview of considerations when backing up and restoring files on raw devices

**To restore backup datafiles to their default location:**

1. Determine which datafiles to recover by using the techniques described in "Determining Which Datafiles Require Recovery" on page 3-5.
2. If the database is open, then take the tablespaces containing the inaccessible datafiles offline. For example, enter:

   ```
   ALTER TABLESPACE users OFFLINE IMMEDIATE;
   ```

3. Copy backups of the damaged datafiles to their default location using operating system commands. For example, to restore `tbs_24.f` on UNIX you might issue:

   ```
   % cp /disk2/backup/tbs_24.bak /disk1/oracle/dbs/tbs_24.f
   ```

4. Recover the affected tablespace. For example, enter:

   ```
   RECOVER TABLESPACE users
   ```

5. Bring the recovered tablespace online. For example, enter:

   ```
   ALTER TABLESPACE users ONLINE;
   ```

## Re-Creating Datafiles When Backups Are Unavailable

If a datafile is damaged and no backup of the file is available, then you can still recover the datafile if:

- All archived log files written after 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)

**To re-create a datafile for recovery:**

1. Create a new, empty datafile to replace a damaged datafile that has no corresponding backup. For example, assume that the datafile

/disk1/users1.f has been damaged, and no backup is available. The following statement re-creates the original datafile (same size) on disk2:

```
ALTER DATABASE CREATE DATAFILE '/disk1/users1.f' AS '/disk2/users1.f';
```

This statement creates an empty file that is the same size as the lost file. Oracle looks at information in the control file and the data dictionary to obtain size information. The old datafile is renamed as the new datafile.

2. Perform media recovery on the empty datafile. For example, enter:

```
RECOVER DATAFILE '/disk2/users1.f'
```

3. All archived redo logs written after the original datafile was created must be mounted and reapplied to the new, empty version of the lost datafile during recovery.

---

**Note:** You cannot re-create any of the datafiles for the SYSTEM tablespace by using the CREATE DATAFILE clause of the ALTER DATABASE statement because the necessary redo data is not available.

---

## Restoring and Re-Creating Control Files

If a media failure has affected the control files of a database (whether control files are multiplexed or not), then 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, avoid the automatic shutdown of the database by immediately correcting the media failure. If the database shuts down before you correct the temporary media failure, however, then 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 multiplexed the control files. The following sections describe the appropriate procedures:

- Losing a Member of a Multiplexed Control File
- Losing All Members of a Multiplexed Control File When a Backup Is Available
- Losing All Current and Backup Control Files

## Losing a Member of a Multiplexed Control File

Use the following procedures to recover a database if a permanent media failure has damaged one or more control files of a database and at least one control file has *not* been damaged by the media failure.

### Copying a Multiplexed Control File to a Default Location

Assuming that the disk and file system containing the lost control file are intact, then you can simply copy one of the intact control files to the location of the missing control file. In this case, you do not have to alter the `CONTROL_FILES` initialization parameter setting.

**To replace a damaged control file by copying a multiplexed control file:**

1. If the instance is still running, then abort it:

   ```
   SHUTDOWN ABORT
   ```

2. Correct the hardware problem that caused the media failure. If you cannot repair the hardware problem quickly, then you can proceed with database recovery by restoring damaged control files to an alternative storage device, as described in "Copying a Multiplexed Control File to a Nondefault Location" on page 3-9.

3. Use an intact multiplexed copy of the database's current control file to copy over the damaged control files. For example, to replace `bad_cf.f` with `good_cf.f`, you might enter:

   ```
   % cp /oracle/good_cf.f /oracle/dbs/bad_cf.f
   ```

4. Start a new instance and mount and open the database. For example, enter:

   ```
   STARTUP
   ```

### Copying a Multiplexed Control File to a Nondefault Location

Assuming that the disk and file system containing the lost control file are not intact, then you cannot copy one of the "good" control files to the location of the missing control file. In this case, you must alter the `CONTROL_FILES` initialization parameter to indicate a new location for the missing control file.

**To restore a control file to a nondefault location:**

1. If the instance is still running, then abort it:

   ```
   SHUTDOWN ABORT
   ```

2. If you cannot correct the hardware problem that caused the media failure, then copy the intact control file to alternative locations. For example, to copy `good_cf.f` to `new_cf.f` you might issue:

   ```
   % cp /oracle/dbs/good_cf.f /oracle/dbs/new_cf.f
   ```

3. Edit the parameter file of the database so that the `CONTROL_FILES` parameter reflects the current locations of all control files and excludes all control files that were not restored. For example, assume the initialization parameter file contains:

   ```
   CONTROL_FILES = '/oracle/dbs/good_cf.f', '/oracle/dbs/bad_cf.f'
   ```

   Then, you can edit it as follows:

   ```
   CONTROL_FILES = '/oracle/dbs/good_cf.f', '/oracle/dbs/new_cf.f'
   ```

4. Start a new instance and mount and open the database. For example, enter the following in SQL*Plus:

   ```
   STARTUP
   ```

## Losing All Members of a Multiplexed Control File When a Backup Is Available

Use the following procedures to restore a backup control file if a permanent media failure has damaged all control files of a database and you have a backup of the control file. When a control file is inaccessible, then you can start the instance, but not mount the database. If you attempt to mount the database when the control file is unavailable, you see this error message:

```
ORA-00205: error in identifying controlfile, check alert log for more info
```

You cannot mount and open the database until you make the control file accessible again. If you restore a backup control file, then you must open the database with the `RESETLOGS` option.

As indicated in Table 3–1, the procedure for restoring the control file depends on whether the online redo logs are available.

*Table 3–1 Scenarios When Control Files Are Lost*

| Status of Online Logs | Status of Datafiles | Response |
|---|---|---|
| Available | Current | If the online logs contain redo necessary for recovery, then restore a backup control file apply the logs during recovery. Hence, you must specify the filename of the online logs containing the changes in order to open the database. After recovery, open RESETLOGS. |
| Unavailable | Current | If the online logs contain redo necessary for recovery, then you must re-create the control file. Because the logs are inaccessible, open RESETLOGS. |
| Available | Backup | Restore a backup control file, perform complete recovery, and then open RESETLOGS. |
| Unavailable | Backup | Restore a backup control file, perform incomplete recovery, and then open RESETLOGS. |

## Restoring a Backup Control File to the Default Location

If possible, restore the control file to its original location. In this way, you avoid having to specify new control file locations in the initialization parameter file.

**To restore a backup control file to its default location:**

1. If the instance is still running, abort it:

   ```
   SHUTDOWN ABORT
   ```

2. Correct the hardware problem that caused the media failure.

3. Restore the backup control file to all locations specified in the CONTROL_FILES initialization parameter. For example, if /dsk1/oracle/dbs/cf1.f and /dsk2/cf2.f are the control file locations listed in the server parameter file, then use an operating system utility to restore the backup control file to these locations:

   ```
   % cp /backup/cf.bak /dsk1/oracle/dbs/cf1.f
   % cp /backup/cf.bak /dsk2/cf2.f
   ```

4. Start a new instance and mount the database. For example, enter:

   ```
   STARTUP MOUNT
   ```

5. Begin recovery by executing the RECOVER command with the USING BACKUP CONTROLFILE clause. Specify UNTIL CANCEL if you are performing incomplete recovery. For example, enter:

```
RECOVER DATABASE USING BACKUP CONTROLFILE UNTIL CANCEL
```

6. Apply the prompted archived logs. If you then receive another message saying that the required archived log is missing, it probably means that a necessary redo record is located in the online redo logs. This situation can occur when unarchived changes were located in the online logs when the instance crashed.

   For example, assume that you see the following:

```
ORA-00279: change 55636 generated at 06/08/2000 16:59:47 needed for thread 1
ORA-00289: suggestion : /oracle/work/arc_dest/arcr_1_111.arc
ORA-00280: change 55636 for thread 1 is in sequence #111
Specify log: {<RET>=suggested | filename | AUTO | CANCEL}
```

   You can specify the name of an online redo log and press Enter (you may have to try this a few times until you find the correct log):

```
/oracle/dbs/t1_log1.f
Log applied.
Media recovery complete.
```

   If for some reason the online logs are not accessible, then you can cancel recovery without applying the online logs. Note that if all datafiles are current, and redo is located in the online logs that is required for recovery, then you cannot open the database without applying the online logs. If the online logs are inaccessible, then you must re-create the control file (refer to "Losing All Current and Backup Control Files" on page 3-13).

7. Open the database with the RESETLOGS option after finishing recovery:

```
ALTER DATABASE OPEN RESETLOGS;
```

8. Immediately back up the database as a precautionary measure, as described in "Making User-Managed Backups of the Whole Database" on page 2-4.

### Restoring a Backup Control File to a Nondefault Location

If you cannot restore the control file to its original place because the media damage is too severe, then you must specify new control file locations in the server parameter file. A valid control file must be available in all locations specified by the CONTROL_FILES initialization parameter. If not, then Oracle prevents you from the mounting the database.

**To restore a control file to a nondefault location:**

Follow the steps in "Restoring a Backup Control File to the Default Location" on page 3-11, except after step 2 add the following step:

Edit all locations specified in the CONTROL_FILES initialization parameter to reflect the new control file locations. For example, if the control file locations listed in the server parameter file are as follows:

```
CONTROL_FILES = '/dsk1/oracle/dbs/cf1.f', '/dsk2/cf2.f'
```

You can change the initialization parameter to read:

```
CONTROL_FILES = '/dsk3/tmp/cf1.f', 'dsk3/tmp/cf2.f'
```

## Losing All Current and Backup Control Files

If all control files have been lost or damaged by a permanent media failure, but all online redo logfiles remain intact, then you can recover the database after creating a new control file. Note that this procedure does *not* require you to open the database with the RESETLOGS option.

Depending on the existence and currency of a control file backup, you have the options listed in Table 3–2 for generating the text of the CREATE CONTROLFILE statement. Note that changes to the database are recorded in the alert_*SID*.log, so check this log when deciding which option to choose.

***Table 3–2 Options for Creating the Control File*** **(Page 1 of 2)**

| If you . . . | Then . . . |
|---|---|
| Executed ALTER DATABASE BACKUP CONTROLFILE TO TRACE NORESETLOGS after you made the last structural change to the database, and if you have saved the SQL command trace output | Use the CREATE CONTROLFILE statement from the trace output as-is. |
| Performed your most recent execution of ALTER DATABASE BACKUP CONTROLFILE TO TRACE before you made a structural change to the database | Edit the output of ALTER DATABASE BACKUP CONTROLFILE TO TRACE to reflect the change. For example, if you recently added a datafile to the database, then add this datafile to the DATAFILE clause of the CREATE CONTROLFILE statement. |

***Table 3–2 Options for Creating the Control File*** **(Page 2 of 2)**

| If you . . . | Then . . . |
|---|---|
| Backed up the control file with the ALTER DATABASE BACKUP CONTROLFILE TO *filename* statement (not the TO TRACE option) | Use the control file copy to obtain SQL output. Copy the backup control file and execute STARTUP MOUNT before ALTER DATABASE BACKUP CONTROLFILE TO TRACE NORESETLOGS. If the control file copy predated a recent structural change, then edit the trace output to reflect the structural change. |
| Do not have a control file backup in either TO TRACE format or TO *filename* format | Create the CREATE CONTROLFILE statement manually (see *Oracle9i SQL Reference*). |

**Note:** If your character set is not the default US7ASCII, then you must specify the character set as an argument to the CREATE CONTROLFILE statement.

**To create a new control file:**

1. Start the database in NOMOUNT mode. For example, enter:

```
STARTUP NOMOUNT
```

2. Create the control file with the CREATE CONTROLFILE statement, specifying the NORESETLOGS option (refer to Table 3–2 for options). For example, enter:

```
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,
DATAFILE
    '/diska/prod/sales/db/database1.dbf',
```

```
'/diskb/prod/sales/db/filea.dbf';
```

After creating the control file, Oracle mounts the database.

3. Recover the database as normal (*without* specifying the `USING BACKUP CONTROLFILE` clause):

```
RECOVER DATABASE
```

4. Open the database after media recovery completes:

```
ALTER DATABASE OPEN
```

Note that a `RESETLOGS` is not necessary.

5. Immediately back up the control file. The following SQL statement backs up a database's control file to `/oracle/backup/cf.bak`:

```
ALTER DATABASE BACKUP CONTROLFILE TO '/oracle/backup/cf.bak' REUSE;
```

**See Also:** "Backing Up the Control File to a Trace File" on page 2-19.

## Restoring Archived Redo Logs

All archived redo log files generated between the time a restored backup was created and the target recovery time are required for the pending media recovery. The archived logs will eventually need to be on disk so that they are available to Oracle during the recovery.

**To restore necessary archived redo logs:**

1. To determine which archived redo log files are needed, query `V$ARCHIVED_LOG` and `V$RECOVERY_LOG`. If a datafile requires recovery, but not backup of the datafile exists, then you need all redo generated starting from the time when the datafile was added to the database.

| View | Description |
|---|---|
| `V$ARCHIVED_LOG` | Lists all the filenames for all the archived logs. |
| `V$RECOVERY_LOG` | Lists only the archived redo logs that Oracle needs to perform media recovery. It also includes the probable names of the files, using `LOG_ARCHIVE_FORMAT`.<br>**Note:** This view is only populated when recovery is required for a datafile. Hence, this view is not useful in the case of a planned recovery such as a user error. |

2. If space is available, then restore the required archived redo log files to the location specified by LOG_ARCHIVE_DEST_1. Oracle locates the correct log automatically when required during media recovery. For example, enter:

```
% cp /disk2/arc_backup/*.arc /disk1/oracle/dbs/arc_dest
```

3. If sufficient space is not available at the location indicated by the archiving destination initialization parameter, restore some or all of the required archived redo log files to an alternate location. Specify the location before or during media recovery using the LOGSOURCE parameter of the SET statement in SQL*Plus or the RECOVER . . . FROM parameter of the ALTER DATABASE statement in SQL. For example, enter:

```
SET LOGSOURCE /disk2/temp   # set location using SET statement
DATABASE RECOVER FROM '/disk2/temp';  # set location in RECOVER statement itself
```

4. After an archived log is applied, and after making sure that a copy of each archived log group still exists in offline storage, delete the restored copy of the archived redo log file to free disk space. For example, after making the log directory your working directory, enter:

```
% rm *.arc
```

**See Also:** *Oracle9i Database Reference* for more information about the data dictionary views, and "Performing User-Managed Media Recovery: Overview" on page 4-2 for an overview of log application during media recovery

# 4

# Performing User-Managed Media Recovery

This chapter describes how to recover a database, and includes the following topics:

# Performing User-Managed Media Recovery: Overview

During complete or incomplete media recovery, Oracle applies redo log files to the datafiles during the roll forward phase of media recovery. Because changes to undo segments are recorded in the online redo log, rolling forward regenerates the corresponding undo segments. Rolling forward proceeds through as many redo log files as necessary to bring the database forward in time.

If you do not use Recovery Manager (RMAN) to perform recovery, then you should use the SQL*Plus RECOVER command. It is also possible to use the SQL statement ALTER DATABASE RECOVER, but it is highly recommended that you use the SQL*Plus RECOVER command instead.

This section contains these topics:



## Preconditions of Performing User-Managed Recovery

To start any type of media recovery, you must adhere to the following restrictions:

- You must have administrator privileges.
- All recovery sessions must be compatible.
- One session cannot start complete media recovery while another performs incomplete media recovery.
- You cannot start media recovery if you are connected to the database through a shared server process.

## Applying Logs Automatically with the RECOVER Command

Oracle Corporation recommends that you use the SQL*Plus RECOVER command rather than the ALTER DATABASE RECOVER statement to perform media recovery. In almost all cases, the SQL*Plus method is easier.

When using SQL*Plus to perform media recovery, the easiest strategy is to perform automatic recovery. Automatic recovery initiates recovery without manually prompting SQL*Plus to apply each individual archived log.

When using SQL*Plus, you have two options for automating the application of the default filenames of archived redo logs needed during recovery:

- Issuing SET AUTORECOVERY ON before issuing the RECOVER command
- Specifying the AUTOMATIC keyword as an option of the RECOVER command

In either case, no interaction is required when you issue the RECOVER command if the necessary files are in the correct locations with the correct names.

The filenames used when you use automatic recovery are derived from the concatenated values of LOG_ARCHIVE_FORMAT with LOG_ARCHIVE_DEST_*n*, where *n* is the highest value among all enabled, local destinations.

For example, assume the following initialization parameter settings are in effect in the database instance:

```
LOG_ARCHIVE_DEST_1 = "LOCATION=/arc_dest/loc1/"
LOG_ARCHIVE_DEST_2 = "LOCATION=/arc_dest/loc2/"
LOG_ARCHIVE_DEST_STATE_1 = DEFER
LOG_ARCHIVE_DEST_STATE_2 = ENABLE
LOG_ARCHIVE_FORMAT = arch_%t_%s.arc
```

In this case, SQL*Plus automatically suggests the filename /arc_dest/loc2/arch_%t_%s.arc (where %t is the thread and %s is the sequence).

If you run SET AUTORECOVERY OFF, which is the default option, then you must enter the filenames manually, or accept the suggested default filename by pressing the Enter key.

### Using SET AUTORECOVERY for Automatic Recovery

Run the SET AUTORECOVERY ON command to enable on automatic recovery.

**To automate the recovery using SET AUTORECOVERY:**

1. Restore a backup of the offline datafiles. This example restores an inconsistent backup of all datafiles using an operating system utility:

   ```
   % cp /fs2/BACKUP/tbs* /oracle/dbs
   ```

2. Ensure the database is mounted. For example, if the database is shut down, run:

   ```
   STARTUP MOUNT
   ```

3. Enable automatic recovery. For example, in SQL*Plus run:

```
SET AUTORECOVERY ON
```

4. Recover the desired datafiles. This example recovers the whole database:

```
RECOVER DATABASE
```

Oracle automatically suggests and applies the necessary archived logs, as in this sample output:

```
ORA-00279: change 53577 generated at 01/26/00 19:20:58 needed for thread 1
ORA-00289: suggestion : /oracle/work/arc_dest/arcr_1_802.arc
ORA-00280: change 53577 for thread 1 is in sequence #802
Log applied.
ORA-00279: change 53584 generated at 01/26/00 19:24:05 needed for thread 1
ORA-00289: suggestion : /oracle/work/arc_dest/arcr_1_803.arc
ORA-00280: change 53584 for thread 1 is in sequence #803
ORA-00278: log file "/oracle/work/arc_dest/arcr_1_802.arc" no longer needed for this
recovery
Log applied.
Media recovery complete.
```

5. Open the database. For example:

```
ALTER DATABASE OPEN;
```

---

**Note:** After issuing the ALTER DATABASE RECOVER statement, 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.

---

### Using RECOVERY AUTOMATIC for Automatic Recovery

Besides using SET AUTORECOVERY to turn on automatic recovery, you can also simply specify the AUTOMATIC keyword in the RECOVER command.

**To automate the recovery with the RECOVER AUTOMATIC command:**

1. Restore a backup of the offline datafiles. This example restores a backup of all datafiles:

```
% cp /oracle/work/BACKUP/tbs* /oracle/dbs
```

2. Ensure the database is mounted. For example, if the database is shut down, run:

```
STARTUP MOUNT
```

3. Recover the desired datafiles by specifying the AUTOMATIC keyword. This example performs automatic recovery on the whole database:

```
RECOVER AUTOMATIC DATABASE
```

4. Oracle automatically suggests and applies the necessary archived logs as illustrated in the following output:

```
ORA-00279: change 53577 generated at 01/26/00 19:20:58 needed for thread 1
ORA-00289: suggestion : /oracle/work/arc_dest/arcr_1_802.arc
ORA-00280: change 53577 for thread 1 is in sequence #802
Log applied.
ORA-00279: change 53584 generated at 01/26/00 19:24:05 needed for thread 1
ORA-00289: suggestion : /oracle/work/arc_dest/arcr_1_803.arc
ORA-00280: change 53584 for thread 1 is in sequence #803
ORA-00278: log file "/oracle/work/arc_dest/arcr_1_802.arc" no longer needed for this
recovery
Log applied.
Media recovery complete.
```

5. Open the database. For example:

```
ALTER DATABASE OPEN;
```

If you use an Oracle Real Application Clusters configuration, and if you are performing incomplete recovery or using a backup control file, then Oracle can only compute the name of the first archived redo log file from the *first* redo thread. You may have to manually apply the first log file from the other redo threads. After the first log file in a given thread has been supplied, Oracle can suggest the names of the subsequent logs in this thread.

> **See Also:** Your operating system specific Oracle documentation for examples of log file application

## Recovering When Archived Logs Are in the Default Location

Recovering when the archived logs are in their default location is the simplest case. As a log is needed, Oracle suggests the filename. If you are running nonautomatic media recovery with SQL*Plus, then the output is displayed in this format:

```
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 ]
```

For example, SQL*Plus displays output similar to the following:

```
ORA-00279: change 53577 generated at 01/26/00 19:20:58 needed for thread 1
ORA-00289: suggestion : /oracle/arc_dest/arcr_1_802.arc
ORA-00280: change 53577 for thread 1 is in sequence #802

Specify log: [<RET> for suggested | AUTO | FROM logsource | CANCEL ]
```

Similar messages are returned when you use an ALTER DATABASE . . . RECOVER statement. However, no prompt is displayed.

Oracle suggests archived redo log filenames by concatenating the current values of the initialization parameters LOG_ARCHIVE_DEST_*n* (where *n* is the highest value among all enabled, local destinations) and LOG_ARCHIVE_FORMAT and using log history information from the control file. For example, the following are possible settings for archived redo logs:

```
LOG_ARCHIVE_DEST_1 = 'LOCATION = /oracle/arc_dest/'
LOG_ARCHIVE_FORMAT = arcr_%t_%s.arc

SELECT NAME FROM V$ARCHIVED_LOG;

NAME
------------------------------
/oracle/arc_dest/arcr_1_467.arc
/oracle/arc_dest/arcr_1_468.arc
/oracle/arc_dest/arcr_1_469.arc
```

Thus, if all the required archived log files are mounted at the LOG_ARCHIVE_DEST_1 destination, and if the value for LOG_ARCHIVE_FORMAT is never altered, then Oracle can suggest and apply log files to complete media recovery automatically.

## Recovering When Archived Logs Are in a Nondefault Location

Performing media recovery when archived logs are not in their default location adds an extra step into the recovery procedure. You have the following mutually exclusive options:

- Edit the LOG_ARCHIVE_DEST_*n* parameter that specifies the location of the archived redo logs, then recover as usual.
- Use the SET statement in SQL*Plus to specify the nondefault log location before recovery, or the LOGFILE parameter of the RECOVER command

## Resetting the Archived Log Destination

You can edit the initialization parameter file or issue ALTER SYSTEM statements to change the default location of the archived redo logs.

**To change the default archived log location before recovery:**

1. Use an operating system utility to restore the archived logs to the nondefault location. For example, enter:

   ```
   % cp /disk3/arc_bak/* /disk2/tmp
   ```

2. Change the value for the archive log parameter to the desired nondefault location. You can issue ALTER SYSTEM statements while the instance is started, or edit the initialization parameter file and then start the database instance. For example, while the instance is shut down edit the parameter file as follows:

   ```
   LOG_ARCHIVE_DEST_1 = 'LOCATION=/disk2/tmp/arc'
   LOG_ARCHIVE_FORMAT = r_%t_%s.arc
   ```

3. Using SQL*Plus, start a new instance by specifying the edited initialization parameter file, and then mount the database. For example, enter:

   ```
   STARTUP MOUNT
   ```

4. Begin media recovery as usual. For example, enter:

   ```
   RECOVER DATABASE
   ```

## Overriding the Archived Log Destination

In some cases, you may want to override the current setting for the archiving destination parameter as a source for redo 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 redo log files at the location specified by the archiving destination parameter. In this case, use one of the following procedures.

**To recover using logs in a nondefault location with SET LOGSOURCE:**

1. Using an operating system utility, move the archived redo logs to an alternative location. For example, enter:

   ```
   % cp /disk1/oracle/arc_dest/* /disk2/temp
   ```

2. Specify the alternative location within SQL*Plus for the recovery operation. Use the LOGSOURCE parameter of the SET statement or the RECOVER . . . FROM clause of the ALTER DATABASE statement. For example, start SQL*Plus and run:

```
SET LOGSOURCE "/disk2/temp"
```

3. Recover the offline tablespace:

```
RECOVER AUTOMATIC TABLESPACE offline_tbsp
```

4. Alternatively, you can avoid running SET LOGSOURCE and simply run:

```
RECOVER AUTOMATIC TABLESPACE offline_tbsp FROM "/disk2/temp"
```

**Note:** Overriding the redo log source does not affect the archive redo log destination for online redo logs groups being archived.

## Responding to Unsuccessful Application of Redo Logs

If you are using SQL*Plus's recovery options (not SQL statements), then each time Oracle successfully applies a redo log file, the following message is returned:

```
Log applied.
```

Oracle then prompts for the next log in the sequence or, if the most recently applied log is the last required log, terminates recovery.

If the suggested file is incorrect or you provide an incorrect filename, then Oracle returns an error message. For example, you may see something like:

```
ORA-00308: cannot open archived log "/oracle/work/arc_dest/arcr_1_811.arc"
ORA-27037: unable to obtain file status
SVR4 Error: 2: No such file or directory
Additional information: 3
```

Recovery cannot continue until the required redo log file is applied. If Oracle returns an error message after supplying a redo log filename, then the following responses are possible.

| Error | Possible Cause | Solution |
|---|---|---|
| ORA-27037: unable to obtain file status | Entered wrong filename.<br>Log is missing. | Reenter correct filename.<br>Restore backup archived redo log. |

| Error | Possible Cause | Solution |
|---|---|---|
| `ORA-27047: unable to read the header block of file` | The log may have been partially written or become corrupted. | If you can locate an uncorrupted or complete log copy, then apply the intact copy and continue recovery.<br>If no copy of the log exists and you know the time of the last valid redo entry, then you must use incomplete recovery. Restart recovery from the beginning, including restoring backups. |

# Performing Complete User-Managed Media Recovery

When you perform complete recovery, you recover the backups to the current SCN. You can either recover the whole database at once or recover individual tablespaces or datafiles. Because you do not have to open the database with the `RESETLOGS` option after complete recovery as you do after incomplete recovery, you have the option of recovering some datafiles at one time and the remaining datafiles later.

This section describes the steps necessary to complete media recovery operations, and includes the following topics:

- Performing Closed Database Recovery
- Performing Datafile Recovery in an Open Database

> **See Also:** *Oracle9i Backup and Recovery Concepts* to familiarize yourself with fundamental recovery concepts and strategies

## Performing Closed Database Recovery

This section describes steps to perform complete recovery while the database is not open. You can recover either all damaged datafiles in one operation, or perform individual recovery of each damaged datafile in separate operations.

Perform the media recovery in the following stages:

1. Prepare for closed database recovery as described in "Preparing for Closed Database Recovery" on page 4-10.
2. Restore the necessary files as described in "Restoring Backups of the Damaged or Missing Files" on page 4-10.
3. Recover the restored datafiles as described in "Recovering the Database" on page 4-11.

### Preparing for Closed Database Recovery

In this stage, you shut down the instance and inspect the media device that is causing the problem.

**To prepare for closed database recovery:**

1. If the database is open, then shut it down by using the ABORT option:

   ```
   SHUTDOWN ABORT
   ```

2. If you are recovering from a media error, then correct it if possible. If the hardware problem that caused the media failure was temporary, and if the data was undamaged (for example, a disk or controller power failure), then no media recovery is required: simply start the database and resume normal operations. If you cannot repair the problem, then proceed to the next step.

### Restoring Backups of the Damaged or Missing Files

In this stage, you restore all necessary backups.

**To restore the necessary files:**

1. Determine which datafiles to recover by using the techniques described in "Determining Which Datafiles Require Recovery" on page 3-5.

2. If the files are permanently damaged, then identify the most recent backups for the damaged files. Restore *only* the datafiles damaged by the media failure: do not restore any undamaged datafiles or any online redo log files.

   For example, if `/oracle/dbs/tbs_10.f` is the only damaged file, then you may consult your records and determine that `/oracle/backup/tbs_10.backup` is the most recent backup of this file. If you do not have a backup of a specific datafile, then you may be able to create an empty replacement file that can be recovered.

3. Use an operating system utility to restore the files to their default location or to a new location. Restore the necessary files as described in Chapter 3, "Performing User-Managed Restore Operations". For example, a UNIX user restoring `/oracle/dbs/tbs_10.f` to its default location might enter:

   ```
   % cp /oracle/backup/tbs_10.backup /oracle/dbs/tbs_10.f
   ```

Follow these guidelines when determining where to restore datafile backups:

| If . . . | Then . . . |
|---|---|
| The hardware problem is repaired and you can restore the datafiles to their default locations | Restore the datafiles to their default locations and begin media recovery. |
| The hardware problem persists and you cannot restore datafiles to their original locations | Restore the datafiles to an alternative storage device. Indicate the new location of these files in the control file. Use the operation described in "Renaming and Relocating Datafiles" in the *Oracle9i Database Administrator's Guide*, as necessary. |

### Recovering the Database

In the final stage, you recover the datafiles that you have restored.

**To recover the restored datafiles:**

1. Connect to the database with administrator privileges, then start a new instance and mount, but do not open, the database. For example, enter:

   ```
   STARTUP MOUNT
   ```

2. Obtain the datafile names and statuses of all datafiles by checking the list of datafiles that normally accompanies the current control file or querying the V$DATAFILE view. For example, enter:

   ```
   SELECT NAME,STATUS FROM V$DATAFILE;
   ```

3. Ensure that all datafiles of the database are online. All datafiles of the database requiring recovery must be online unless an offline tablespace was taken offline normally or is part of a read-only tablespace. For example, to guarantee that a datafile named /oracle/dbs/tbs_10.f is online, enter the following:

   ```
   ALTER DATABASE DATAFILE '/oracle/dbs/tbs_10.f' ONLINE;
   ```

   If a specified datafile is already online, then Oracle ignores the statement. If you prefer, create a script to bring all datafiles online at once as in the following:

   ```
   SPOOL onlineall.sql
   SELECT 'ALTER DATABASE DATAFILE '''||name||''' ONLINE;' FROM V$DATAFILE;
   SPOOL OFF

   SQL> @onlineall
   ```

4. Issue the statement to recover the database, tablespace, or datafile. For example, enter one of the following RECOVER command:

```
RECOVER DATABASE   # recovers whole database
RECOVER TABLESPACE users   # recovers specific tablespace
RECOVER DATAFILE '/oracle/dbs/tbs_10';   # recovers specific datafile
```

Follow these guidelines when deciding which statement to execute:

| To . . . | Then . . . |
|---|---|
| Recover all damaged files in one step | Execute RECOVER DATABASE |
| Recover an individual tablespace | Execute RECOVER TABLESPACE |
| Recover an individual damaged datafile | Execute RECOVER DATAFILE |
| Parallelize recovery of the whole database or an individual datafile | See "Performing Media Recovery in Parallel" on page 4-25 |

5. If you choose not to automate the application of archived logs, then you must accept or reject each required redo log that Oracle prompts you for. If you automate recovery, then Oracle applies the necessary logs automatically. Oracle continues until all required archived and online redo log files have been applied to the restored datafiles.
6. Oracle notifies you when media recovery is complete:

```
Media recovery complete.
```

If no archived redo log files are required for complete media recovery, then Oracle applies all necessary online redo log files and terminates recovery.

7. After recovery terminates, then open the database for use:

```
ALTER DATABASE OPEN;
```

> **See Also:** "Performing User-Managed Media Recovery: Overview" on page 4-2 for more information about applying redo log files

## Performing Datafile Recovery in an Open Database

It is possible for a media failure to occur while the database remains open, leaving the undamaged datafiles online and available for use. Oracle automatically takes the damaged datafiles offline—but not the tablespaces that contain them—if the

database writer is unable to write to them. Queries that cannot read damaged files return errors, but Oracle does not take the files offline because of the failed queries. For example, you may run a query and see output such as:

```
ERROR at line 1:
ORA-01116: error in opening database file 11
ORA-01110: data file 11: '/oracle/dbs/tbs_32.f'
ORA-27041: unable to open file
SVR4 Error: 2: No such file or directory
Additional information: 3
```

The media recovery procedure in this section 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, then Oracle automatically shuts down the database.

Perform media recovery in these stages:

1. Prepare the database for recovery by making sure it is open and taking the tablespaces requiring recovery offline, as described in "Preparing for Open Database Recovery" on page 4-13.
2. Restore the necessary files in the affected tablespaces as described in "Restoring Backups of the Damaged or Missing Files" on page 4-14.
3. Recover the affected tablespaces as described in "Recovering Offline Tablespaces in an Open Database" on page 4-14.

**See Also:**

- "Determining Which Datafiles Require Recovery" on page 3-5 for more information about determining when recovery is necessary
- "Performing Closed Database Recovery" on page 4-9 for procedures for proceeding with complete media recovery of SYSTEM tablespaces datafiles

### Preparing for Open Database Recovery

In this stage, you take affected tablespaces offline and inspect the media device that is causing the problem.

**To prepare for datafile recovery when the database is open:**

1. If the database is open when you discover that recovery is required, take all tablespaces containing damaged datafiles offline. For example, if tablespace `users` contains damaged datafiles, enter:

   ```
   ALTER TABLESPACE users OFFLINE TEMPORARY;
   ```

2. Correct the hardware problem that caused the media failure. If the hardware problem cannot be repaired quickly, proceed with database recovery by restoring damaged files to an alternative storage device.

### Restoring Backups of the Damaged or Missing Files

In this stage, you restore all necessary backups in the offline tablespaces.

**To restore datafiles in an open database:**

1. If files are permanently damaged, then 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, then do so. If the hardware problem persists, then restore the datafiles to an alternative storage device.

   **Note:** In some circumstances, if you do not have a backup of a specific datafile, you can use `ALTER DATABASE CREATE DATAFILE` to create an empty replacement file that is recoverable.

2. If you restored one or more damaged datafiles to alternative locations, rename the datafiles in the control file of the database. For example, to change the filename of the datafile in tablespace `users` you might enter:

   ```
   ALTER DATABASE RENAME FILE '/d1/oracle/dbs/tbs1.f' TO '/d3/oracle/dbs/tbs1.f';
   ```

   **See Also:** *Oracle9i SQL Reference* for more information about `ALTER DATABASE RENAME FILE`

### Recovering Offline Tablespaces in an Open Database

In the final stage, you recover the datafiles in the offline tablespaces.

**To recover offline tablespaces in an open database:**

1. Connect to the database with administrator privileges. For example, connect as `SYSTEM` to database `prod1`:

   ```
   % sqlplus SYSTEM/manager@prod1
   ```

2. Start offline tablespace recovery of all damaged datafiles in one or more offline tablespaces using one step. For example, recover the `users` and `sales` tablespaces as follows:

   ```
   RECOVER TABLESPACE users, sales  # begins recovery on datafiles in users and sales
   ```

   **Note:** For maximum performance, use parallel recovery to recover the datafiles. See "Performing Media Recovery in Parallel" on page 4-25.

3. 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 with `RECOVER AUTOMATIC` or `SET AUTORECOVERY ON`, 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, then Oracle does not prompt for any. Instead, all necessary online redo log files are applied, and media recovery is complete.

4. When the damaged tablespaces are recovered up to the moment that media failure occurred, bring the offline tablespaces online. For example, to bring tablespaces `users` and `sales` online, issue the following statements:

   ```
   ALTER TABLESPACE users ONLINE;
   ALTER TABLESPACE sales ONLINE;
   ```

   **See Also:** *Oracle9i Database Administrator's Guide* for more information about creating datafiles

# Performing Incomplete User-Managed Media Recovery

This section describes the steps necessary to complete the different types of incomplete media recovery operations, and includes the following topics:

- Preparing for Incomplete Recovery
- Restoring Datafiles Before Performing Incomplete Recovery
- Performing Cancel-Based Incomplete Recovery
- Performing Time-Based Incomplete Recovery
- Performing Change-Based Incomplete Recovery

Note that if your database is affected by seasonal time changes (for example, daylight savings time), then 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 handle time changes, perform cancel-based or change-based recovery.

## Preparing for Incomplete Recovery

In this phase, you examine the source of the media problem.

**To prepare for cancel-based recovery:**

1. If you are uncertain about performing incomplete media recovery, then 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 occurs during the recovery procedure.
2. If the database is still open and incomplete media recovery is necessary, then abort the instance:

   ```
   SHUTDOWN ABORT
   ```

3. If a media failure occurred, correct the hardware problem that caused the failure. If the hardware problem cannot be repaired quickly, then proceed with database recovery by restoring damaged files to an alternative storage device.

## Restoring Datafiles Before Performing Incomplete Recovery

In this phase, you restore a whole database backup.

**To restore the files necessary for cancel-based recovery and bring them online:**

1. If the current control files do not match the physical structure of the database at the intended time of recovery, then restore a backup control file as described in

"Restoring and Re-Creating Control Files" on page 3-8. The restored control file should reflect the database's physical file structure at the point at which incomplete media recovery should finish. To determine which control file backup to use:

- 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.
- Alternatively, create a new control file to replace the missing one.

---

**Note:** If you are unable to restore a control file backup to one of the `CONTROL_FILES` locations, then edit the initialization parameter file so that this `CONTROL_FILES` location is removed.

---

2. Restore backups of *all* the datafiles of the database. All backups used to replace existing datafiles must have been taken before the intended time of recovery. For example, if you intend to recover to January 2 at 2:00 p.m., then restore all datafiles with backups completed before this time. Follow these guidelines:

| If . . . | Then . . . |
|---|---|
| You do not have a backup of a datafile | Create an empty replacement file that can be recovered as described in "Re-Creating Datafiles When Backups Are Unavailable" on page 3-7. |
| A datafile was added after the intended time of recovery | Do not restore a backup of this file because it will no longer be used for the database after recovery completes. |
| The hardware problem causing the failure has been solved and all datafiles can be restored to their default locations | Restore the files as described in "Restoring Datafiles" on page 3-6 and skip Step 5 of this procedure. |
| 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.

---

3. Start SQL*Plus and connect to Oracle with administrator privileges. For example, enter:

```
% sqlplus SYS/change_on_install@prod1
```

4. Start a new instance and mount the database:

```
STARTUP MOUNT
```

5. If one or more damaged datafiles were restored to alternative locations in Step 2, then indicate the new locations of these files to the control file of the associated database. For example, enter:

```
ALTER DATABASE RENAME FILE '/oracle/dbs/df2.f' TO '/oracle/newloc/df2.f';
```

6. Obtain the datafile names and statuses of all datafiles by checking the list of datafiles that normally accompanies the current control file or querying the V$DATAFILE view. For example, enter:

```
SELECT NAME,STATUS FROM V$DATAFILE;
```

7. Ensure that all datafiles of the database are online. All datafiles of the database requiring recovery must be online unless an offline tablespace was taken offline normally or is part of a read-only tablespace. For example, to guarantee that a datafile named `/oracle/dbs/tbs_10.f` is online, enter the following:

```
ALTER DATABASE DATAFILE '/oracle/dbs/tbs_10.f' ONLINE;
```

   If a specified datafile is already online, Oracle ignores the statement. If you prefer, create a script to bring all datafiles online at once as in the following:

```
SPOOL onlineall.sql
SELECT 'ALTER DATABASE DATAFILE '''||name||''' ONLINE;' FROM V$DATAFILE;
SPOOL OFF
SQL> @onlineall
```

## Performing Cancel-Based Incomplete Recovery

In cancel-based recovery, recovery proceeds by prompting you with the suggested filenames of archived redo log files. Recovery stops when you specify CANCEL instead of a filename or when all redo has been applied to the datafiles.

Cancel-based recovery is better than change-based or time-based recovery if you want to control which archived log terminates recovery. For example, you may know that you have lost all logs past sequence 1234, so you want to cancel recovery after log 1233 is applied.

You should perform cancel-based media recovery in these stages:

1. Prepare for recovery by backing up the database and correct any media failures as described in "Preparing for Incomplete Recovery" on page 4-16.
2. Restore backup datafiles as described in "Restoring Datafiles Before Performing Incomplete Recovery" on page 4-16. If you have a current control file, then do not restore a backup control file.
3. Perform media recovery on the restored database backup as described in the following procedure.

**To perform cancel-based recovery:**

1. Start SQL*Plus and connect to Oracle with administrator privileges. For example, enter:

   ```
   % sqlplus '/ AS SYSDBA'
   ```

2. Start a new instance and mount the database:

   ```
   STARTUP MOUNT
   ```

3. Begin cancel-based recovery by issuing the following command:

   ```
   RECOVER DATABASE UNTIL CANCEL
   ```

   If you are using a backup control file with this incomplete recovery, then specify the `USING BACKUP CONTROLFILE` option in the `RECOVER` command.

   ```
   RECOVER DATABASE UNTIL CANCEL USING BACKUP CONTROLFILE
   ```

   > **Note:** If you fail to specify the `UNTIL` clause on the `RECOVER` command, then you will not be able to open the database until a complete recovery is done.

4. Oracle applies the necessary redo log files to reconstruct the restored datafiles. Oracle supplies the name it expects to find from `LOG_ARCHIVE_DEST_1` and requests you to stop or proceed with applying the log file. Note that if the control file is a backup, then you must supply the names of the online logs if you want to apply the changes in these logs.

> **Note:** If you use an Oracle Real Application Clusters configuration, and you are performing incomplete recovery or using a backup control file, then Oracle can only compute the name of the first archived redo log file from the *first* thread. The first redo log file from the other threads must be supplied by the user. After the first log file in a given thread has been supplied, Oracle can suggest the names of the subsequent log files in this thread.

5. Continue applying redo log files until the last log has been applied to the restored datafiles, then cancel recovery by executing the following command:

   ```
   CANCEL
   ```

   Oracle returns a message indicating whether recovery is successful. Note that if you cancel recovery before all the datafiles have been recovered to a consistent SCN and then try to open the database, you will get an `ORA-1113` error if more recovery is necessary for the file. You can query `V$RECOVER_FILE` to determine whether more recovery is needed, or if a backup of a datafile was not restored prior to starting incomplete recovery.

6. Open the database in `RESETLOGS` mode. You must always reset the online logs after incomplete recovery or recovery with a backup control file. For example, enter:

   ```
   ALTER DATABASE OPEN RESETLOGS;
   ```

> **See Also:** "Opening the Database After User-Managed Media Recovery" on page 4-26

## Performing Time-Based Incomplete Recovery

This section describes how to perform the time-based media recovery procedure in the following stages:

1. Prepare for recovery by backing up the database and correct any media failures as described in "Preparing for Incomplete Recovery" on page 4-16.
2. Restore backup datafiles as described in "Restoring Datafiles Before Performing Incomplete Recovery" on page 4-16. If you have a current control file, then do not restore a backup control file.

3. Perform media recovery on the restored backup by using the following procedure.

**To perform time-based recovery:**

1. 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'. The following statement recovers the database up to a specified time:

   ```
   RECOVER DATABASE UNTIL TIME '2000-12-31:12:47:30'
   ```

   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), then indicate this in the statement used to start recovery. The following statement recovers the database up to a specified time using a control file backup:

   ```
   RECOVER DATABASE UNTIL TIME '2000-12-31:12:47:30' USING BACKUP CONTROLFILE
   ```

2. Apply the necessary redo log files to recover the restored datafiles. Unless the application of files is automated, Oracle supplies the name it expects to find from LOG_ARCHIVE_DEST_1 and requests you to stop or proceed with applying the log file. If the control file is a backup, then you after the archived logs have been applied you must supply the names of the online logs in order to apply their changes.
3. Apply redo logs until the last required redo log 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.
4. Open the database in RESETLOGS mode. You must always reset the online logs after incomplete recovery or recovery with a backup control file. For example, enter:

   ```
   ALTER DATABASE OPEN RESETLOGS;
   ```

**See Also:** "Opening the Database After User-Managed Media Recovery" on page 4-26

## Performing Change-Based Incomplete Recovery

This section describes how to perform recovery to a specified SCN in these stages:

1. Prepare for recovery by backing up the database and correct any media failures as described in "Preparing for Incomplete Recovery" on page 4-16.

2. Restore backup datafiles as described in "Restoring Datafiles Before Performing Incomplete Recovery" on page 4-16. If you have a current control file, then do not restore a backup control file.
3. Perform media recovery on the restored backup by using the following procedure.

**To perform change-based recovery:**

1. Begin change-based recovery, specifying the SCN for recovery termination. The SCN is specified as a decimal number without quotation marks. For example, to recover through SCN 10034 issue:

```
RECOVER DATABASE UNTIL CHANGE 10034;
```

2. 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_1` and requests you to stop or proceed with applying the log file. If the control file is a backup, then you after the archived logs have been applied you must supply the names of the online logs in order to apply their changes.
3. 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 SCN, and returns a message indicating whether recovery is successful.
4. Open the database in `RESETLOGS` mode. You must always reset the online logs after incomplete recovery or recovery with a backup control file. For example, enter:

```
ALTER DATABASE OPEN RESETLOGS;
```

> **See Also:** "Opening the Database After User-Managed Media Recovery" on page 4-26

## Recovering a Database in NOARCHIVELOG Mode

If a database is in `NOARCHIVELOG` mode and a media failure damages some or all of the datafiles, then the only option for recovery is usually to restore the most recent whole database backup. If you are using Export to supplement regular backups, then you can also attempt to restore the database by importing an exported backup

of the database into a re-created database or a database restored from an old backup.

The disadvantage of NOARCHIVELOG mode is that to recover the database from the time of the most recent full backup up to the time of the media failure, you have to reenter manually all of the changes executed in that interval. If the database was in ARCHIVELOG mode, however, the redo log covering this interval would have been available as archived log files or online log files. Using archived redo logs would have enabled you to use complete or incomplete recovery to reconstruct your database, thereby minimizing the amount of lost work.

If you have a database damaged by media failure and operating in NOARCHIVELOG mode, and if you want to restore from your most recent consistent whole database backup (your only option at this point), then follow the procedure below.

## Restoring the Database to its Default Location

In this scenario, the media failure is repaired so that you are able to restore all database files to their original location.

**To restore the most recent whole database backup to the default location:**

1. If the database is open, then shut down the database. For example, enter:

   ```
   SHUTDOWN IMMEDIATE
   ```

2. If possible, correct the media problem so that the backup database files can be restored to their original locations.

3. Restore the most recent whole database backup with operating system commands as described in "Restoring Datafiles" on page 3-6. Restore all of the datafiles and control files of the whole database backup, not just the damaged files. The following example restores a whole database backup:

   ```
   % cp /oracle/work/BACKUP/tbs* /oracle/dbs  # restores datafiles
   % cp /oracle/work/BACKUP/cf.f /oracle/dbs  # restores control file
   ```

4. Because online redo logs are not backed up, you cannot restore them with the datafiles and control files. In order to allow Oracle to reset the online redo logs, you must first mimic incomplete recovery:

   ```
   RECOVER DATABASE UNTIL CANCEL
   CANCEL
   ```

5. Open the database in RESETLOGS mode. This command resets the current redo log sequence to 1:

```
ALTER DATABASE OPEN RESETLOGS;
```

A RESETLOGS operation invalidates all redo in the online logs. 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 failure.

## Restoring the Database to a New Location

In this scenario, you restore the database files to an alternative location because the original location is damaged by a media failure.

**To restore the most recent whole database backup to a new location:**

1. If the database is open, then shut it down. For example, enter:

```
SHUTDOWN IMMEDIATE
```

2. Restore all of the datafiles and control files of the whole database backup, not just the damaged files. If the hardware problem has not been corrected and some or all of the database files must be restored to alternative locations, then restore the whole database backup to a new location. For example, enter:

```
% cp /disk2/BACKUP/tbs* /disk3/oracle/dbs  # default location
% cp /disk2/BACKUP/cf.f /disk3/oracle/dbs  # new location
% cp /disk2/BACKUP/system01.dbf /disk4/temp  # new location
```

3. If necessary, edit the restored parameter file to indicate the new location of the control files. For example:

```
CONTROL_FILES = "/disk3/oracle/dbs/cf.f"
```

4. Start an instance using the restored and edited parameter file and mount, but do not open, the database. For example:

```
STARTUP MOUNT
```

5. If the restored datafile filenames will be different, then rename the restored datafiles in the control file. For example, you might enter:

```
ALTER DATABASE RENAME FILE '/disk1/oracle/dbs/system01.dbf' TO
                           '/disk4/temp/system01.dbf';
```

6. If the online redo logs were located on a damaged disk, and the hardware problem is not corrected, then specify a new location for each online log. For example, enter:

```
ALTER DATABASE RENAME FILE '/disk1/oracle/dbs/log1.f' TO '/disk3/oracle/dbs/log1.f';
ALTER DATABASE RENAME FILE '/disk1/oracle/dbs/log2.f' TO '/disk3/oracle/dbs/log2.f';
```

7. Because online redo logs are not backed up, you cannot restore them with the datafiles and control files. In order to allow Oracle to reset the online redo logs, you must first mimic incomplete recovery:

```
RECOVER DATABASE UNTIL CANCEL;
CANCEL;
```

8. Open the database in RESETLOGS mode. This command resets the current redo log sequence to 1:

```
ALTER DATABASE OPEN RESETLOGS;
```

A RESETLOGS operation invalidates all redo in the online logs. 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 failure.

> **See Also:** *Oracle9i Database Administrator's Guide* for more information about renaming and relocating datafiles, and *Oracle9i SQL Reference* for more information about ALTER DATABASE RENAME FILE

## Performing Media Recovery in Parallel

Use **parallel media recovery** to tune the roll forward phase of media recovery. In parallel media recovery, Oracle uses a "division of labor" approach to allocate different processes to different data blocks while rolling forward, thereby making the procedure more efficient. For example, if parallel recovery is performed with PARALLEL 4, and only one datafile is recovered, then four spawned processes read blocks from the datafile and apply records instead of only one process.

> **Note:** Typically, recovery is I/O-bound on reads to data blocks. Parallelism at the block level may only help recovery performance if it increases total I/Os, for example, by bypassing operating system restrictions on asynchronous I/Os. Systems with efficient asynchronous I/O typical see little improvement from using parallel media recovery.

The SQL*Plus RECOVER PARALLEL command specifies parallel media recovery (the default is NOPARALLEL). This command selects a degree of parallelism equal to the number of CPUs available on all participating instances times the value of the PARALLEL_THREADS_PER_CPU initialization parameter.

The format for the RECOVER PARALLEL command is the following:

```
RECOVER PARALLEL integer;
```

where:

| | |
|---|---|
| *integer* | Specifies the number of recovery processes used for media recovery. If multiple instances exist in a Real Application Clusters configuration, then Oracle decides how to distribute these recovery processes among these instances. If *integer* is not specified, then Oracle picks a default number of recovery processes. |

**Note:** The RECOVERY_PARALLELISM initialization parameter specifies the number of concurrent recovery processes for instance or crash recovery *only*. Media recovery is not affected.

**See Also:**

- *Oracle9i Database Performance Guide and Reference* for more information on parallel recovery
- *SQL*Plus User's Guide and Reference* for more information about the SQL*Plus RECOVER . . . PARALLEL statement

# Opening the Database After User-Managed Media Recovery

Whenever you perform incomplete recovery or recovery with a backup control file, you must reset the online logs when you open the database. The new version of the reset database is called a new **incarnation**. All archived redo logs generated after the point of the RESETLOGS on the old incarnation are invalid in the new incarnation.

If you perform complete recovery, then you do not have to open the database with the RESETLOGS option. All previous backups and archived logs created during the lifetime of this incarnation of the database are valid.

This section contains the following topics:

- About RESETLOGS Operations
- Determining Whether to Reset the Online Redo Logs

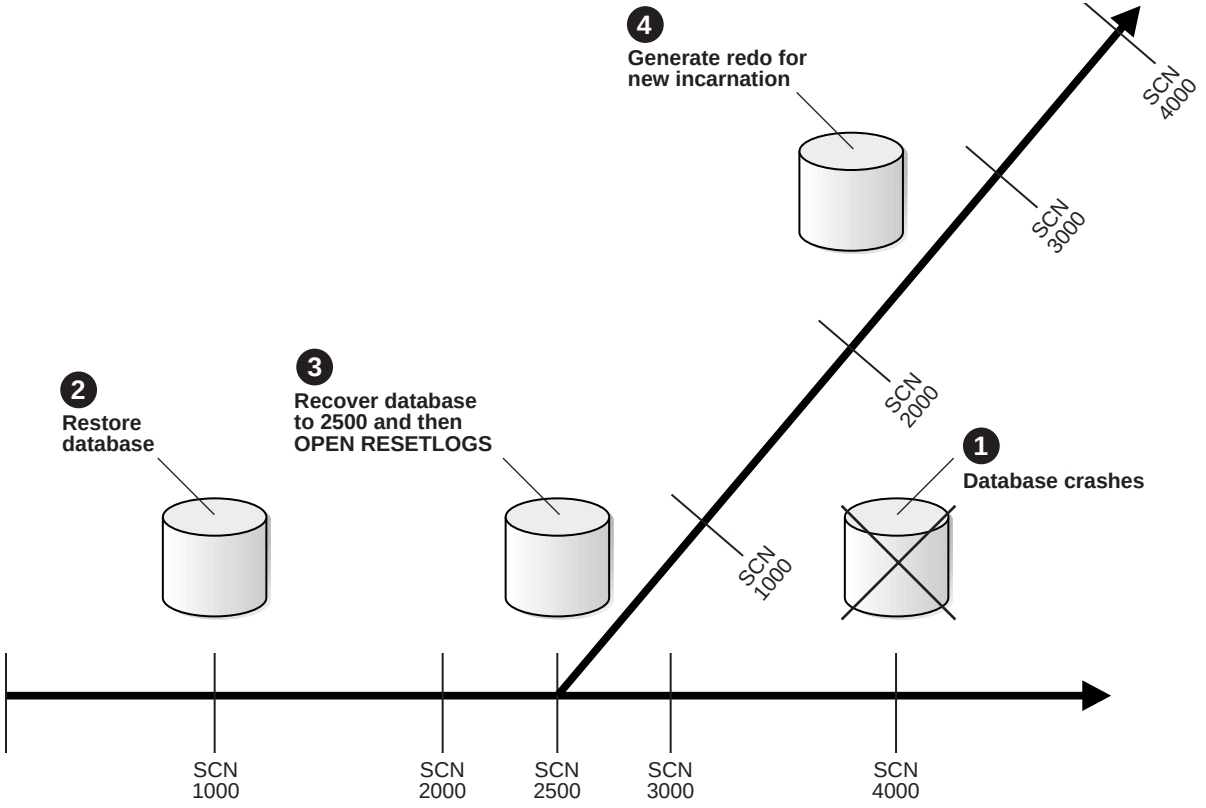
## About RESETLOGS Operations

Whenever you open the database with the RESETLOGS option, all datafiles get a new RESETLOGS SCN and time stamp. Archived redo logs also have these two values in their file header. Because Oracle will not apply an archived redo log to a datafile unless the RESETLOGS SCN and time stamps match, the RESETLOGS operations prevents you from corrupting your datafiles with old archived logs.

Figure 4–1 shows the case of a database that can only be recovered to SCN 2500 because an archived redo log is missing. At SCN 4000, the database crashes. You restore the SCN 1000 backup and prepare for complete recovery. Unfortunately, one of your archived redo logs is corrupted. The log before the missing log contains SCN 2500, so you recover to this point and open with the RESETLOGS option.

As the diagram illustrates, you generate new changes in the new incarnation of the database, eventually reaching SCN 4000. The changes between SCN 2500 and SCN 4000 for the new incarnation of the database are completely different from the changes between SCN 2500 and SCN 4000 for the old incarnation. Oracle does not allow you to apply logs from an old incarnation to the new incarnation. You cannot restore backups from before SCN 2500 in the old incarnation to the new incarnation.

***Figure 4–1 Creating a New Database Incarnation***




## Determining Whether to Reset the Online Redo Logs

To open the database with the RESETLOGS option, all datafiles must be recovered to the same SCN. If a backup control file is restored, then the backup control file must also be recovered to the same SCN.

The RESETLOGS option is always *required* after incomplete media recovery or recovery using a backup control file. Resetting the redo log does the following:

- Discards any redo information that was not applied during recovery, ensuring that it will never be applied.
- Reinitializes the control file metadata about online redo logs and redo threads.
- Erases 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.

> **Caution:** Resetting the redo log discards all changes to the database contained in the online logs. Hence, after opening RESETLOGS, you cannot perform recovery again to a point within the reset logs.

Use the following rules when deciding whether to specify RESETLOGS or NORESETLOGS:

- Always specify the RESETLOGS option after incomplete recovery. For example, you must have specified a previous time or SCN, not one in the future.
- Always specify RESETLOGS if you used a backup of the control file in recovery, regardless of whether you performed complete or incomplete recovery.
- Specify either no option or the NORESETLOGS option after performing complete media recovery (unless you used a backup control file, in which case you must open with the RESETLOGS option).
- Avoid specifying the RESETLOGS option if you are using the archived logs of the database for a standby database. If you must reset the online logs, then you have to re-create the standby database.

### Executing the ALTER DATABASE OPEN Statements

To preserve the log sequence number when opening a database after media recovery, execute either of the following statements:

```
ALTER DATABASE OPEN NORESETLOGS;
ALTER DATABASE OPEN;
```

To reset the log sequence number when opening a database after recovery and thereby create a new incarnation of the database, execute the following statement:

```
ALTER DATABASE OPEN RESETLOGS;
```

If you open with the RESETLOGS option, Oracle returns different messages depending on whether recovery was complete or incomplete. If the recovery was complete, then the following message appears in the `alert_`*`SID`*`.log` file:

```
RESETLOGS after complete recovery through change scn
```

If the recovery was incomplete, then this message is reported in the alert_*SID*.log file, where *scn* refers to the end point of incomplete recovery:

```
RESETLOGS after incomplete recovery UNTIL CHANGE scn
```

If you attempt to OPEN RESETLOGS when you should not, or if you neglect to reset the log when you should, then Oracle returns an error and does not open the database. Correct the problem and try again.

> **See Also:** "About User-Managed Media Recovery Problems" on page 5-2 for descriptions of situations that can cause ALTER DATABASE OPEN RESETLOGS to fail

## Following Up After a RESETLOGS Operation

This section describes actions that you should perform after opening the database in RESETLOGS mode.

### Making a Whole Database Backup

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 is to repeat the procedures you just finished, up to resetting the logs. You do not need to make another backup of the database if you did not reset the log sequence.

In general, backups made before a RESETLOGS operation are not allowed in the new incarnation. There is, however, an exception to the rule: you can restore a pre-RESETLOGS backup *only if* Oracle does not need to access archived redo logs from before the RESETLOGS to perform recovery.

> **See Also:** "Recovering a Backup Created Before a RESETLOGS" on page 4-31.

### Checking the Alert Log

After opening the database using the RESETLOGS option, check the alert_*SID*.log to see whether Oracle 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, then 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 datafile corresponding to MISSING*nnnn* can be made accessible by renaming MISSING*nnnn* so that it points to the datafile only if the datafile was read-only or offline normal between the time the backup was taken to the point where the RESETLOGS is issued.On the other hand, if MISSING*nnnn* corresponds to a datafile that was not read-only or offline normal during the recovery period, 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, you must drop the tablespace containing the datafile.

In contrast, if a datafile indicated in the control file is not in the data dictionary, Oracle removes references to it from the new control file. In both cases, Oracle includes an message in the alert_*SID*.log file to let you know what was found.

## Recovering a Backup Created Before a RESETLOGS

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 technique was disk maintenance, which required the database to be backed up, deleted, the disks reconfigured, and the database restored. DBAs realized that by not restarting in RESETLOGS mode, they would not have to back up the database immediately after the restore. This backup was required since it was impossible to perform recovery on a backup taken before the RESETLOGS—especially if any errors occurred after resetting the logs.

### Restoring Backups Created Before a RESETLOGS

You can restore the following backups made before a RESETLOGS in a new incarnation:

- Backups of a tablespace made after it was made read-only (only if it was not made read / write again before the RESETLOGS)
- Backups of a tablespace after it was taken offline-normal (only if it was not brought online again before the RESETLOGS)
- Consistent backups of read / write tablespaces made after recovery ends and before you open RESETLOGS, that is, you do not perform further recovery or alter the datafiles between the backup and the RESETLOGS—but only if you have a control file that is valid after you open RESETLOGS

You are prevented from restoring backups of read / write tablespaces that were *not* made immediately before the RESETLOGS. This restriction applies even if no changes were made to the datafiles in the read / write tablespace between the

backup and the ALTER DATABASE OPEN RESETLOGS. Because the checkpoint in the datafile header of a backup will be older than the checkpoint in the control file, Oracle has to search the archived logs to determine whether changes need to be applied—and the archived logs generated prior to the RESETLOGS are not valid in the new incarnation.

### Restoring a Backup Created Before a RESETLOGS: Scenario

The following scenario illustrates a situation when you can use a backup created before a RESETLOGS. Suppose you wish 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 you perform the following actions:

1. Shut down the database consistently. For example:

   ```
   SHUTDOWN IMMEDIATE
   ```

2. Perform a whole database backup. For example, enter

   ```
   % cp /oracle/dbs/* /oracle/backup
   ```

   **Note:** At this point you must not reopen the database.

3. Perform operating system maintenance.
4. Restore the datafiles and control files from the backup that you just made. For example, enter:

   ```
   % cp /oracle/backup/* /oracle/dbs
   ```

5. Mount the database. For example, enter:

   ```
   STARTUP MOUNT
   ```

6. Initiate cancel-based recovery. For example, enter:

   ```
   RECOVER DATABASE UNTIL CANCEL
   ```

7. Open the database with the RESETLOGS option. For example, enter:

   ```
   ALTER DATABASE OPEN RESETLOGS;
   ```

On Saturday morning the scheduled jobs run, generating archived logs. If a hardware error occurs Saturday night that requires you to restore the whole

database, then you can restore the backup taken immediately before opening with the RESETLOGS option, and roll forward using the logs produced on Saturday.

On Saturday night you do the following:

1. Abort the instance (if it still exists). For example, enter:

```
SHUTDOWN ABORT
```

2. Restore all damaged files from the backup made on Friday night. For example, enter:

```
% cp /oracle/backup/* /oracle/dbs
```

> **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.

3. Begin complete recovery, applying all the archived logs produced on Saturday. Use SET AUTORECOVERY ON to automate the log application. For example, enter:

```
SET AUTORECOVERY ON
RECOVER DATABASE
```

4. Open the database. For example, enter:

```
STARTUP
```

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

## Interrupting User-Managed Media Recovery

If you start media recovery and must then interrupt it, for example, because a recovery operation must end for the night and resume the next morning, then take 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, you can resume it later with the RECOVER command. Recovery resumes where it left off when it was canceled.

Several factors may cause you to restart recovery. For example, if 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.

If you are recovering parts of database with RECOVER TABLESPACE or RECOVER DATAFILE, then you will have to restart recovery and finish recovery in order to make these parts of the database available.

If you are performing incomplete recovery of the whole database, then you may be able to open the database read only or RESETLOGS after canceling media recovery. This strategy can succeed if all datafiles have been recovered to a consistent SCN. For serial media recovery, this works even after interrupting media recovery. If not all datafiles have been recovered to a consistent SCN, then the RESETLOGS may fail, requiring you to perform more media recovery.

# User-Managed Media Recovery Restrictions

Before performing media recovery, make sure that you understand the following issues:

- User-Managed Recovery of Unrecoverable Tables and Indexes
- User-Managed Recovery of Read-Only Tablespaces with a Noncurrent Control File

## User-Managed Recovery of Unrecoverable Tables and Indexes

You can create tables and indexes with the CREATE TABLE AS SELECT statement. 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, you cannot recover objects created unrecoverable, even if you are running in ARCHIVELOG mode.

---

**Note:** If you cannot afford to lose tables or indexes created unrecoverable, then make a backup after the unrecoverable table or index is created.

---

Be aware that when you perform media recovery, and some tables or indexes are created as recoverable while others are unrecoverable, the unrecoverable objects are marked logically corrupt by the RECOVER operation. Any attempt to access the unrecoverable objects returns an ORA-01578 error message. Drop the unrecoverable objects and re-create 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 media recovery. The table was unrecoverable (and thus marked as corrupt after recovery), however, 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:** *Oracle9i Data Guard Concepts and Administration* for information about the impact of unrecoverable operations on a standby database.

## User-Managed Recovery of Read-Only Tablespaces with a Noncurrent Control File

If you have a read-only tablespace on read-only or slow media, then you may encounter errors or poor performance when performing media recovery with the USING BACKUP CONTROLFILE option. This situation occurs when the backup control file indicates that a tablespace was read / write when the control file was backed up. In this case, media recovery may attempt to write to the files. For read-only media, Oracle issues an error saying that it cannot write to the files. For slow media, such as a hierarchical storage system backed up by tapes, performance may suffer.

To avoid these recovery problems, use current control files rather than backups to recover the database. If you need to use a backup control file, then you can also avoid this problem if the read-only tablespace has not suffered a media failure.

### Recovery of Read-Only or Slow Media with a Backup Control File

You have these alternatives for recovering read-only and slow media when using a backup control file:

- Take datafiles from read-only tablespaces offline before doing recovery with a backup control file, and then bring the files online at the end of media recovery.
- Use the correct version of the control file for the recovery. If the tablespace will be read-only when recovery completes, 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, then the control file must be from a time when the tablespace was read / write.

### Recovery of Read-Only Files with a Re-Created Control File

If a current or backup control file is unavailable for the recovery, then you can execute a CREATE CONTROLFILE statement as described in "Losing All Current and Backup Control Files" on page 3-13. Read-only files should not be listed in the CREATE CONTROLFILE statement so that recovery can skip these files. No recovery is required for read-only files unless you restored backups of these files from a time when they were read/write.

After you create a new control file and attempt to mount and open the database, Oracle performs a data dictionary check against the files listed in the control file. Any files that were not listed in the CREATE CONTROLFILE statement but are present in the data dictionary have entries created for them in the control file. Oracle names these files as MISSING*nnnnn*, where *nnnnn* is a five digit number starting with 0.

After the database is open, rename the read-only files to their correct filenames by executing the ALTER DATABASE RENAME FILE statement for all the files whose name is prefixed with MISSING.

To prepare for a scenario in which you might have to re-create the control file, run the following statement when the database is mounted or open to obtain the CREATE CONTROLFILE syntax:

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

This SQL statement produces a trace file that you can edit and then use as a script to re-create the control file in a recovery scenario. You can specify either the RESETLOGS or NORESETLOGS (default) keywords to generate CREATE CONTROLFILE ... RESETLOGS or CREATE CONTROLFILE ... NORESETLOGS versions of the script.

Note that all the restrictions related to read-only files in CREATE CONTROLFILE statements also apply to offline normal tablespaces, except that you need to bring the tablespace online after the database is open. You should leave out tempfiles from the CREATE CONTROLFILE statement and add them after database open.

> **See Also:** "Backing Up the Control File to a Trace File" on page 2-19 to learn about taking trace backups of the control file

# 5

# Troubleshooting User-Managed Media Recovery

This chapter describes how to troubleshoot user-managed media recovery, and includes the following topics:

# About User-Managed Media Recovery Problems

Table 5–1 describes potential problems that can occur during media recovery.

***Table 5–1 Media Recovery Problems***

| Problem | Description |
|---|---|
| Missing or misnamed archived log | Recovery stops because Oracle cannot find the archived log recorded in the control file. |
| When you attempt to open the database, error ORA-1113 indicates that a file needs media recovery | This error commonly occurs because:<br>■ You are performing incomplete recovery but failed to restore all needed datafile backups.<br>■ Incomplete recovery stopped before datafiles reached a consistent SCN.<br>■ You are recovering datafiles from an online backup, but not enough redo was applied to make the datafiles consistent.<br>■ You are performing recovery with a backup control file, and did not specify the location of a needed online log.<br>■ A datafile is undergoing media recovery when you attempt to open the database.<br>■ Datafiles needing recovery were not brought online before executing RECOVER DATABASE, and so were not recovered. |
| Redo record problems | Two possible cases are as follows:<br>■ Recovery stops because of failed consistency checks, a problem called **stuck recovery**. Stuck recovery can occur when an underlying operating system or storage system loses a write issued by Oracle during normal operation of the database.<br>■ Oracle signals an internal error when applying the redo. This problem can be caused by an Oracle bug. If checksums are not being used, it can also be caused by corruptions to the redo or data blocks. |
| Corrupted archived logs | Logs may be corrupted while they are stored on or copied between storage systems. If DB_BLOCK_CHECKSUM is enabled, then Oracle usually signals checksum errors. If checksumming is not on, then log corruption may appear as a problem with redo. |
| Corrupted data blocks | A datafile backup may have contained a corrupted data block, or the data block may become corrupted either during recovery or when it was copied to the backup. If checksums are being used, then Oracle signals a checksum error. Otherwise, the problem may also appear as a redo corruption. |
| Random problems | Memory corruptions and other transient problems can occur during recovery. |

The symptoms of media recovery problems are usually external or internal errors signaled during recovery. For example, an external error indicates that a redo block or a data block has failed checksum verification checks. Internal errors can be caused by either bugs in Oracle or errors arising from the underlying operating system and hardware.

If serial media recovery encounters a problem while recovering a database backup, whether it is a stuck recovery problem or a problem during redo application, Oracle always stops and leaves the datafiles undergoing recovery in a consistent state, that is, at an SCN preceding the failure. You can then do one of the following:

- Open the database read-only to investigate the problem.
- Open the database with the RESETLOGS option, as long as the requirements for opening RESETLOGS have been met (as described in "Opening the Database After User-Managed Media Recovery" on page 4-26). Note that the RESETLOGS restrictions apply to opening the standby database as well, because a standby database is updated by a form of media recovery.

In general, opening the database read-only or opening with the RESETLOGS option require all online datafiles to be recovered to the same SCN. If this requirement is not met, then Oracle may signal ORA-1113 or other errors when you attempt to open. Some common causes of ORA-1113 are described in Table 5–1.

The basic methodology for responding to media recovery problems occurs in the following phases:

1. Try to identify the cause of the problem. Run a trial recovery if needed.
2. If the problem is related to missing logs or you suspect there is a log, memory, or data block corruption, then try to resolve it using the methods described in Table 5–2.
3. If you cannot resolve the problem using the methods described in Table 5–2, then do one of the following:
   - Open the database with the RESETLOGS option if you are recovering a whole database backup. If you have performed serial media recovery, then the database contains all the changes up to but not including the changes at the SCN where the corruption occurred. No changes from this SCN onward are in the recovered part of the database. If you have restored online backups, opening RESETLOGS succeeds only if you have recovered through all the ALTER . . . END BACKUP operations in the redo stream.

- Proceed with recovery by allowing media recovery to corrupt data blocks. After media recovery completes, try performing block media recovery using RMAN.
- Call Oracle Support Services as a last resort.

**See Also:** *Oracle9i Recovery Manager User's Guide and Reference* to learn about block media recovery

# Investigating the Media Recovery Problem: Phase 1

If media recovery encounters a problem, then obtain as much information as possible after recovery halts. You do not want to waste time fixing the wrong problem, which may in fact make matters worse.

The goal of this initial investigation is to determine whether the problem is caused by incorrect setup, corrupted logs, corrupted data blocks, memory corruption, or other problems. If you see a checksum error on a data block, then the data block is corrupted. If you see a checksum error on a redo log block, then the redo log is corrupted.

Sometimes the cause of a recovery problem can be difficult to determine. Nevertheless, the methods in this chapter allow you to quickly recover a database sometimes even when you do not completely understand the cause of the problem.

**To investigate media recovery problems:**

1. Examine the `alert.log` to see whether the error messages give general information about the nature of the problem. For example, does the `alert_`*`SID`*`.log` indicate any checksum failures? Does the `alert_`*`SID`*`.log` indicate that media recovery may have to corrupt data blocks in order to continue?
2. Check the trace file generated by the Oracle process during recovery. It may contain additional error information.

# Trying to Fix the Recovery Problem Without Corrupting Blocks: Phase 2

Depending on the type of media recovery problem you suspect, you have different solutions at your disposal. You can try one or a combination of the methods described in Table 5–2. Note that these methods are fairly safe: in almost all cases, they should not cause any damage to the database.

***Table 5–2 Media Recovery Solutions***

| If you suspect . . . | Then . . . |
|---|---|
| Missing / misnamed archived logs | Check to see whether you entered the correct filename. If you did, then check to see whether the log is missing from the operating system. If it is missing, and you have a backup, then restore the backup and apply the log. If you do not have a backup, then if possible perform incomplete recovery up to the point of the missing log. |
| `ORA-1113` for `ALTER DATABASE OPEN` | Review the many causes of this error in Table 5–1. Make sure that all read / write datafiles requiring recovery are online. If you use a backup control file for recovery, then the control file and datafiles must be at a consistent SCN for the database to be opened. If you do not have the necessary redo, then you must re-create the control file. |
| Corrupt archived logs | The log is corrupted if the checksum verification on the log redo block fails. If `DB_BLOCK_CHECKSUM` is not enabled either during the recovery session or when the database generated the redo, then recovery problems may be caused by corrupted logs. If the log is corrupt and an alternate copy of the corrupt log is available, then try to apply it and see whether this tactic fixes the problem.<br>The `DB_BLOCK_CHECKSUM` initialization parameter determines whether checksums are computed for redo log and data blocks. |
| Memory corruption or other transient problems | Shut down the database and then restart recovery. In some cases, this tactic fixes the problem. Oracle should leave the database in a consistent state if the second attempt also fails. |
| Corrupt data blocks | Restore and recover the datafile again with user-managed methods, or restore and recover individual data blocks with the RMAN `BLOCKRECOVER` command. This tactic may fix the problem.<br>Note that a data block is corrupted if the checksum verification on the data block fails. If `DB_BLOCK_CHECKING` is not enabled, a corrupted data block problem may appear as a redo problem. If you must proceed with recovery, then you may want to corrupt the block now and continue recovery, and use RMAN to perform block media recovery later. |

If you cannot fix the problem with the methods described in Table 5–2, then there may be no easy way to fix the problem without losing data. You have these options:

- Open the database with the `RESETLOGS` option (for whole database recovery). This solution discards all changes after the point where the redo problem occurred, but guarantees a logically consistent database.
- Allow media recovery to corrupt one or more data blocks and proceed with media recovery. This option will only succeed if the `alert_`*`SID`*`.log` indicates that recovery can continue if it is allowed to corrupt a data block, which should be the case for most recovery problems. This option is best if it is important to bring up the database quickly and recover all changes. If you are contemplating

this option as a last resort, then proceed to "Deciding Whether to Allow Recovery to Corrupt Blocks: Phase 3" on page 5-6.

> **See Also:** *Oracle9i Recovery Manager User's Guide and Reference* to learn how to perform block media recovery with the BLOCKRECOVER command

## Deciding Whether to Allow Recovery to Corrupt Blocks: Phase 3

When media recovery encounters a problem, the `alert_SID.log` may indicate that recovery can continue if it is allowed to corrupt the data block causing the problem. The `alert_SID.log` always contains information about the block: its block type, block address, the tablespace it belongs to, and so forth. For blocks containing user data, the log may also report the data object number.

In this case, Oracle can proceed with recovery if it is allowed to mark the problem block as corrupt. Nevertheless, this response is not always advisable. For example, if the block is an important block in the SYSTEM tablespace, marking the block as corrupt can eventually prevent you from opening the recovered database. Another consideration is whether the recovery problem is isolated. If this problem is followed immediately by many other problems in the redo stream, then you may want to open the database with the RESETLOGS option.

For a block containing user data, you can usually query the database to find out which object or table owns this block. If the database is not open, then you should be able to open the database read-only, even if you are recovering a whole database backup. The following example cancels recovery and opens read-only:

```
CANCEL
ALTER DATABASE OPEN READ ONLY;
```

Assume that the data object number reported in the `alert_SID.log` is `8031`. You can determine the owner, object name, and object type by issuing this query:

```
SELECT OWNER, OBJECT_NAME, SUBOBJECT_NAME, OBJECT_TYPE
FROM DBA_OBJECTS
WHERE DATA_OBJECT_ID = 8031;
```

To determine whether a recovery problem is isolated, you can run a diagnostic **trial recovery**, which scans the redo stream for problems but does not actually make any changes to the recovered database. If a trial recovery discovers any recovery problems, it reports them in the `alert_SID.log`. You can use the RECOVER ... TEST statement to invoke trial recovery.

**See Also:** "Performing Trial Recovery" on page 5-8

After you have done these investigations, you can follow the guidelines in Table 5–3 to decide whether to allow recovery to corrupt blocks.

***Table 5–3 Guidelines for Allowing Recovery to Permit Corruption***

| If the problem is . . . | and the block is . . . | Then . . . |
|---|---|---|
| not isolated | | You should probably open the database with the RESETLOGS option. This response is important for stuck recovery problems, because stuck recovery can be caused by the operating system or a storage system losing writes. If an operating system or storage system suddenly fails, it can cause stuck recovery problems on several blocks. |
| isolated | in the SYSTEM tablespace | Do not corrupt the block, because it may eventually prevent you from opening the database. However, sometimes data in the SYSTEM tablespace is unimportant. If you must corrupt a SYSTEM block and recover all changes, then call Oracle Support. |
| isolated | index data | Consider corrupting index blocks because the index can be rebuilt later after the database has been recovered. |
| isolated | user data | Decide based on the importance of the data. If you continue with datafile recovery and corrupt a block, you lose data in the block. However, you can use RMAN to perform block media recovery later after datafile recovery completes. If you open RESETLOGS, then the database is consistent but loses any changes made after the point where recovery was stopped. |
| isolated | rollback or undo data | Consider corrupting the rollback or undo block. Corrupting a rollback or undo block does not harm the database if the transactions that generated the undo are never rolled back. However, if those transactions are rolled back, then corrupting the undo block can cause problems. If you are unsure, then call Oracle Support. |

**See Also:** "Performing Trial Recovery" on page 5-8 to learn how to perform trial recovery, and "Allowing Recovery to Corrupt Blocks: Phase 4" on page 5-7 if you decide to corrupt blocks

## Allowing Recovery to Corrupt Blocks: Phase 4

If you decide to allow recovery to proceed in spite of block corruptions, then run the RECOVER command with the ALLOW *n* CORRUPTION clause, where *n* is the number of allowable corrupt blocks.

**To allow recovery to corrupt blocks:**

1. Ensure that all normal recovery preconditions are met. For example, if the database is open, then take tablespaces offline before attempting recovery.
2. Run the RECOVER command, allowing a single corruption, repeating as necessary for each corruption to be made. The following statements shows a valid example:

```
RECOVER DATABASE ALLOW 1 CORRUPTION
```

# Performing Trial Recovery

This section contains these topics:

- About Trial Recovery
- How Trial Recovery Works
- Initiating Trial Recovery

## About Trial Recovery

When problems such as stuck recovery occur, you have a difficult choice. If the block is relatively unimportant, and if the problem is isolated, then it is better to corrupt the block. But if the problem is not isolated, then it may be better to open the database with the RESETLOGS option.

Because of this situation, Oracle supports trial recovery. A trial recovery applies redo in a way similar to normal media recovery, but it never writes its changes to disk and it always rolls back its changes. Trial recovery occurs only in memory.

> **See Also:** "Allowing Recovery to Corrupt Blocks: Phase 4" on page 5-7

## How Trial Recovery Works

By default, if a trial recovery encounters a stuck recovery or similar problem, then it always marks the data block as corrupt in memory when this action can allow recovery to proceed. Oracle writes errors generated during trial recovery to alert files. Oracle clearly marks these errors as test run errors.

Like normal media recovery, trial recovery can prompt you for archived log filenames and ask you to apply them. Trial recovery ends when:

- Oracle runs out of the maximum number of buffers in memory that trial recovery is permitted to use
- An unrecoverable error is signaled, that is, an error that cannot be resolved by corrupting a data block
- You cancel or interrupt the recovery session
- The next redo record in the redo stream changes the control file
- All requested redo has been applied

When trial recovery ends, Oracles removes all effects of the test run from the system—except the possible error messages in the alert files. If the instance fails during trial recovery, then Oracle removes all effects of trial recovery from the system because trial recovery never writes changes to disk.

Trial recovery lets you foresee what problems might occur if you were to continue with normal recovery. For problems caused by ongoing memory corruption, trial recovery and normal recovery can encounter different errors.

## Initiating Trial Recovery

You can use the TEST option for any RECOVER command. For example, you can start SQL*Plus and then issue any of the following commands:

```
RECOVER DATABASE TEST
RECOVER DATABASE USING BACKUP CONTROLFILE UNTIL CANCEL TEST
RECOVER TABLESPACE users TEST
RECOVER DATABASE UNTIL CANCEL TEST
```

By default, trial recovery always attempts to corrupt blocks in memory if this action allows trial recovery to proceed. In other words, trial recovery by default can corrupt an unlimited number of data blocks. You can specify the ALLOW *n* CORRUPTION clause on the RECOVER . . . TEST statement to limit the number of data blocks trial recovery can corrupt in memory.

Note that a trial recovery command is usable in any scenario in which a normal recovery command is usable. Nevertheless, you should only need to run trial recovery when recovery runs into problems.

# 6

# User-Managed Media Recovery Scenarios

This chapter describes how to recover from common media failures, and includes the following topics:

# Recovering After the Loss of Datafiles: Scenarios

If a media failure affects datafiles, then the recovery procedure depends on:

- The archiving mode of the database: ARCHIVELOG or NOARCHIVELOG
- The type of media failure
- The files affected by the media failure

The following sections explain the appropriate recovery strategies based on the database archiving mode:

- Losing Datafiles in NOARCHIVELOG Mode
- Losing Datafiles in ARCHIVELOG Mode

## Losing Datafiles in NOARCHIVELOG Mode

If either a permanent or temporary media failure affects any datafiles of a database operating in NOARCHIVELOG mode, then Oracle automatically shuts down the database. Depending on the type of media failure, use one of the following recovery methods:

| If the media failure is . . . | Then . . . |
|---|---|
| Temporary | Correct the hardware problem and restart the database. Usually, crash recovery is possible, and all committed transactions can be recovered using the online redo log. |
| Permanent | Follow the procedure "Performing Complete User-Managed Media Recovery" on page 4-9. |

## Losing Datafiles in ARCHIVELOG Mode

If either a permanent or temporary media failure affects the datafiles of a database operating in ARCHIVELOG mode, then the following scenarios can occur.

| Damaged Datafiles | Database Status | Solution |
|---|---|---|
| Datafiles in the SYSTEM tablespace or datafiles with active rollback or undo segments. | Oracle shuts down. | If the hardware problem is temporary, then fix it and restart the database. Usually, crash recovery recovers lost transactions. If the hardware problem is permanent, then refer to "Performing Closed Database Recovery" on page 4-9. |
| Datafiles not in the SYSTEM tablespace or datafiles that do not contain active rollback or undo segments. | Oracle takes affected datafiles offline, but the database stays open. | If the unaffected portions of the database must remain available, then do not shut down the database. Take tablespaces containing problem datafiles offline using the temporary option, then follow the procedure in "Performing Datafile Recovery in an Open Database" on page 4-12. |

# Recovering Through an Added Datafile: Scenario

If database recovery with a backup control file rolls forward through a CREATE TABLESPACE or an ALTER TABLESPACE ADD DATAFILE operation, then Oracle stops recovery when applying the redo record for the added files and lets you confirm the filenames.

For example, suppose you make a whole database backup, and then later create a new tablespace containing two datafiles: `/oracle/dbs/db2.f` and `/oracle/dbs/db3.f`. If you later restore a backup control file and perform media recovery through the CREATE TABLESPACE operation, then Oracle may signal the following error when applying the CREATE TABLESPACE redo data:

```
ORA-00283: recovery session canceled due to errors
ORA-01244: unnamed datafile(s) added to controlfile by media recovery
ORA-01110: data file 3: '/oracle/dbs/db2.f'
ORA-01110: data file 2: '/oracle/dbs/db3.f'
```

**To recover through an ADD DATAFILE operation:**

1. View the files added by selecting from V$DATAFILE. For example:

```
SELECT FILE#,NAME
FROM V$DATAFILE;

FILE#          NAME
-------------- ---------------------
1              /oracle/dbs/db1.f
2              /oracle/dbs/UNNAMED00002
3              /oracle/dbs/UNNAMED00003
```

2. If multiple unnamed files exist, then determine which unnamed file corresponds to which datafile by using one of these methods:
   - Open the `alert_SID.log`, which contains messages about the original file location for each unnamed file.
   - Derive the original file location of each unnamed file from the error message and `V$DATAFILE`: each unnamed file corresponds to the file in the error message with the same file number.
3. Issue the `ALTER DATABASE RENAME FILE` statement to rename the datafiles. For example, enter:

```
ALTER DATABASE RENAME FILE '/db/UNNAMED00002' TO '/oracle/dbs/db3.f';
ALTER DATABASE RENAME FILE '/db/UNNAMED00003' TO '/oracle/dbs/db2.f';
```

4. Continue recovery by issuing the previous recovery statement. For example:

```
RECOVER AUTOMATIC DATABASE USING BACKUP CONTROLFILE UNTIL CANCEL
```

## Recovering Transportable Tablespaces: Scenario

The **transportable tablespace** feature of Oracle allows a user to transport a set of tablespaces from one database to another. Transporting a tablespace into a database is like creating a tablespace with preloaded data. Using this feature is often an advantage because:

- It is faster than using the Export or SQL*Loader utilities because it involves only copying datafiles and integrating metadata
- You can use it to move index data, hence avoiding the necessity of rebuilding indexes

Like normal tablespaces, transportable tablespaces are recoverable. While you can recover normal tablespaces without a backup, you must have a version of the transported datafiles in order to recover a transported tablespace.

**To recover a transportable tablespace:**

1. If the database is open, then take the transported tablespace offline. For example, if you want to recover the `users` tablespace, then issue:

```
ALTER TABLESPACE users OFFLINE IMMEDIATE;
```

2. Restore a backup of the transported datafiles using an operating system utility. The backup can be the initial version of the transported datafiles or any backup taken after the tablespace is transported. For example, enter:

```
% cp /backup/users.dbf /oracle/dbs/users.dbf
```

3. Recover the tablespace as normal. For example, enter:

```
RECOVER TABLESPACE users
```

Oracle may signal ORA-01244 when recovering through a transportable tablespace operation just as when recovering through a CREATE TABLESPACE operation. In this case, rename the unnamed files to the correct locations using the procedure in "Recovering Through an Added Datafile: Scenario" on page 6-3.

> **See Also:** *Oracle9i Database Administrator's Guide* for detailed information about using the transportable tablespace feature.

## Recovering After the Loss of Online Redo Log Files: Scenarios

If a media failure has affected the online redo logs of a database, then the appropriate recovery procedure depends on the following:

- The configuration of the online redo log: mirrored or non-mirrored
- The type of media failure: temporary or permanent
- The types of online redo log files affected by the media failure: current, active, unarchived, or inactive

Table 6–1 displays V$LOG status information that can be crucial in a recovery situation involving online redo logs.

***Table 6–1 STATUS Column of V$LOG***

| Status | Description |
|---|---|
| UNUSED | The online redo log has never been written to. |
| CURRENT | The log is active, that is, needed for instance recovery, and it is the log to which Oracle is currently writing. The redo log can be open or closed. |
| ACTIVE | The log is active, that is, needed for instance recovery, but is not the log to which Oracle is currently writing.It may be in use for block recovery, and may or may not be archived. |

*Table 6–1 STATUS Column of V$LOG*

| Status | Description |
|---|---|
| CLEARING | The log is being re-created as an empty log after an ALTER DATABASE CLEAR LOGFILE statement. After the log is cleared, then the status changes to UNUSED. |
| CLEARING_CURRENT | The current log is being cleared of a closed thread. The log can stay in this status if there is some failure in the switch such as an I/O error writing the new log header. |
| INACTIVE | The log is no longer needed for instance recovery. It may be in use for media recovery, and may or may not be archived. |

The following sections describe the appropriate recovery strategies for these situations:

- Recovering After Losing a Member of a Multiplexed Online Redo Log Group
- Recovering After the Loss of All Members of an Online Redo Log Group

## Recovering After Losing a Member of a Multiplexed Online Redo Log Group

If the online redo log of a database is multiplexed, and if at least one member of each online redo log group is not affected by the media failure, then Oracle allows the database to continue functioning as normal. Oracle writes error messages to the LGWR trace file and the `alert_SID.log` of the database.

Solve the problem by taking one of the following actions:

- If the hardware problem is temporary, then correct it. LGWR accesses the previously unavailable online redo log files as if the problem never existed.
- If the hardware problem is permanent, then drop the damaged member and add a new member by following the procedure below.

> **Note:** The newly added member provides no redundancy until the log group is reused.

**To replace a damaged member of a redo log group:**

1. Locate the filename of the damaged member in V$LOGFILE. The status is INVALID if the file is inaccessible:

```
SELECT GROUP#, STATUS, MEMBER
```

```
FROM V$LOGFILE
WHERE STATUS='INVALID';

GROUP#    STATUS       MEMBER
-------   -----------  --------------------
0002      INVALID       /oracle/dbs/log2b.f
```

2. Drop the damaged member. For example, to drop member `log2b.f` from group 2, issue:

```
ALTER DATABASE DROP LOGFILE MEMBER '/oracle/dbs/log2b.f';
```

3. Add a new member to the group. For example, to add `log2c.f` to group 2, issue:

```
ALTER DATABASE ADD LOGFILE MEMBER '/oracle/dbs/log2c.f' TO GROUP 2;
```

   If the file you want to add already exists, then it must be the same size as the other group members, and you must specify REUSE. For example:

```
ALTER DATABASE ADD LOGFILE MEMBER '/oracle/dbs/log2b.f' REUSE TO GROUP 2;
```

## Recovering After the Loss of All Members of an Online Redo Log Group

If a media failure damages all members of an online redo log group, then different scenarios can occur depending on the type of online redo log group affected by the failure and the archiving mode of the database.

If the damaged log group is inactive, then it is not needed for crash recovery; if it is active, then it is needed for crash recovery.

| If the group is . . . | Then . . . | And you should . . . |
|---|---|---|
| Inactive | It is not needed for crash recovery | Clear the archived or unarchived group. |
| Active | It is needed for crash recovery | Attempt to issue a checkpoint and clear the log; if impossible, then you must restore a backup and perform incomplete recovery up to the most recent available log. |
| Current | It is the log that Oracle is currently writing to | Attempt to clear the log; if impossible, then you must restore a backup and perform incomplete recovery up to the most recent available log. |

Your first task is to determine whether the damaged group is active or inactive.

**To determine whether the damaged groups are active:**

1. Locate the filename of the lost redo log in V$LOGFILE and then look for the group number corresponding to it. For example, enter:

```
SELECT GROUP#, STATUS, MEMBER FROM V$LOGFILE;

GROUP#    STATUS       MEMBER
-------   -----------  --------------------
0001                    /oracle/dbs/log1a.f
0001                    /oracle/dbs/log1b.f
0002      INVALID       /oracle/dbs/log2a.f
0002      INVALID       /oracle/dbs/log2b.f
0003                    /oracle/dbs/log3a.f
0003                    /oracle/dbs/log3b.f
```

2. Determine which groups are active. For example, enter:

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

GROUP#  MEMBERS          STATUS     ARCHIVED
------  -------          ---------  -----------
 0001   2                INACTIVE   YES
 0002   2                ACTIVE     NO
 0003   2                CURRENT    NO
```

3. If the affected group is inactive, follow the procedure in "Losing an Inactive Online Redo Log Group" on page 6-8. If the affected group is active (as in the preceding example), then follow the procedure in "Losing an Active Online Redo Log Group" on page 6-10.

## Losing an Inactive Online Redo Log Group

If all members of an online redo log group with INACTIVE status are damaged, then the procedure depends on whether you can fix the media problem that damaged the inactive redo log group.

| If the failure is . . . | Then . . . |
|---|---|
| Temporary | Fix the problem. LGWR can reuse the redo log group when required. |
| Permanent | The damaged inactive online redo log group eventually halts normal database operation. Reinitialize the damaged group manually by issuing the ALTER DATABASE CLEAR LOGFILE statement as described below. |

You can clear an active redo log group when the database is open or closed. The procedure depends on whether the damaged group has been archived.

**To clear an inactive, online redo log group that has been archived:**

1. If the database is shut down, then start a new instance and mount the database:

   ```
   STARTUP MOUNT
   ```

2. Reinitialize the damaged log group. For example, to clear redo log group 2, issue the following statement:

   ```
   ALTER DATABASE CLEAR LOGFILE GROUP 2;
   ```

**To clear an inactive, online redo log group that has not been archived:**

Clearing an unarchived log allows it to be reused without archiving it. This action makes 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 you need the cleared log file for recovery of a backup, then you cannot recover that backup. Also, it prevents complete recovery from backups due to the missing log.

1. If the database is shut down, then start a new instance and mount the database:

   ```
   STARTUP MOUNT
   ```

2. Clear the log using the UNARCHIVED keyword. For example, to clear log group 2, issue:

   ```
   ALTER DATABASE CLEAR LOGFILE UNARCHIVED GROUP 2;
   ```

   If there is an offline datafile that requires the cleared unarchived log to bring it online, then the keywords UNRECOVERABLE DATAFILE are required. The datafile and its entire tablespace have to be dropped because the redo necessary to bring it online is being cleared, and there is no copy of it. For example, enter:

   ```
   ALTER DATABASE CLEAR LOGFILE UNARCHIVED GROUP 2 UNRECOVERABLE DATAFILE;
   ```

3. Immediately back up the database with an operating system utility as described in "Making User-Managed Backups of the Whole Database" on page 2-4. Now you can use this backup for complete recovery without relying on the cleared log group. For example, enter:

   ```
   % cp /disk1/oracle/dbs/*.f /disk2/backup
   ```

4. Back up the database's control file using the ALTER DATABASE statement as described in "Backing Up the Control File to a Binary File" on page 2-19. For example, enter:

```
ALTER DATABASE BACKUP CONTROLFILE TO '/oracle/dbs/cf_backup.f';
```

**Failure of CLEAR LOGFILE Operation** The ALTER DATABASE CLEAR LOGFILE statement can fail with an I/O error due to media failure when it is not possible to:

- Relocate the redo log file onto alternative media by re-creating it under the currently configured redo log filename
- Reuse the currently configured log filename to re-create the redo log file because the name itself is invalid or unusable (for example, due to media failure)

In these cases, the ALTER DATABASE CLEAR LOGFILE statement (before receiving the I/O error) would have successfully informed the control file that the log was being cleared and did not require archiving. The I/O error occurred at the step in which the CLEAR LOGFILE statement attempts to create the new redo log file and write zeros to it. This fact is reflected in V$LOG.CLEARING_CURRENT.

## Losing an Active Online Redo Log Group

If the database is still running and the lost active log is *not* the current log, then issue the ALTER SYSTEM CHECKPOINT statement. If successful, then the active log is rendered inactive, and you can follow the procedure in "Losing an Inactive Online Redo Log Group" on page 6-8. If unsuccessful, or if your database has halted, then perform one of procedures in this section, depending on the archiving mode.

Note that the current log is the one LGWR is currently writing to. If a LGWR I/O fails, then LGWR terminates and the instance crashes. In this case, you must restore a backup, perform incomplete recovery, and open the database with the RESETLOGS option.

**To recover from loss of an active online redo log group in NOARCHIVELOG mode:**

1. If the media failure is temporary, then correct the problem so that Oracle can reuse the group when required.

2. Restore the database from a consistent, whole database backup (datafiles and control files) as described in "Restoring Datafiles" on page 3-6. For example, enter:

```
% cp /disk2/backup/*.f /disk1/oracle/dbs
```

3. Mount the database:

```
STARTUP MOUNT
```

4. Because online redo logs are not backed up, you cannot restore them with the datafiles and control files. In order to allow Oracle to reset the online redo logs, you must first mimic incomplete recovery:

```
RECOVER DATABASE UNTIL CANCEL
CANCEL
```

5. Open the database using the RESETLOGS option:

```
ALTER DATABASE OPEN RESETLOGS;
```

6. Shut down the database consistently. For example, enter:

```
SHUTDOWN IMMEDIATE
```

7. Make a whole database backup as described in "Making User-Managed Backups of the Whole Database" on page 2-4. For example, enter:

```
% cp /disk1/oracle/dbs/*.f /disk2/backup
```

**To recover from loss of an active online redo log group in ARCHIVELOG mode:**

If the media failure is temporary, then correct the problem so that Oracle can reuse the group when required. If the media failure is not temporary, then use the following procedure.

1. Begin incomplete media recovery. Use the procedure given in "Performing Incomplete User-Managed Media Recovery" on page 4-16, recovering up through the log before the damaged log.

2. Ensure that the current name of the lost redo log can be used for a newly created file. If not, then rename the members of the damaged online redo log group to a new location. For example, enter:

```
ALTER DATABASE RENAME FILE "/oracle/dbs/log_1.rdo" TO "/temp/log_1.rdo";
ALTER DATABASE RENAME FILE "/oracle/dbs/log_2.rdo" TO "/temp/log_2.rdo";
```

3. Open the database using the RESETLOGS option:

```
ALTER DATABASE OPEN RESETLOGS;
```

**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, then use the recovery method for the most difficult log to recover. The order of difficulty, from most difficult to least difficult, follows:

1. The current online redo log
2. An active online redo log
3. An unarchived online redo log
4. An inactive online redo log

## Recovering After the Loss of Archived Redo Log Files: Scenario

If the database is operating in ARCHIVELOG mode, and if the only copy of an archived redo log file is damaged, then the damaged file does not affect the present operation of the database. The following situations can arise, however, depending on when the redo log was written and when you backed up the datafile.

| If you backed up . . . | Then . . . |
|---|---|
| All datafiles 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. |
| A specific datafile before the filled online redo log group was written | If the corresponding datafile is damaged by a permanent media failure, use the most recent backup of the damaged datafile and perform incomplete recovery up to the damaged log. |

**Caution:** If you know that an archived redo log group has been damaged, immediately back up all datafiles so that you will have a whole database backup that does not require the damaged archived redo log.

## Recovering from User Errors: Scenario

An accidental 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 you have granted powerful privileges (such as DROP ANY TABLE) to only selected, appropriate users, you can minimize user errors that require database recovery.

**To recover a table that has been accidentally dropped:**

1. If possible, keep the database that experienced the user error online and available for use. Back up all datafiles of the existing database in case an error is made during the remaining steps of this procedure.
2. Restore a database backup to an alternative location, then perform incomplete recovery of this backup using a restored backup control file, to the point just before the table was dropped (as described in "Performing Incomplete User-Managed Media Recovery" on page 4-16).
3. Export the lost data from the temporary, restored version of the database using the Oracle utility Export. In this case, export the accidentally dropped table.

   > **Note:** System audit options are exported.

4. Use the Import utility to import the data back into the production database.
5. Delete the files of the temporary copy of the database to conserve space.

> **See Also:** *Oracle9i Database Utilities* for more information about the Import and Export utilities.

## Performing Media Recovery in a Distributed Environment: Scenario

The manner in which you perform media recovery depends on whether your database participates in a distributed database system. The Oracle distributed database architecture is autonomous. Therefore, depending on the type of recovery operation selected for a single, damaged database, you may have to coordinate recovery operations globally among all databases in the distributed system.

Table 6–2 summarizes different types of recovery operations and whether coordination among nodes of a distributed database system is required.

***Table 6–2 Recovery Operations in a Distributed Database Environment***

| If you are . . . | Then . . . |
|---|---|
| Restoring a whole backup for a database that was never accessed from a remote node | Use non-coordinated, autonomous database recovery. |
| Restoring a whole backup for a database that was accessed by a remote node for a database in `NOARCHIVELOG` mode | Shut down all databases and restore them using the same coordinated full backup. |
| Performing complete media recovery of one or more databases in a distributed database | Use non-coordinated, autonomous database recovery. |
| Performing incomplete media recovery of a database that was never accessed by a remote node | Use non-coordinated, autonomous database recovery. |
| Performing incomplete media recovery of a database that was accessed by a remote node | Use coordinated, incomplete recovery to the same global point in time for all databases in the distributed system. |

## Coordinating Time-Based and Change-Based Distributed Database Recovery

In special circumstances, one node in a distributed database may require recovery to a past 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 operation 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, as described in "Performing Time-Based Incomplete Recovery" on page 4-20. For example, if a database needs to be recovered because of a user error (such as an accidental table drop), then 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, search the `alert_`*`SID`*`.log` of the database for the `RESETLOGS` message.

   If the message is, "RESETLOGS after complete recovery through change xxx", then 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 message is, "RESETLOGS after incomplete recovery UNTIL CHANGE xxx", then 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.

# 7

# Performing User-Managed TSPITR

This chapter describes how to perform user-managed tablespace point-in-time recovery (TSPITR) with the transportable tablespace feature.

This chapter includes the following topics:

# Introduction to User-Managed Tablespace Point-in-Time Recovery

Tablespace point-in-time recovery (TSPITR) with the transportable tablespace feature enables you to quickly recover one or more tablespaces (other than the SYSTEM tablespace) to a time that is prior to the rest of the database.

User-managed TSPITR is most useful for recovering the following:

- An erroneous DROP TABLE or TRUNCATE TABLE operation
- An erroneous DROP TABLESPACE operation
- A table that is logically corrupted
- An incorrect batch job or other DML statement that has affected only a subset of the database
- A logical schema to a point different from the rest of the physical database when multiple schemas exist in separate tablespaces of one physical database
- A tablespace in a VLDB (very large database) when TSPITR is more efficient than restoring the whole database from a backup and rolling it forward (refer to "Preparing for Tablespace Point-in-Time Recovery: Basic Steps" on page 7-4 before making any decisions)

## TSPITR Terminology

Familiarize yourself with the following terms and abbreviations, which are used throughout this chapter:

**TSPITR**
Tablespace point-in-time recovery

**Primary Database**
The database containing the tablespace or tablespaces that you want to recover to a prior point in time.

**Auxiliary Database**
A copy of the current database that is restored from a backup. It includes restored backups of the following:

- Datafiles belonging to the SYSTEM tablespace
- Datafiles in the set of tablespaces to be recovered
- Datafiles belonging to a system managed undo tablespace (when you run the database in automatic undo management mode) or tablespace that contains

rollback segments (when you run the database in manual undo management mode)

All backups must be from a point in time prior to the desired recovery time.

**Recovery Set**
All the tablespaces that require point-in-time recovery to be performed on them.

**Recovery Set Self-Containment Check**
All objects that are part of the recovery set must be self-contained: there can be no dependencies on objects outside the recovery set. For example, if a table is part of the recovery set and its indexes are in a separate tablespace, then the recovery set must include the tablespace containing the index. Alternatively, the index can be dropped. The recovery set tablespaces can be checked for self-containment with the procedure `DBMS_TTS.TRANSPORT_SET_CHECK`.

**Auxiliary Set**
Any other items required for restoring the auxiliary database, including:

- Backup control file
- Datafiles from the `SYSTEM` tablespace
- Datafiles in an undo tablespace or datafiles containing rollback segments

**Transportable Tablespace**
A rapid method of transporting tablespaces across databases by unplugging them from a source database and plugging them into a target database. The unplugging and plugging is done with the Export and Import utilities. Note that there is no actual export and import of the table data, but simply an export and import of internal metadata. During the procedure, the datafiles of the transported tablespaces are made part of the target database.

## TSPITR Methods

In releases prior to Oracle9*i*, you had the following two methods for performing user-managed TSPITR:

- Traditional user-managed TSPITR, which required you to create a special type of database called a **clone database**
- User-managed TSPITR with the transportable tablespace feature

Oracle9*i* TSPITR should be performed by using the transportable tablespace feature. This procedure is relatively easy to use and is less error prone than the traditional method, which is currently deprecated (although not yet unsupported).

Conceptually, TSPITR is performed by dropping the tablespaces to be recovered from the primary database, restoring a copy of the database called an **auxiliary database** and recovering it to the desired point in time, then transporting the relevant tablespaces from the auxiliary database to the current version of the primary database.

For ease of use, it is highly recommended that you place the auxiliary and primary databases on different hosts. Nevertheless, you can also perform TSPITR when the databases are located on the same host.

The basic procedure for performing user-managed TSPITR is as follows:

1. Take the tablespaces requiring TSPITR offline
2. Plan the setup of the auxiliary database.
3. Create the auxiliary database and recover it to the desired point in time.
4. Drop the tablespaces requiring TSPITR from the primary database.
5. Use the transportable tablespace feature to transport the set of tablespaces from the auxiliary database to the primary database.

> **See Also:** *Oracle9i Database Administrator's Guide* for a complete account of how to use the transportable tablespace feature

## Preparing for Tablespace Point-in-Time Recovery: Basic Steps

TSPITR requires careful planning. Before proceeding you should read this chapter thoroughly.

This section contains the following topics:

**Caution:** You should not perform TSPITR for the first time on a production system, or when there is a time constraint.

## Step 1: Review TSPITR Requirements

Satisfy the following requirements before performing TSPITR:

- Ensure that you have backups of all datafiles in the recovery and auxiliary set tablespaces. The datafile backups must have been created *before* the desired TSPITR time.
- Ensure that you have a control file backup that is usable on the auxiliary database. To be usable, the control file must meet these requirements:
  - The control file must have been backed up *before* the desired TSPITR time.
  - The control file must have been backed up with the following SQL statement, where *cf_name* refers to the fully specified filename:

    ```
    ALTER DATABASE BACKUP CONTROLFILE TO 'cf_name';
    ```

- Ensure that all files constituting the recovery set tablespaces are in the recovery set on the auxiliary database; otherwise, the export phase during tablespace transport fails.
- Allocate enough disk space on the auxiliary host to accommodate the auxiliary database.
- Provide enough real memory to start the auxiliary instance.

**See Also:** "Step 6: Create the Initialization Parameter File for the Auxiliary Instance" on page 7-7

## Step 2: Identify All of the Files in the Recovery and Auxiliary Set Tablespaces

Before you create the auxiliary database, make sure that you connect to the primary database with administrator privileges and obtain all of the following information about the primary database:

- The filenames of the datafiles in the recovery set tablespaces
- The filenames of the datafiles in the SYSTEM tablespace
- The filenames of the datafiles in an undo tablespace or datafiles containing rollback segments

- The filenames of the control files

The following useful query displays the filenames of all datafiles, control files, and online redo logs in the database:

```
SELECT NAME FROM V$DATAFILE
UNION ALL
SELECT MEMBER FROM V$LOGFILE
UNION ALL
SELECT NAME FROM V$CONTROLFILE;
```

To determine the filenames of the datafiles in the SYSTEM and recovery set tablespaces, execute the following query and replace *RECO_TBS_1*, *RECO_TBS_2*, and so forth with the names of the recovery set tablespaces:

```
SELECT t.NAME AS "reco_tbs", d.NAME AS "dbf_name"
FROM V$DATAFILE d, V$TABLESPACE t
WHERE t.TS# = d.TS#
AND t.NAME IN ('SYSTEM', 'RECO_TBS_1', 'RECO_TBS_2');
```

If you run the database in manual undo management mode, then the following query displays the names of the tablespaces containing rollback segments as well as the names of the datafiles in the tablespaces:

```
SELECT r.TABLESPACE_NAME AS "rbs_tbs", d.FILE_NAME AS "dbf_name"
FROM DBA_ROLLBACK_SEGS r, DBA_DATA_FILES d
WHERE r.TABLESPACE_NAME=d.TABLESPACE_NAME;
```

If you run the database in automatic undo management mode, then the following query displays the names of the undo tablespaces as well as the names of the datafiles in the tablespaces:

```
SELECT u.TABLESPACE_NAME AS "undo_tbs", d.FILE_NAME AS "dbf_name"
FROM DBA_UNDO_EXTENTS u, DBA_DATA_FILES d
WHERE u.TABLESPACE_NAME=d.TABLESPACE_NAME;
```

## Step 3: Determine Whether Objects Will Be Lost

When TSPITR is performed on a tablespace, any objects created after the recovery time are lost. To determine 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 7–1.

***Table 7–1 TS_PITR_OBJECTS_TO_BE_DROPPED View***

| Column Name | Meaning |
|---|---|
| OWNER | Owner of the object to be dropped. |

*Table 7–1 TS_PITR_OBJECTS_TO_BE_DROPPED View*

| Column Name | Meaning |
|---|---|
| NAME | The name of the object that will be lost as a result of TSPITR |
| CREATION_TIME | Creation time stamp for the object. |
| TABLESPACE_NAME | Name of the tablespace containing the object. |

When querying this view, supply all the elements of the date field, otherwise the default setting is used. Also, use the TO_CHAR and TO_DATE functions. For example, with a recovery set consisting of sales_1 and sales_2, and a recovery point in time of '2000-06-02:07:03:11', execute the following SQL script:

```
SELECT OWNER, NAME, TABLESPACE_NAME, TO_CHAR(CREATION_TIME, 'YYYY-MM-DD:HH24:MI:SS')
FROM SYS.TS_PITR_OBJECTS_TO_BE_DROPPED
WHERE TABLESPACE_NAME IN ('SALES_1','SALES_2')
AND CREATION_TIME > TO_DATE('00-JUN-02:07:03:11','YY-MON-DD:HH24:MI:SS')
ORDER BY TABLESPACE_NAME, CREATION_TIME;
```

> **See Also:** *Oracle9i Database Reference* for more information about the TS_PITR_OBJECTS_TO_BE_DROPPED view

## Step 4: Choose a Method for Connecting to the Auxiliary Instance

You must be able to connect to the auxiliary instance. You can either use Oracle Net or operating system authentication. To learn how to configure networking files, refer to *Oracle Net Services Administrator's Guide.*

## Step 5: Create an Oracle Password File for the Auxiliary Instance

For information about creating and maintaining Oracle password files, refer to the *Oracle9i Database Administrator's Guide.* If you do not use a password file, then you can skip this step.

## Step 6: Create the Initialization Parameter File for the Auxiliary Instance

Create a new initialization parameter file rather than copying and then editing the production database initialization parameter file. Save memory by using low settings for parameters such as the following:

- DB_CACHE_SIZE
- SHARED_POOL_SIZE

- LARGE_POOL_SIZE

Note that reducing the preceding parameter settings can prevent the auxiliary database from starting when other dependent parameters are set too high—for example, the initialization parameter ENQUEUE_RESOURCES, which allocates memory from within the shared pool.

The auxiliary database can be either on the same host as the primary database or on a different host. Because the auxiliary database filenames are identical to the primary database filenames in the auxiliary control file, you must rename them in this control file so that they point to the restored locations. If the auxiliary database is on the same machine as the primary database, or if the auxiliary database is on a different machine that uses different path names, then you must rename the control files, datafiles, and online redo logs. If the auxiliary database is on a different machine with the same path names, then you can rename just the online redo logs.

> **Caution:** If the auxiliary and primary database are on the same machine, then failing to rename the online redo log files may cause primary database corruption.

Set the parameters shown in Table 7–2 in the auxiliary initialization parameter file.

***Table 7–2 Auxiliary Initialization Parameters***

| Parameter | Purpose |
|---|---|
| DB_NAME | Names the auxiliary database. Leave the name of the auxiliary database *the same as* the primary database. |
| CONTROL_FILES | Identifies auxiliary control files. Set to the filename of the auxiliary control file. Make sure the control file name is *different from* the primary database control file name. |
| LOCK_NAME_SPACE | Allows the auxiliary database to start even though it has the same name as the primary database. Set to any unique value, for example, = AUX. This parameter is only needed if the auxiliary and primary database are on the same host. |
| DB_FILE_NAME_CONVERT | Uses patterns to convert filenames for the datafiles of the auxiliary database. This parameter is only necessary if you are either restoring the auxiliary database on the same host as the primary host, or on a different host that uses different path names from the primary host. |
| LOG_FILE_NAME_CONVERT | Uses patterns to convert filenames for the online redo logs of the auxiliary database. This parameter is mandatory. |

***Table 7–2 Auxiliary Initialization Parameters***

| Parameter | Purpose |
|---|---|
| LOG_ARCHIVE_DEST_1 | Specifies the default directory containing the archived redo logs required for recovery.This parameter specifies the location on the auxiliary host in which the archived logs will be located. |
| LOG_ARCHIVE_FORMAT | Specifies the format of the archived logs. You should use the same format setting used in the primary initialization parameter file. |

Set other parameters as needed, including the parameters that allow you to connect as SYSDBA through Oracle Net.

For example, the auxiliary parameter file for a database on the same host as the primary could look like the following:

```
DB_NAME = prod1
CONTROL_FILES = /oracle/aux/cf1.f
LOCK_NAME_SPACE = aux
DB_FILE_NAME_CONVERT=("/oracle/dbs/","/oracle/aux/")
LOG_FILE_NAME_CONVERT=("/oracle/dbs/","/oracle/aux/")
LOG_ARCHIVE_DEST_1 = 'LOCATION=/oracle/work/arc_dest/arc'
LOG_ARCHIVE_FORMAT = r_%t_%s.arc
```

The auxiliary parameter file for a database on a different host with the same path names as the primary could look like the following:

```
DB_NAME = prod1
CONTROL_FILES = /oracle/aux/cf1.f
LOG_FILE_NAME_CONVERT=("/oracle/dbs/","/oracle/aux/")
LOG_ARCHIVE_DEST_1 = 'LOCATION=/oracle/work/arc_dest/arc'
LOG_ARCHIVE_FORMAT = r_%t_%s.arc
```

## Restoring and Recovering the Auxiliary Database: Basic Steps

The procedure for restore and recovery of the auxiliary database differs depending on whether the auxiliary database is on the same host as the primary database. The examples in this section assume:

- You are performing TSPITR on production database called `prod1` located on host `prim_host`.

- The recovery set tablespaces are `sales_1` and `sales_2`. Tablespace `sales_1` contains datafile `/oracle/dbs/sales_1.f` and tablespace `sales_2` contains datafile `/fs2/sales_2.f`.
- The auxiliary set contains the `SYSTEM` tablespace datafile `/oracle/dbs/system.f`, the undo tablespace datafile `/oracle/dbs/undo.f`, and the control file `/oracle/dbs/cf1.f`.
- The online redo logs are named `/oracle/dbs/log1.f` and `/oracle/dbs/log2.f`.
- All the primary database files are contained in `/oracle/dbs`

The different cases are described in the following sections:

- Restoring and Recovering the Auxiliary Database on the Same Host
- Restoring and Recovering the Auxiliary Database on a Different Host with the Same Path Names
- Restoring and Recovering the Auxiliary Database on a Different Host with Different Path Names

## Restoring and Recovering the Auxiliary Database on the Same Host

The following examples assume the case in which you restore the auxiliary database to the same host as the primary database. In this scenario, all of the primary database files are contained in `/oracle/dbs`, and you want to restore the auxiliary database to `/oracle/dbs/aux`. So, you set `DB_FILE_NAME_CONVERT` and `LOG_FILE_NAME_CONVERT` to convert the filenames from `/oracle/dbs` to `/oracle/dbs/aux`.

**Perform the following tasks to restore and recover the auxiliary database:**

1. Restore the auxiliary set and the recovery set to a location different from that of the primary database. For example, assume that the auxiliary set consists of the following files:

```
/oracle/dbs/cf1.f      # control file
/oracle/dbs/undo.f     # datafile in undo tablespace
/oracle/dbs/system.f   # datafile in SYSTEM tablespace
```

   And the recovery set consists of the following datafiles:

```
/oracle/dbs/sales_1.f  # datafile in sales_1 tablespace
/oracle/dbs/sales_2.f  # datafile in sales_2 tablespace
```

You can restore backups of the auxiliary set files and recovery set files to a new location as follows:

```
cp /backup/cf1.f /aux/cf1.f
cp /backup/undo.f /aux/undo.f
cp /backup/system.f /aux/system.f
cp /backup/sales_1.f /aux/sales_1.f
cp /backup/sales_2.f /aux/sales_2.f
```

2. Start the auxiliary database without mounting it, specifying the initialization parameter file if necessary. For example, enter:

```
STARTUP NOMOUNT PFILE=/aux/initAUX.ora
```

3. Mount the auxiliary database, specifying the CLONE keyword:

```
ALTER DATABASE MOUNT CLONE DATABASE;
```

   The CLONE keyword causes Oracle to take all datafiles offline automatically.

4. Manually rename all auxiliary database files to reflect their new locations *only if* these files are not renamed by DB_FILE_NAME_CONVERT and LOG_FILE_NAME_CONVERT. In our scenario, all datafiles and online redo logs are renamed by initialization parameters, so no manual renaming is necessary.

5. Run the following SQL script to ensure that all datafiles are named correctly:

```
SELECT NAME FROM V$DATAFILE
UNION ALL
SELECT MEMBER FROM V$LOGFILE
UNION ALL
SELECT NAME FROM V$CONTROLFILE
/
```

   If not, then rename the files manually as in the previous step.

6. Bring only the datafiles in the auxiliary and recovery set tablespaces online. For example, bring the four datafiles in the recovery and auxiliary sets online:

```
ALTER DATABASE DATAFILE /oracle/dbs/aux/system.f ONLINE;
ALTER DATABASE DATAFILE /oracle/dbs/aux/sales_1.f ONLINE;
ALTER DATABASE DATAFILE /oracle/dbs/aux/sales_2.f ONLINE;
ALTER DATABASE DATAFILE /oracle/dbs/aux/undo.f ONLINE;
```

---

**Note:** The export phase of TSPITR will not work if all the files of each recovery set tablespace are not online.

---

At this point, the auxiliary database is mounted and ready for media recovery.

7. Recover the auxiliary database to the specified point in time with the USING BACKUP CONTROLFILE option. Use any form of incomplete recovery as described in "Performing Incomplete User-Managed Media Recovery" on page 4-16. The following example uses cancel-based incomplete recovery:

```
RECOVER DATABASE UNTIL CANCEL USING BACKUP CONTROLFILE
```

8. Open the auxiliary database with the RESETLOGS option using the following statement:

```
ALTER DATABASE OPEN RESETLOGS;
```

## Restoring and Recovering the Auxiliary Database on a Different Host with the Same Path Names

The following example assumes that you create the auxiliary database on a different host called aux_host. The auxiliary host has the same path names as the primary host. Hence, you do not need to rename the auxiliary database datafiles. So, you do *not* need to set DB_FILE_NAME_CONVERT, although you should set LOG_FILE_NAME_CONVERT.

**To restore and recover the auxiliary database:**

1. Restore the auxiliary set and the recovery set to the auxiliary host. For example, assume that the auxiliary set consists of the following files:

```
/oracle/dbs/cf1.f      # control file
/oracle/dbs/undo.f     # datafile in undo tablespace
/oracle/dbs/system.f   # datafile in SYSTEM tablespace
```

   And the recovery set consists of the following datafiles:

```
/oracle/dbs/sales_1.f  # 1st datafile in sales_1 tablespace
/oracle/dbs/sales_2.f  # 2nd datafile in sales_2 tablespace
```

   These files will occupy the same locations in the auxiliary host.

2. Start the auxiliary database without mounting it, specifying the initialization parameter file if necessary. For example, enter:

```
STARTUP NOMOUNT PFILE=/aux/initAUX.ora
```

3. Mount the auxiliary database, specifying the CLONE keyword:

```
ALTER DATABASE MOUNT CLONE DATABASE;
```

The CLONE keyword causes Oracle to take all datafiles offline automatically.

4. Rename all auxiliary database files to reflect their new locations *only if* these files are not renamed by DB_FILE_NAME_CONVERT and LOG_FILE_NAME_CONVERT. In our scenario, the datafiles do not require renaming, and the logs are converted with LOG_FILE_NAME_CONVERT. So, no manual renaming is necessary.

5. Run the following script in SQL*Plus to ensure that all datafiles are named correctly.

```
SELECT NAME FROM V$DATAFILE
UNION ALL
SELECT MEMBER FROM V$LOGFILE
UNION ALL
SELECT NAME FROM V$CONTROLFILE
/
```

If not, then rename them manually as in the previous step.

6. Bring all datafiles in the auxiliary and recovery set tablespaces online. For example, bring the four datafiles in the recovery and auxiliary sets online:

```
ALTER DATABASE DATAFILE /oracle/dbs/system.f ONLINE;
ALTER DATABASE DATAFILE /oracle/dbs/sales_1.f ONLINE;
ALTER DATABASE DATAFILE /oracle/dbs/sales_2.f ONLINE;
ALTER DATABASE DATAFILE /oracle/dbs/undo.f ONLINE;
```

> **Note:** The export phase of TSPITR will not work if all the files of each recovery set tablespace are not online.

At this point, the auxiliary database is mounted and ready for media recovery.

7. Recover the auxiliary database to the specified point in time with the USING BACKUP CONTROLFILE option. Use any form of incomplete recovery as described in "Performing Incomplete User-Managed Media Recovery" on page 4-16. The following example uses cancel-based incomplete recovery:

```
RECOVER DATABASE UNTIL CANCEL USING BACKUP CONTROLFILE
```

8. Open the auxiliary database with the RESETLOGS option using the following statement:

```
ALTER DATABASE OPEN RESETLOGS;
```

### Restoring and Recovering the Auxiliary Database on a Different Host with Different Path Names

This case should be treated exactly like "Restoring and Recovering the Auxiliary Database on the Same Host" on page 7-10. The same guidelines for renaming files apply in both cases.

## Performing TSPITR with Transportable Tablespaces

After you have completed the preparation stage, begin the actual TSPITR procedure as described in *Oracle9i Database Administrator's Guide*. The procedure occurs in the following steps:

- Step 1: Unplugging the Tablespaces from the Auxiliary Database
- Step 2: Transporting the Tablespaces into the Primary Database

### Step 1: Unplugging the Tablespaces from the Auxiliary Database

In this step, you recover the auxiliary database to the desired noncurrent time, then unplug the desired tablespaces.

**To unplug the auxiliary database tablespaces:**

1. Make the tablespaces in the recovery set read-only by running the `ALTER TABLESPACE ... READ ONLY` statement. For example, make `sales_1` and `sales_2` read-only as follows:

   ```
   ALTER TABLESPACE sales_1 READ ONLY;
   ALTER TABLESPACE sales_2 READ ONLY;
   ```

2. Ensure that the recovery set is self-contained. For example:

   ```
   EXECUTE SYS.DBMS_TTS.TRANSPORT_SET_CHECK('sales_1,sales_2',TRUE,TRUE);
   ```

3. Query the transportable tablespace violations table to manage any dependencies. For example:

   ```
   SELECT * FROM SYS.TRANSPORT_SET_VIOLATIONS;
   ```

   This query should return no rows after all dependencies are managed. Refer to *Oracle9i Database Administrator's Guide* for more information about this table.

4. Generate the transportable set by running the Export utility as described in *Oracle9i Database Administrator's Guide*. Include all tablespaces in the recovery set, as in the following example:

```
% exp SYS/pwd TRANSPORT_TABLESPACE=y TABLESPACES=(sales_1,sales_2) TTS_FULL_CHECK=y
```

This command generates an export file named `expdat.dmp`.

## Step 2: Transporting the Tablespaces into the Primary Database

In this step, you transport the recovery set tablespaces into the primary database.

**To plug the recovery set tablespaces into the primary database:**

1. In the primary database (*not* the auxiliary database), drop the tablespaces in the recovery set through the DROP TABLESPACE statement. For example:

```
DROP TABLESPACE sales_1 INCLUDING CONTENTS;
DROP TABLESPACE sales_2 INCLUDING CONTENTS;
```

2. Restore the recovery set datafiles from the auxiliary database to the recovery set file locations in the primary database. For example:

```
% cp /net/aux_host/aux/sales_1.f /net/primary_host/oracle/dbs/sales_1.f
% cp /net/aux_host/aux/sales_2.f /net/primary_host/oracle/dbs/sales_2.f
```

3. Move the export file `expdat.dmp` to the primary host. For example, enter:

```
% cp /net/aux_host/aux/expdat.dmp /net/primary_host/oracle/dbs/expdat.dmp
```

4. Plug in the transportable set into the primary database by running Import as described in *Oracle9i Database Administrator's Guide*. For example:

```
% imp TRANSPORT_TABLESPACE=y FILE=expat.dmp
   DATAFILES='/oracle/dbs/sales_1.f','/oracle/dbs/sales_2.f')
```

5. Make the recovered tablespaces read write by issuing the ALTER TABLESPACE READ WRITE statement. For example:

```
ALTER TABLESPACE sales_1 READ WRITE;
ALTER TABLESPACE sales_2 READ WRITE;
```

6. Back up the recovered tablespaces with an operating system utility as described in "Making User-Managed Backups of Online Tablespaces and Datafiles" on page 2-7.

---

**Caution:** You must back up the tablespace because otherwise you might lose it. For example, a media failure occurs, but the archived logs from the last backup of the database do not logically link to the recovered tablespaces. If you attempt to recover any recovery set tablespaces from a backup taken before TSPITR, then recovery fails.

---

## Performing Partial TSPITR of Partitioned Tables

Partitioned tables can span multiple tablespaces. Follow this procedure only if the recovery set does not fully contain all of the partitions.

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 Standalone Tables
- Step 4: Drop the Recovery Set Tablespace
- Step 5: Create Tables at Auxiliary Database
- Step 6: Drop Indexes on Partitions Being Recovered
- Step 7: Exchange Partitions with Standalone Tables on the Auxiliary Database
- Step 8: Transport the Recovery Set Tablespaces
- Step 9: Exchange Partitions with Standalone Tables on the Primary Database
- Step 10: Back Up the Recovered Tablespaces in the Primary Database

---

**Note:** Often you have to recover the dropped partition along with recovering a partition whose range has expanded. Refer to "Performing TSPITR of Partitioned Tables When a Partition Has Been Dropped" on page 7-19.

---

## Step 1: Create a Table on the Primary Database for Each Partition Being Recovered

This table should have the exact same column names and column datatypes as the partitioned table you are recovering. Create the table using the following template:

```
CREATE TABLE new_table AS
  SELECT * FROM partitioned_table
  WHERE 1=2;
```

These tables are used to swap each recovery set partition (see "Step 3: Exchange Partitions with Standalone Tables" on page 7-17).

**Note:** The table and the partition must belong to the same schema.

## 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. If you drop the indexes on the partition being recovered, then you need to drop them on the auxiliary database (see "Step 6: Drop Indexes on Partitions Being Recovered" on page 7-18). Rebuild the indexes after TSPITR is complete.

## Step 3: Exchange Partitions with Standalone Tables

Exchange each partition in the recovery set with its associated standalone table (created in Step 1) by issuing the following statement, replacing the variables with the names of the appropriate objects:

```
ALTER TABLE table_name EXCHANGE PARTITION partition_name WITH TABLE table_name;
```

## Step 4: Drop the Recovery Set Tablespace

On the primary database, drop each tablespace in the recovery set. For example, enter the following, replacing *tablespace_name* with the name of the tablespace:

```
DROP TABLESPACE tablespace_name INCLUDING CONTENTS;
```

## Step 5: Create Tables at Auxiliary Database

After recovering the auxiliary database and opening it with the RESETLOGS option, create a table in the SYSTEM tablespace that has the same column names and column data types as the partitioned table you are recovering. You must create the table in the SYSTEM tablespace: otherwise, Oracle issues the ORA-01552 error.

Create a table for each partition you wish to recover. These tables are used later to swap each recovery set partition.

**Note:** The table and the partition must belong to the same schema.

## 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 Standalone Tables on the Auxiliary Database

For each partition in the auxiliary database recovery set, exchange the partitions with the standalone tables (created in Step 5) by executing the following SQL script, replacing the variables with the appropriate object names:

```
ALTER TABLE partitioned_table_name
EXCHANGE PARTITION partition_name
WITH TABLE table_name;
```

## Step 8: Transport the Recovery Set Tablespaces

Export the recovery set tablespaces from the auxiliary database and then import them into the primary database as described in "Performing TSPITR with Transportable Tablespaces" on page 7-14.

## Step 9: Exchange Partitions with Standalone Tables on the Primary Database

For each recovered partition on the primary database, swap its associated standalone table using the following statement, replacing the variables with the appropriate object names:

```
ALTER TABLE table_name EXCHANGE PARTITION partition_name WITH TABLE table_name;
```

If the associated indexes have been dropped, then re-create them.

## Step 10: Back Up the Recovered Tablespaces in the Primary Database

Back up the recovered tablespaces on the primary database. Failure to do so results in loss of data in the event of media failure.

# 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

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, run the following SQL script at the primary database, replacing the variables with the appropriate values:

```
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, then create a temporary table in which to store these records so that if necessary they can be inserted into the recovered partition later.

## Step 3: Delete Records From the Partitioned Table

Delete all the records stored in the temporary table from the partitioned table.

## Step 4: Drop the Recovery Set Tablespace

On the primary database, drop each tablespace in the recovery set. For example, enter the following, replacing *tablespace_name* with the name of the tablespace:

```
DROP TABLESPACE tablespace_name INCLUDING CONTENTS;
```

## Step 5: Create Tables at the Auxiliary Database

After opening the auxiliary database with the RESETLOGS option, create a table in the SYSTEM tablespace that has the same column names and column data types as the partitioned table you are recovering. You must create the table in the SYSTEM tablespace: otherwise, Oracle issues the ORA-01552 error. Create a table for each partition that you want 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, nonpartitioned indexes that exist on the partition you wish to recover.

## Step 7: Exchange Partitions with Standalone Tables

For each partition in the auxiliary recovery set, exchange the partitions into the standalone tables created in Step 5 by issuing the following statement, replacing the variables with the appropriate values:

```
ALTER TABLE partitioned_table_name
EXCHANGE PARTITION partition_name
WITH TABLE table_name;
```

## Step 8: Transport the Recovery Set Tablespaces

Export the recovery set tablespaces from the auxiliary database and then import them into the primary database as described in "Performing TSPITR with Transportable Tablespaces" on page 7-14.

## Step 9: Insert Standalone Tables into Partitioned Tables

At this point you must insert the standalone tables into the partitioned tables; you can do this by first issuing the following statement, replacing the variables with the appropriate values:

```
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 standalone table into the partition, replacing the variables with the appropriate values:

```
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, then add it with the ALTER TABLE ADD PARTITION statement.

---

### Step 10: Back Up the Recovered Tablespaces in the Primary Database

Back up the recovered tablespaces in the primary database. Failure to do so results in loss of data in the event of media failure.

## Performing TSPITR of Partitioned Tables When a Partition Has 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
- Steps 2: Follow Same Procedure as for Partial TSPITR of Partitioned Tablespaces

### Step 1: Drop the Lower of the Two Partitions 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 expands downwards. In other words, the higher partition has the same range as before the split. For example, if P1 was split into 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. For example, execute the following, replacing the variables with the appropriate values:

```
CREATE TABLE new_table
AS SELECT * FROM partitioned_table
WHERE 1=2;
```

These tables will be used to exchange each recovery set partition in Step 3.

## Steps 2: Follow Same Procedure as for Partial TSPITR of Partitioned Tablespaces

Follow the same procedure as for "Performing Partial TSPITR of Partitioned Tables" on page 7-16, but skip the first step of this procedure. In other words, start with "Step 2: Drop the Indexes on the Partition Being Recovered" and follow all subsequent steps.

# Index

## A



## B

## C



## D

## E



## F



## G



## H



## I



## L



## M

## N

## O



## P



## R

## S



## T

## U



## V

## W