Oracle® Database

Backup and Recovery Basics

10*g* Release 1 (10.1)

Part No. B10735-01

December 2003

Oracle Database Backup and Recovery Basics 10*g* Release 1 (10.1)

Part No. B10735-01

Primary Author: Antonio Romero

Contributing Author: Lance Ashdown

Contributors: Anand Beldalker, Tammy Bednar, Don Beusee, Senad Dizdar, Wei Hu, Donna Keesling, Bill Lee, Muthu Olagappan, Francisco Sanchez, Vinay Srihari, Steve Wertheimer

Graphic Artist: Valarie Moore

Contents

Send Us Your Comments

Oracle Database Backup and Recovery Basics 10*g* Release 1 (10.1)

Part No. B10735-01

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 guide provides a basic conceptual overview of Oracle database backup and recovery.

This preface contains these topics:

Audience

This manual is intended for database administrators who perform backup and recovery of an Oracle database server.

To use this document, you need to know the following:

- Relational database concepts and basic database administration as described in *Oracle Database Concepts* and *Oracle Database Administrator's Guide*
- The operating system environment under which you are running the Oracle database

Organization

This document contains:

Chapter 1, "Backup and Recovery Overview"

This chapter briefly introduces the basic concepts of Oracle database backup and recovery.

Chapter 2, "Backup and Recovery Strategies"

This chapter gives general recommendations for a backup and recovery strategy.

Chapter 3, "Setting Up and Configuring Backup and Recovery"

This chapter describes how to prepare RMAN for initial use.

Chapter 4, "Making Backups with Recovery Manager"

This chapter describes how to use the RMAN `BACKUP` command.

Chapter 5, "Performing Recovery"

This chapter describes how to use the RMAN `RESTORE` and `RECOVER` commands.

Chapter 6, "Recovery Manager Maintenance Tasks"

This chapter describes how to maintain the RMAN backup metadata, which is stored in the control file of the target database.

"Glossary"

This chapter defines common backup and recovery terms.

Related Documentation

For more information, see these Oracle resources:

- *Oracle Database Recovery Manager Quick Start Guide*
- *Oracle Database Backup and Recovery Advanced User's Guide*
- *Oracle Database Recovery Manager Reference*
- *Oracle Database SQL Reference*
- *Oracle Database Utilities*

Many of the examples in this book use the sample schemas of the seed database, which is installed by default when you install Oracle. Refer to *Oracle Database Sample Schemas* for information on how these schemas were created and how you can use them yourself.

Printed documentation is available for sale in the Oracle Store at

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

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://otn.oracle.com/membership/
```

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://otn.oracle.tcom/documentation/
```

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.	When you specify this clause, you create an **index-organized table**.
Italics	Italic typeface indicates book titles or emphasis.	*Oracle Database Concepts* 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, usernames, and roles.	You can specify this clause only for a `NUMBER` column. You can back up the database by using the `BACKUP` command. Query the `TABLE_NAME` column in the `USER_TABLES` data dictionary view. Use the `DBMS_STATS.GENERATE_STATS` procedure.
`lowercase monospace (fixed-width) font`	Lowercase monospace typeface indicates executables, filenames, directory names, 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, usernames and roles, program units, and parameter values. **Note:** Some programmatic elements use a mixture of UPPERCASE and lowercase. Enter these elements as shown.	Enter `sqlplus` to open SQL*Plus. The password is specified in the `orapwd` file. Back up the datafiles and control files in the `/disk1/oracle/dbs` directory. The `department_id`, `department_name`, and `location_id` columns are in the `hr.departments` table. Set the `QUERY_REWRITE_ENABLED` initialization parameter to `true.` Connect as `oe` user. The `JRepUtil` class implements these methods.
`lowercase italic monospace (fixed-width) font`	Lowercase italic monospace font represents placeholders or variables.	You can specify the *`parallel_clause`*. Run `U`*`old_release`*`.SQL` where *`old_release`* refers to the release you installed prior to upgrading.

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} [COMPRESS | NOCOMPRESS]

...	Horizontal ellipsis points indicate either:
	■ That we have omitted parts of the code that are not directly related to the example
	■ That you can repeat a portion of the code
		CREATE TABLE ... AS *subquery*;
		
		SELECT *col1*, *col2*, ... , *coln* FROM employees;

. . .	Vertical ellipsis points indicate that we have omitted several lines of code not directly related to the example.	SQL> SELECT NAME FROM V$DATAFILE; NAME ------------------------------------ /fsl/dbs/tbs_01.dbf /fs1/dbs/tbs_02.dbf . . . /fsl/dbs/tbs_09.dbf 9 rows selected.

Other notation	You must enter symbols other than brackets, braces, vertical bars, and ellipsis points as shown.	acctbal NUMBER(11,2);
		acct CONSTANT NUMBER(4) := 3;

Italics	Italicized text indicates placeholders or variables for which you must supply particular values.	CONNECT SYSTEM/*system_password* DB_NAME = *database_name*

Convention	Meaning	Example

`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;`
		`SELECT * FROM USER_TABLES;`
		`DROP TABLE hr.employees;`

`lowercase`	Lowercase typeface indicates programmatic elements that you supply. For example, lowercase indicates names of tables, columns, or files.
	Note: Some programmatic elements use a mixture of UPPERCASE and lowercase. Enter these elements as shown.
		`SELECT last_name, employee_id FROM employees;`
		`sqlplus hr/hr`
		`CREATE USER mjones IDENTIFIED BY ty3MU9;`

Documentation Accessibility

Our goal is to make Oracle products, services, and supporting documentation accessible, with good usability, to the disabled community. 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 Corporation 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/`

Accessibility of Code Examples in Documentation 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.

Accessibility of Links to External Web Sites in Documentation This documentation may contain links to Web sites of other companies or organizations that Oracle does not own or control. Oracle neither evaluates nor makes any representations regarding the accessibility of these Web sites.

1 Backup and Recovery Overview

This chapter provides a general overview of backup and recovery concepts, the files in an Oracle database related to backup and recovery, and the tools available for making backups of your database, recovering from data loss or other error, and maintaining records of your backups.

This chapter includes the following topics:

What is Backup and Recovery?

In general, **backup and recovery** refers to the various strategies and procedures involved in protecting your database against data loss and reconstructing the database after any kind of data loss.

Physical Backups and Logical Backups

A **backup** is a copy of data from your database that can be used to reconstruct that data. Backups can be divided into **physical backups** and **logical backups**.

Physical backups are backups of the physical files used in storing and recovering your database, such as datafiles, control files, and archived redo logs. Ultimately, every physical backup is a copy of files storing database information to some other location, whether on disk or some offline storage such as tape.

Logical backups contain logical data (for example, tables or stored procedures) exported from a database with an Oracle export utility and stored in a binary file, for later re-importing into a database using the corresponding Oracle import utility..

> **See also:** *Oracle Database Utilities* for more details about importing and exporting data using Oracle export and import utilities.

Physical backups are the foundation of any sound backup and recovery strategy. Logical backups are a useful supplement to physical backups in many circumstances but are not sufficient protection against data loss without physical backups.

Unless otherwise specified, the term "backup" as used in the backup and recovery documentation refers to physical backups, and to **backup** part or all of your database is to take some kind of physcial backup. The focus in the backup and recovery documentation set will be almost exclusively on physical backups.

Errors and Failures Requiring Recovery from Backup

While there are several types of problem that can halt the normal operation of an Oracle database or affect database I/O operations, only two typically require DBA intervention and media recovery: media failure, and user errors.

Other failures may require DBA intervention to restart the database (after an instance failure) or allocate more disk space (after statement failure due to, for instance, a full datafile) but these situations will not generally cause data loss or require recovery from backup.

User Error

User errors occur when, either due to an error in application logic or a manual mis-step, data in your database is changed or deleted incorrectly. Data loss due to user error includes such missteps as dropping important tables or deleting or changing the contents of a table. While user training a nd careful management of privileges can prevent most user errors, your backup strategy determines how gracefully you recover the lost data when user error does cause data loss.

Media Failure

A **media failure** is the failure of a read or write of a disk file required to run the database, due to a physical problem with the disk such as a head crash. Any database file can be vulnerable to a media failure.

The appropriate recovery from a media failure depends on the files affected and the types of backup available.

Oracle Backup and Recovery Solutions: RMAN and User-Managed Backup

For performing backup and recovery based on physical backups, you have two solutions available:

- **Recovery Manager (RMAN)**, a tool (with command-line client and Enterprise Manager GUI interfaces) that integrates with sessions running on the Oracle server to perform a range of backup and recovery activities, as well as maintaining a repository of historical data about your backups
- The traditional **user-managed backup and recovery**, where you directly manage the files that make up your database with a mixture of host operating system commands and SQL*Plus backup and recovery-related capabilities

Both methods are supported by Oracle Corporation and are fully documented. Recovery Manager is, however, the preferred solution for database backup and recovery. It can perform the same types of backup and recovery available through user-managed methods more easily, provides a common interface for backup tasks across different host operating systems, and offers a number of backup techniques not available through user-managed methods.

Most of the backup and recovery documentation set will focus on RMAN-based backup and recovery. User-managed backup and recovery techniques are covered in the later chapters of *Oracle Database Backup and Recovery Advanced User's Guide*.

Whether you use RMAN or user-managed methods, you can supplement your physical backups with logical backups of schema objects made using data export utilities. Data thus saved can later be imported to re-create this data after restore and recovery. However, logical backups are for the most part beyond the scope of the backup and recovery documentation.

Backup and Recovery: Basic Concepts

The physical structures of the database and the role each plays in the database recovery process are what determine the forms of backup and recovery available through user-managed techniques and through RMAN.

Physical Database Structures Used in Recovering Data

The files and other structures that make up an Oracle database store data and safeguard it against possible failures. This section introduces each of the physical structures that make up an Oracle database and their role in the reconstruction of a database from backup. This section contains these topics:

- Datafiles and Data Blocks
- Redo Logs
- Undo Segments
- Control Files

Datafiles and Data Blocks

An Oracle database consists of one or more logical storage units called tablespaces. Each tablespace in an Oracle database consists of one or more files called datafiles, physical files under the host operating system in which the database is running.

A database's data is collectively stored in the datafiles that constitute each tablespace of the database. The simplest Oracle database would have one tablespace, stored in one datafile. The datbase manages the storage space in the datafiles of a database in units called data blocks. A data block is the smallest unit of data used by a database. Data blocks are the smallest units of storage that the database can use or allocate.

Modified or new data is not written to datafiles immediately. Updates are buffered in memory and written to datafiles at intervals. If a database has not gone through a normal shutdown (that is, if it is open, or exited abnormally, as in an instance failure or a `SHUTDOWN ABORT`) then there are typically changes in memory that have not been written to the datafiles. Datafiles that were restored from backup, or were not closed during a **consistent shutdown,** are typically not completely up to date.

Copies of the datafiles of a database are a critical part of any backup.

> **See also:** *Oracle Database Concepts* for more detail about the structure and contents of datafiles and data blocks.

Redo Logs

Redo logs record all changes made to a database's data files. With a complete set of redo logs and an older copy of a datafile, the database can reapply the changes recorded in the redo logs to re-create the database at any point between the backup time and the end of the last redo log. Each time data is changed in the database, that change is recorded in the **online redo log** first, before it is applied to the datafiles. An Oracle database requires at least two **online redo log groups**, and in each group there is at least one **online redo log member**, an individual redo log file where the changes are recorded.

At intervals, the database rotates through the online redo log groups, storing changes in the **current online redo log** while the groups not in use can be copied to an archive location, where they are called **archived redo log**s (or, collectively, the **archived redo log**). You can run your database in ARCHIVELOG mode (in which this archiving of redo log files is enabled) or NOARCHIVELOG mode (in which redo log files are simply overwritten).

Preserving the archived redo log is a major part of most backup strategies, as they contain a record of all updates to datafiles. Backup strategies often involve copying the archived redo logs to disk or tape for longer-term storage. Running in NOARCHIVELOG mode limits your data recovery options.

> **See also:** *Oracle Database Administrator's Guide* for more details about the online redo logs, *Oracle Database Administrator's Guide* for more details about archived redo logs, and "Deciding Between ARCHIVELOG and NOARCHIVELOG Mode" on page 2-10 for a discussion of the implications of archiving or discarding your redo log files.

Control Files

The control file contains a crucial record of the physical structures of the database and their status. Several types of information stored in the control file are related to backup and recovery:

- Database information (RESETLOGS SCN and time stamp)
- Tablespace and datafile records (filenames, datafile checkpoints, read/write status, offline ranges)
- Information about redo threads (current online redo log)
- Log records (log sequence numbers, SCN range in each log)
- A record of past RMAN backups

- Information about corrupt datafile blocks

The recovery process for datafiles is in part guided by status information in the control file, such as the database checkpoints, current online redo log file, and the **datafile header** checkpoints for the datafiles. Loss of the control file makes recovery from a data loss much more difficult.

> **See also:** *Oracle Database Concepts* for more information about control files.

Undo Segments

In general, when data in a datafile is updated, "before images" of that data are written into **undo segments**. If a transaction is rolled back, this undo information can be used to restore the original datafile contents.

In the context of recovery, the undo information is used to undo the effects of uncommitted transactions, once all the datafile changes from the redo logs have been applied to the datafiles. The database is actually opened before the undo is applied.

You should not have to concern yourself with undo segments or manage them directly as part of your backup and recovery process.

> **See also:** *Oracle Database Concepts* for detailed information about undo segements.

The Database Recovery Process: Basic Concepts

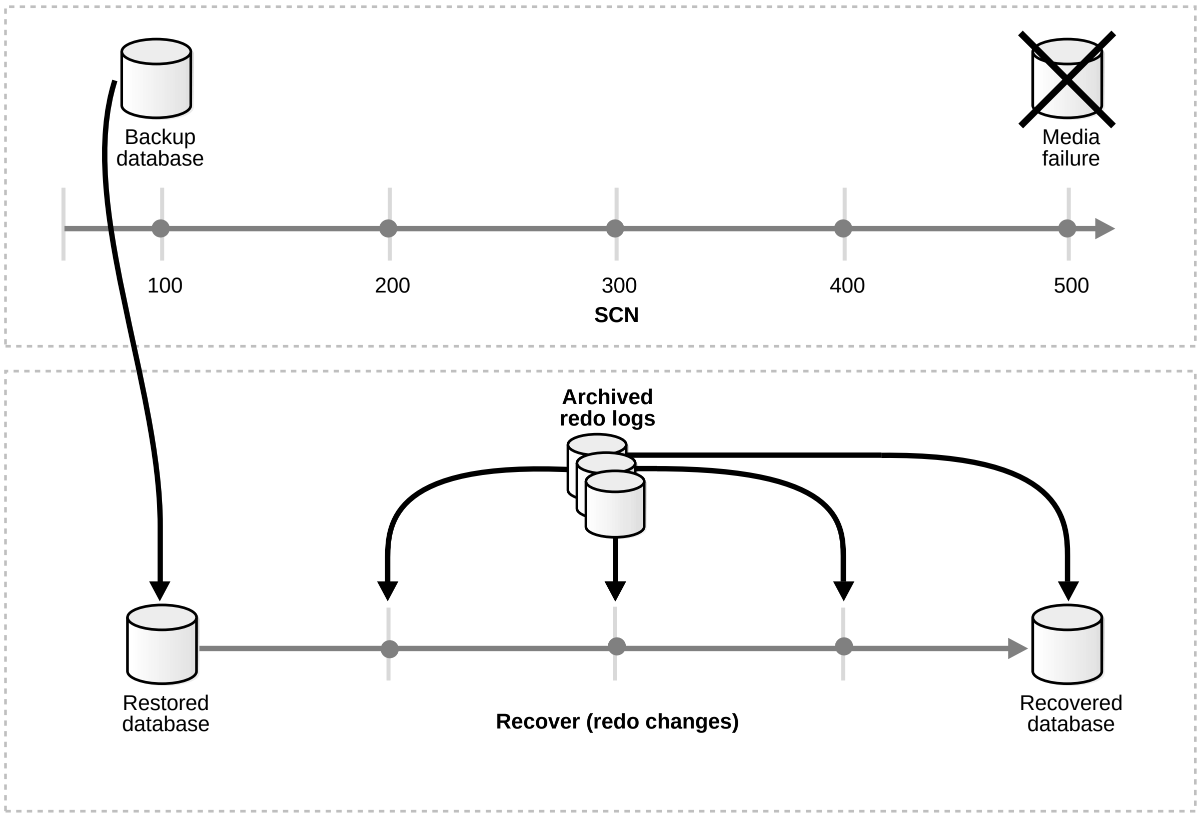
Reconstructing the contents of all or part of a database from a backup typically involves two phases: retrieving a copy of the datafile from a backup, and reapplying changes to the file since the backup from on the archived and online redo logs, to bring the database to the desired SCN (usually the most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

To **restore** a datafile or control file from backup is to retrieve the file onto disk from a backup location on tape, disk or other media, and make it available to the database server.

To **recover** a datafile (also called **performing recovery** on a datafile), is to take a restored copy of the datafile and apply to it changes recorded in the database's redo logs. To recover a whole database is to perform recovery on each of its datafiles.

Figure 1–1 illustrates the basic principle of backing up, restoring, and recovering a database. Most of the data recovery procedures supported by the Oracle database are variations on the process described here.

Figure 1–1 Restoring and Recovering a Database



In this example a full backup of a database (copies of its datafiles and control file) is taken at SCN 100. Redo logs generated during the operation of the database capture all changes that occur between SCN 100 and SCN 500. Along the way, some logs fill and are archived. At SCN 500, the datafiles of the database are lost due to a media failure. The database is then returned to its transaction-consistent state at SCN 500, by restoring the datafiles from the backup taken at SCN 100, then applying the transactions captured in the archived and online redo logs and undoing the uncomitted transactions.

Forms of Data Recovery

The preceding scenario outlined the basics of the restore-and-recovery process. Several variants on this scenario are important to your backup and recovery work.

This section contains the following topics:

- Datafile Media Recovery: Restore Datafiles, Apply Redo
- Automatic Recovery After Instance Failure: Crash Recovery
- Complete, Incomplete and Point-In-Time Recovery

Datafile Media Recovery: Restore Datafiles, Apply Redo

Datafile media recovery (often simply called **media recovery**) is the most basic form of user-initiated data recovery. It can be used to recover from a lost or damaged current datafile, SPFILE or control file. It can also recover changes that were recorded in the redo logs but not in the datafiles for a tablespace that went offline without the `OFFLINE NORMAL` option. Datafile media recovery can be performed whether you use Recovery Manager or user-managed backup and recovery. (For user-managed backup and recovery, it is in fact the main option available.)

Like crash recovery, datafile media recovery is intended to restore database integrity. However, there are a number of important differences between the two:

- Media recovery must be explicitly invoked by a user. The database will not run media recovery on its own.
- Media recovery applies needed changes to datafiles that have been restored from backup, not to online datafiles left over after a crash.
- Media recovery must use archived logs as well as the online logs, to find changes reaching back to the time of the datafile backup.

The need to restore a datafile from backup is not detected automatically. The first step in performing media recovery is to manually restore the datafile by copying it from a backup. Once a datafile has been copied from backup, however, the database does automatically detect that this datafile is out of date and must undergo media recovery.

Several situations force you to perform media recovery:

- You restore a backup of a datafile.
- You restore a backup control file (even if all datafiles are current).

- A datafile is taken offline (either by you or automatically by the database) without the `OFFLINE NORMAL` option.

For a datafile to be available for media recovery, one of two things must be true:

- The database that the datafile belongs to must not be open;

or

- The specific datafile needing recovery must be offline, if the database is open.

A datafile that needs media recovery cannot be brought online until media recovery has been completed. A database cannot be opened if any of the online datafiles needs media recovery.

You can manage the expected duration of media recovery as part of your backup and recovery strategy. It is affected by, for example, the frequency of backups and parallel recovery parameters.

Complete, Incomplete and Point-In-Time Recovery

Complete recovery is recovering a database to the most recent point in time, without the loss of any committed transactions. Generally, the term "recovery" refers to complete recovery.

Occasionally, however, you need to return a database to its state at a past point in time. For example, to undo the effect of a user error, such as dropping or deleting the contents of a table, you may want to return the database to its contents before the delete occurred. In **incomplete recovery**, also known as **point-in-time recovery**, the goal is to restore the database to its state at some previous target SCN or time. Point-in-time recovery is one possible response to a data loss caused by, for instance, a user error or logical corruption that goes unnoticed for some time.

Point-in-time recovery is also your only option if you have to perform a recovery and discover that you are missing an archived log covering time between the backup you are restoring from and the target SCN for the recovery. Without the missing log, you have no record of the updates to your datafiles during that period. Your only choice is to recover the database from the point in time of the restored backup, as far as the unbroken series of archived logs permits, then perform an OPEN RESETLOGS and abandon all changes in or after the missing log. (If you discover that you have lost archived logs and your database is still up, you should do a full backup immediately.)

Note: If only one tablespace is affected by the data loss, you have the option of performing point-in-time recovery on that tablespace instead of the entire database. Tablespace point-in-time recovery (often abbreviated TSPITR) is an advanced technique documented in *Oracle Database Backup and Recovery Advanced User's Guide.*

Automatic Recovery After Instance Failure: Crash Recovery

The **crash recovery process** is a special form of recovery, which happens the first time an Oracle database instance is started after a crash (or `SHUTDOWN ABORT`). In crash recovery, the goal is to bring the datafiles to a transaction-consistent state, preserving all committed changes up to the point when the instance failed.

Unlike the forms of recovery performed manually after a data loss, crash recovery uses only the online redo log files and current online datafiles, as left on disk after the instance failure. Archived logs are never used during crash recovery, and datafiles are never restored from backup.

The database applies any pending updates in the online redo logs to the online datafiles of your database. The result is that, whenever the database is restarted after a crash, the datafiles reflect all committed changes up to the moment when the failure occurred. (After the database opens, any changes that were part of uncommitted transactions at the time of the crash are rolled back.)

The duration of crash recovery is a function of the number of instances needing recovery, amount of redo generated in the redo threads of crashed instances since the last checkpoint, and user-configurable factors such as the number and size of redo log files, checkpoint frequency, and the parallel recovery setting.You can set parameters in the database server that can tune the duration of crash recovery. You can also tune checkpointing to optimize recovery time.

Note: Crash recovery in a Real Application Cluster (RAC) database takes place when all instances in the cluster have failed. The related process of **instance recovery** takes place when some but not all instances fail. For more information on crash and instance recovery in the context of RAC, refer to *Real Application Clusters Quick Start*.

Backup and Recovery with RMAN

As noted earlier, using RMAN gives you access to several data backup and recovery techniques and features not available at all with user-managed backup and recovery. The most noteworthy are:

- **Incremental backups**, which provide more compact backups (storing only changed blocks) and faster datafile media recovery (reducing the need to apply redo during datafile media recovery)
- **Block media recovery**, in which a datafile with only a small number of corrupt data blocks can be repaired without being taken offline or restored from backup
- **Unused block compression**, in which never-used data blocks in a datafile are not copied into some backups to save disk space and backup time
- **Binary compression**, which uses a compression mechanism integrated into the Oracle database server to reduce the size of backups saved on disk

A complete list of feature differences between RMAN and user-managed backup and recovery can be found in "Feature Comparison of Backup Methods" on page 1-18.

RMAN also reduces the administration work associated with your backup strategy. RMAN keeps an extensive record of metadata about backups, archived logs, and its own activities, known as the **RMAN repository**. In restore operations, RMAN can use this information to eliminate the need for you to identify backup files for use in restores in most situations. You can also generate reports of backup activity using the information in the repository.

Primary storage for RMAN repository information is in the control file of the production database. You can also set up an independent **recovery catalog**, a schema that stores RMAN repository information for one or many databases in a separate **recovery catalog database**.

The remainder of this book, *Oracle Database Backup and Recovery Basics*, focuses on using RMAN to implement your backup and recovery strategy.

Types of Oracle Database Backup under RMAN

There are several ways of distinguishing among physical backups, according to the state the database was in when the backup was created, what parts of the database were actually backed up, and how the resulting backup was stored.

Consistent and Inconsistent Backups

Physical backups can also be divided into **consistent** and **inconsistent** backups. Consistent backups are those created when the database is in a consistent state, that is, when all changes in the redo log have been applied to the datafiles. A database restored from a consistent backup can be opened immediately, without undergoing media recovery. However, a consistent backup can only be created after the database has been shut down normally, that is, not after a crash or a `SHUTDOWN ABORT`.

For reasons of availability, the Oracle database is designed to work equally well with an inconsistent backup, a backup taken while the database is open. When a database is restored from an inconsistent backup, it must undergo media recovery, so that the database can apply any pending changes from the online and archived redo log before the database is opened again. Because archived logs are required, using inconsistent backups requires that your database be run in `ARCHIVELOG` mode.

Full and Incremental Backups

Full backups are backups which include datafiles in their entirety. Full backups can be created with Recovery Manager or with operating system-level file copy commands. Incremental backups are based on the idea of making copies only of changed data blocks in a data file. In recovery, extracting entire changed blocks from an incremental backup can substitute for applicationof redo for individual datafile updates during the time covered by the backup, shortening recovery times considerably. Incremental backups can only be created with RMAN.

See Also: "Full and Incremental Datafile Backups" on page 4-5 for more details about the different ways to back up datafiles.

Image Copies, Backup Sets and Backup Pieces

The results of an Oracle database backup created through RMAN can be either image copies or backup sets. An **image copy** is a bit-for-bit identical copy of a database file. These can be created using operating system commands such as `cp` in Unix or `COPY` in Windows. RMAN can also create image copy backups, although in the process, RMAN will check the contents for corruption, something that native operating-system file copy utilities cannot do. RMAN records image copies it creates in the RMAN repository, so that it can use them when restoring your database. If you create image copies outside of RMAN, you can catalog them manually into the RMAN repository.

RMAN can also store its backups in an RMAN-exclusive format called a **backup set**. A backup set is a collection of files called **backup pieces**, each of which may contain the backup of one or several database files. A backup task performed in RMAN can create one or more backup sets, which are recorded in the RMAN repository. Backup sets are also the only form in which RMAN can write backups to media manager devices like tape libraries. Backup sets are only created and accessed through Recovery Manager.

> **See Also:** "RMAN Backup Formats: Image Copies and Backup Sets" on page 4-3 for more details about image copies and backup sets.

Matching Failures to Backup and Recovery Techniques

In planning your database backup and recovery strategy, you must try to anticipate the errors that will arise, and put in place the backups needed to recover from them.While there are several types of problem that can halt the normal operation of an Oracle database or affect database I/O operations, only two typically require DBA intervention and media recovery: media failure, and user errors. Instance failures, network failures, failures of Oracle database background processes and failure of a statement to execute due to, for instance, exhaustion of some resource such as space in a datafile may require DBA intervention, and might even crash a database instance, but will not generally cause data loss or the need to recover from backup.

This section contains these topics:

- Media Failure
- User Error

Media Failure

Database operation after a media failure of online redo log files or control files depends on whether the online redo log or control file is protected by **multiplexing**, as recommended. When an online redo log or control file is multiplexed, multiple copies of the file are maintained on the system. Multiplexed files should be stored on separate disks.

If a media failure damages a disk containing one copy of a multiplexed online redo log, then the database can usually continue to operate without significant interruption. Damage to a nonmultiplexed online redo log causes database operation to halt and may cause permanent loss of data.

Damage to any control file, whether it is multiplexed or not, halts database operation when the database attempts to read or write to the damaged control file (which happens frequently, for example at every checkpoint and log switch).

Media failures that affect datafiles can be divided into two categories: **read errors** and **write errors**. In a read error, the instance cannot read a datafile and an operating system error is returned to the application, along with an error indicating that the file cannot be found, cannot be opened, or cannot be read. The database continues to run, but the error is returned each time an unsuccessful read occurs. At the next checkpoint, a write error will occur when the database attempts to write the file header as part of the standard checkpoint process.

The effect of a datafile write error depends upon which tablespace the datafile is in.

If the instance cannot write to a datafile in the **SYSTEM tablespace**, an undo tablespace (if the database is in **automatic undo management mode**, which is the preferred choice in Release 10*g*), or a datafile in a tablespace containing active rollback segments (if in **manual undo management mode**), then the database issues an error and shuts down the instance. All files in the SYSTEM tablespace and all datafiles containing rollback segments must be online in order for the database to operate properly.

If the instance cannot write to a datafile other than those in the preceding list, then the result depends on whether the database is running in ARCHIVELOG mode or not.

In ARCHIVELOG mode, the database records an error in the database writer trace file and takes the affected datafile offline. (All other datafiles in the tablespace containing this datafile remain online.) You can then rectify the underlying problem and restore and recover the affected tablespace.

In NOARCHIVELOG mode, the database writer background process DBWR fails, and the instance fails, the cause of the problem determines the required response. If the problem is temporary (for example, a disk controller was powered off), then crash recovery usually can be performed using the online redo log files. In such situations, the instance can be restarted without resorting to media recovery. However, if a datafile is permanently damaged, then you must restore a **consistent backup** of the database.

Recovery of Read-Only Tablespaces

Recovery is not needed on any **read-only tablespace** during crash or instance recovery. During startup, recovery verifies that each online read-only datafile does not need media recovery. That is, the file was not restored from a backup taken before it was made read-only. If you restore a read-only tablespace from a backup taken before the tablespace was made read-only, then you cannot access the tablespace until you complete media recovery.

User Error

Typically, a user error like dropping a table or deleting rows from a table requires one of the following responses:

- Re-importing the dropped object, if a suitable export file exists or if the object is still available at a standby database
- Performing **tablespace point-in-time recovery (TSPITR)** of one or more tablespaces
- Re-entering the lost data manually, if a record of them exists
- Returning the database to a past state using database point-in-time recovery
- Using one of the flashback features of the Oracle database to recover from logical corruption by returning affected objects to a past state

The recovery options available to you will be a function of your backup strategy. For example, if you are running in NOARCHIVELOG mode then you have limited point-in-time recovery options.

See Also:

- *Oracle Database Backup and Recovery Advanced User's Guide* to learn how to perform point-in-time recovery for an entire database
- *Oracle Database Backup and Recovery Advanced User's Guide* to learn how to perform tablespace point-in-time recovery
- *Oracle Database Backup and Recovery Advanced User's Guide* to learn how to use the flashback features of the Oracle database

Automatic Disk-Based Backup and Recovery: The Flash Recovery Area

The components that creates different backup and recovery-related files have no knowledge of each other or of the size of the file systems where they store their data.   With Automatic Disk-Based Backup and Recovery, you can create a flash recovery area, which automates management of backup-related files. Choose a location on disk and an upper bound for storage space, and set a retention policy that governs how long backup files are needed for recovery, and the database manages the storage used for backups, archived redo logs, and other recovery-related files for your database within that space. Files no longer needed are eligible for deletion when RMAN needs to reclaim space for new files.   If you do not use a flash recovery area, you must manually manage disk space for your backup-related files and balance the use of space among the different types of files. Oracle Corporation recommends that you enable a flash recovery area to simplify your backup management.

System Requirements for Backup and Recovery Methods

When choosing a backup and recovery solution, find one that is appropriate for the database environment. For example, if you manage only databases of release 8.0 or higher, then you can use RMAN to manage your backup and recovery requirements. Releases older than 8.0 will have to be managed using some method besides RMAN.

Table 1–1 describes the version and system requirements for different database backup and recovery methods.

Table 1–1   Requirements for Different Backup Methods

Backup Method	Type	Version Available	Requirements
Recovery Manager (RMAN)	Physical	Oracle version 8.0 and higher	Third-party media manager (only if backing up to tape)
Operating System	Physical	All versions	Operating system backup utility (for example, UNIX `dd` command)
Export	Logical	All versions	N/A

Feature Comparison of Backup Methods

Besides being limited by system requirements, the backup and recovery solution you choose should be driven by the features that you want. Table 1–2 compares the features of the different backup methods.

Table 1–2 Feature Comparison of Backup Methods **(Page 1 of 2)**

Feature	Recovery Manager	User-Managed	Export
Closed database backups	Supported. Requires instance to be mounted.	Supported.	Not supported.
Open database backups	Supported. No need to use `BEGIN`/`END BACKUP` statements.	Supported. Must use `BEGIN`/`END BACKUP` statements.	Requires rollback or undo segments to generate consistent backups.
Incremental backups	Supported.	Not supported.	Not supported.
Corrupt block detection	Supported. Identifies corrupt blocks and logs in `V$DATABASE_CORRUPTION`.	Not supported.	Supported. Identifies corrupt blocks in the export log.
Automatic record keeping of files in backups	Supported. Establishes the name and locations of all files to be backed up (whole database, tablespace, datafile or control file backup).	Not supported. Files to be backed up must be specified manually.	Supported. Performs either full, user, or table backups.
Backup catalogs	Supported. Backups are recorded inthe RMAN repository, which is contained in the control file and optionally in the recovery catalog database.	Not supported.	Not supported.
Backups to media manager	Supported. Interfaces with a media manager. RMAN also supports proxy copy, a feature that allows the media manager to manage the transfer of data.	Supported. Backup to tape is manual or controlled by a media manager.	Supported.

Table 1–2 Feature Comparison of Backup Methods **(Page 2 of 2)**

Feature	Recovery Manager	User-Managed	Export
Backs up initialization parameter file	Supported.	Supported.	Not supported.
Backs up password and networking files	Not supported.	Supported.	Not supported.
Platform-independent language for backups	Supported.	Not supported.	Supported.

2 Backup and Recovery Strategies

This chapter offers guidelines and considerations for developing an effective backup and recovery strategy.

This section includes the following topics:

Data Recovery Strategy Determines Backup Strategy

To decide on backup strategies, start with your data recovery requirements and your data recovery strategy. Each type of data recovery will require that you take certain types of backup.

Failures can run the gamut from user error, datafile block corruption and media failure to situations like the complete loss of a data center. How quickly you can resume normal operation of your database is a function of what kinds of restore and recovery techniques you include in your planning. Each restore and recovery technique will impose requirements on your backup strategy, including which features of the Oracle database you use to take, store and manage your backups.

When thinking about recovery strategies, ask yourself questions like these:

- If a disk failed and destroyed some of the database files, such as datafiles or redo logs, how would you recover the lost files? As described in "Planning a Response to Media Failure: Restore and Media Recovery" on page 2-6, you should be able to handle the loss of datafiles, control files, and online redo logs.
- If a logic error in an application or a user error caused the loss of important data from one or several tables or tablespaces, how could you recover that data, and what would happen to database updates since the error? Could you determine the cause of the error, to prevent it from happening again? As described in "Planning a Response to User Error: Point-in-Time Recovery and Flashback Features" on page 2-5, techniques available to you include point-in-time recovery of the whole database or one or more tablespaces, importing data from earlier logical exports with one of the data import utilities, and using the Oracle database's flashback features.
- If the instance alert log indicates that one or more tables contains corrupt blocks, how can you repair the corruption? Does the tablespace have to remain available during the repair? As described in "Planning a Response to Datafile Block Corruption: Block Media Recovery" on page 2-7, the RMAN `BLOCKRECOVER` command can help you in this situation. Also, troubleshoot recovery with the SQL*Plus `RECOVER ... TEST` command.
- If the entire data center is destroyed, can you perform disaster recovery? Assume that all you have is an archive tape containing backups. How would you recover the database? How long would that recovery take?
- If you were not available to recover your database, could someone else recover it in your absence? Are your recovery procedures sufficiently automated and documented?

With these needs in mind, decide how you can take advantage of features related to backup and recovery, and look at how each feature meets some requirement of your backup strategy. For example:

- Using Recovery Manager simplifies most backup and recovery operations compared to user-managed backup and recovery. It automates management of most backup files, including the deletion of backups and archived redo logs from disk or tape when no longer needed to meet recovery goals. It provides detailed reporting on backup activities, can verify that your available backups can be used to recover your database. Finally, RMAN makes possible many recovery techniques not available if you are using user-managed backup and recovery, such as incremental backups.
- Flashback Database will help you restore a database to a previous time much faster than media recovery. However, you must decide in advance to keep flashback logs, and keeping flashback logs requires that you configure a flash recovery area.
- Block media recovery may be better than datafile media recovery if availability is critical. While block media recovery is possible even if you do not base your backup and recovery strategy on RMAN, RMAN-based block media recovery can be performed more quickly and with less effort.

Once you decide which features to use in your recovery strategy, you can plan your backup strategy, answering the following questions, among others:

- How and where will you store your recovery-related files? Will you use a flash recovery area? Will you use an ASM disk group? Will you store backups on tape or other offline storage, or only on disk?
- At what intervals will you take scheduled backups? And what form of physical backups will you take in each situation?
- What situations require you to take a database backup outside of the regular schedule? Sometimes you must take an unscheduled backup to ensure that you can recover your data, such as after an `OPEN RESETLOGS` or after changes to your database such as NOLOGGING operations that do not appear in the redo log. You may also have business requirements that require backups for auditing purposes or other reasons not related to database recovery.
- How can you validate your backups, to ensure that you can recover your database when necessary?
- How do you manage records of your backups?

- Do you have detailed recovery plans that cover each type of failure? How do your DBAs can execute these plans in a crisis? Can scripts be written to automate execution of these plans in a crisis?
- Can you apply Oracle database availability technologies, such as Data Guard or Real Application Clusters, to improve availability during a database failure? How does using these availability technologies affect your backup and recovery strategy?

These are of course only a few of the considerations you should take into account. Available resources (hardware, media, staff, budget, and so on) will also be factors in your decision.

Planning Data Recovery Strategy

Your data recovery strategy should include responses to any number of database failure scenarios. The key to an effective, efficient strategy is envisioning failure modes, matching Oracle database recovery techniques and tools to the failure modes in which they are useful, and then making sure you incorporate the necessary backup types to support those recovery techniques.

To help match failure modes to recovery techniques that can help resolve them, refer to the following sections:

- Planning a Response to User Error: Point-in-Time Recovery and Flashback Features
- Planning a Response to Media Failure: Restore and Media Recovery
- Planning a Response to Datafile Block Corruption: Block Media Recovery

Planning a Response to User Error: Point-in-Time Recovery and Flashback Features

Your backup and recovery strategy should enable you to handle situations in which a user or application makes unwanted changes to database data, such as deleting the contents of a table or making incorrect updates during a batch run. The goal in such a case will be to restore the affected parts of your database to their state before the user error.

Depending on the situation, your appropriate response will be one of the following:

- If you have performed a logical backup by exporting the contents of the affected tables, sometimes you can import the data back into the table. This technique presumes that you are regularly exporting logical backups of your data, and that any changes between exports are unimportant.
- You can perform point-in-time recovery, bringing one tablespace or the whole database back to its state before the time of the error. In either case, you need

backups from before the time of the error, plus the redo logs from the time of the backup to the time of the error.

Note: Oracle's Flashback Technology provides faster and less disruptive alternatives to media recovery in many circumstances.

- Oracle Flashback Database is a physical-level recovery mechanism similar to media recovery, but generally faster and not requiring the restore of data from backup.
- Oracle Flashback Table and Oracle Flashback Drop work at the logical level, undoing unwanted changes to tables, including reversing the effects of `DROP TABLE` statements.
- Oracle Flashback Query and Oracle Flashback Version Query are useful in viewing past contents of tables and investigating how and when logical corruptions affected your database.

Information about these features is collected in *Oracle Database Backup and Recovery Advanced User's Guide*. This document will allude to such features where they can be helpful and provide pointers for more information. Familiarize yourself with these features before planning your backup and recovery strategy, because you may find that they can be quite valuable and require limited advanced planning.

Planning a Response to Media Failure: Restore and Media Recovery

A media failure occurs when a problem external to the database prevents Oracle from reading from or writing to a file during database operations. Typical media failures include physical failures, such as head crashes, and the overwriting, deletion or corruption of a database file. Media failures are less common than user or application errors, but your backup and recovery strategy should prepare for them.

The type of media failure determines the recovery technique to use. For example, the strategy you use to recover from a corrupted datafile is different from the strategy for recovering from the loss of the control file.

Example: Online Redo Log Recovery

The method of recovery from loss of all members of an online log group depends on a number of factors, such as:

- The state of the database (open, crashed, closed consistently, and so on)
- Whether the lost redo log group was current
- Whether the lost redo log group was archived

For example:

- If you lose the current group, and the database is not closed consistently (either it is open, or it has crashed), then you will have to restore an old backup and perform point-in-time recovery, followed by `OPEN RESETLOGS`. You will lose all transactions that were in the lost log. You should take a new full database backup immediately after the `OPEN RESETLOGS`. Backups from before the `OPEN RESETLOGS` will not be recoverable because of the lost log.
- If you lose the current redo log group, and if the database is closed consistently, then you can perform `OPEN RESETLOGS` with no transaction loss. However, you should take a new full database backup. Backups from before the OPEN RESETLOGS will not be recoverable because of the lost log.
- If you lose a noncurrent redo log group, then you can use the `ALTER DATABASE CLEAR LOGFILE` statement to re-create all members in the group. No transactions are lost. If the lost redo log group was archived before it was lost, then nothing further is required. Otherwise, you should immediately take a new full backup of your database. Backups from before the log was lost will not be recoverable because of the lost log.

Planning a Response to Datafile Block Corruption: Block Media Recovery

If a small number of blocks within one or more datafiles are corrupt, you can perform **block media recovery** instead of restoring the datafiles from backup and performing complete media recovery of those files. The Recovery Manager `BLOCKRECOVER` command can be used to restore and recover specified data blocks while the database is open and the corrupted datafile is online.

See Also: *Oracle Database Backup and Recovery Advanced User's Guide* to learn how to perform block media recovery with RMAN.

Planning Backup Strategy

Your plans for data recovery strategies are the basis of your plans for backup strategy. This discussion describes general guidelines that can help you decide when to perform database backups, which parts of a database you should back up, what tools Oracle provides for those backups, and how to configure your database to improve its robustness and make backup and recovery easier. Of course, the specifics of your strategy must balance the needs of your restore strategy with questions of cost, resources, personnel and other factors.

Protecting Your Redundancy Set

The set of files needed to recover an Oracle database from the failure of any of its files—a datafile, control file, or online redo log—is called the **redundancy set**. The redundancy set should contain:

- The last backup of the control file and all the datafiles
- All archived redo logs generated after the last backup was taken
- Duplicates of the online redo log files, generated by Oracle database multiplexing, operating system mirroring, or both
- Duplicates of the current control file, generated by Oracle database multiplexing, operating system mirroring, or both
- Copies of configuration files such as the server parameter file, `tnsnames.ora`, and `listener.ora`

The first rule of protecting your redundancy set is:

> *The set of disks or other media that contain the redundancy set for your database should be separate from the disks that contain the datafiles, online redo logs, and control files.*

This practice ensures that the failure of a disk that contains a datafile does not also cause the loss of the backups or redo logs needed to recover the datafile. Consequently, a minimal production-level database requires at least two disk drives: one to hold the files in the redundancy set and one to hold the database files. Ideally, separate the redundancy set from the primary files in every way possible: on separate volumes, separate file systems, and separate RAID devices. Keeping the redundancy set separate from the primary files ensures that you will not lose committed transactions in a disk failure.

The simplest way to manage your redundancy set is to use a flash recovery area, on a separate device from the working set files. All recovery-related files will be stored in a single location on disk, disk space usage is managed automatically, backups required to meet your data recovery requirements are never deleted from disk while they are still needed, and recovery time is minimized without compromising the completeness of the redundancy set.

Whether or not you use a flash recovery area, Oracle Corporation recommends following these guidelines:

- Multiplex the online redo log files and current control file at the database level. (For instance, configure the database to write its online logs to two or more destinations, so that each write is a separate operation carried out by the database, rather than by operating system-level or hardware-level redundancy.) If you multiplex at the database level, then an I/O failure or lost write should only corrupt one of the copies.

  Ideally, the multiplexed files should be on different disks mounted under different disk controllers. The flash recovery area is an excellent location for one copy of these files.

  You can also mirror the online redo logs and current control file at the operating system or hardware level, but this is not a substitute for multiplexing at the database level.
- If running in ARCHIVELOG mode, archive the redo logs to multiple locations, ideally on different disks. If you are using a flash recovery area, use it as one of the archiving locations.
- Use operating system or hardware mirroring for the control file. All copies of the control file multiplexed at the database level must be available at all times, or the instance will crash. If you use operating system or hardware mirroring for your control file, your database can continue to operate even if one copy of the control file mirrored at the operating system level is unavailable due to a disk failure.
- Use operating system or hardware mirroring for the primary datafiles if possible, to avoid having to perform media recovery for simple disk failures.
- Keep at least one copy of the entire redundancy set—including the most recent backup—on disk. The flash recovery area is the ideal location for the redundancy set.
- If the target database is stored on a RAID device, then store the redundancy set on a set of disks that are not in the same RAID device.
- If you store the redundancy set on tape, then maintain at least two copies of the data to protect against the risk of tape failure. Also, if you have more than one copy of the same data, then consider keeping backups from different points in

time. In this way, if one backup or split mirror was done when the database was corrupted, then you have an older backup when the database was not corrupted.

Deciding Between ARCHIVELOG and NOARCHIVELOG Mode

The redo logs of your database provide a complete record of changes to the datafiles of your database (with a few exceptions, such as direct path loads).

You can run your database in one of two modes: ARCHIVELOG mode or NOARCHIVELOG mode. In ARCHIVELOG mode, a used online redo log group must be copied to one or more archive destinations before it can be reused. Archiving the redo log preserves all transactions stored in that log, so that they can be used in recovery operations later. In NOARCHIVELOG mode, the online redo log groups are simply overwritten when the log is re-used. All information about transactions recorded in that redo log group is lost.

Implications of Running in NOARCHIVELOG Mode

Running your database in NOARCHIVELOG mode imposes severe limitations on your backup and recovery strategy.

- You cannot perform online backups of your database. You must shut your database down cleanly before you can take a backup in NOARCHIVELOG mode.
- You cannot use any data recovery techniques that require the archived redo logs. These include complete and point-in-time media recovery, as described in "Forms of Data Recovery" on page 1-9, and more advanced recovery techniques such as point-in-time recovery of individual tablespaces and Flashback Database (described in *Oracle Database Backup and Recovery Advanced User's Guide.*).

If you are running in NOARCHIVELOG mode and you must recover from damage to datafiles due to disk failure, you have two main options for recovery:

- Drop all objects that have any extents located in the affected files, and then drop the files. The remainder of the database is intact, but all data in the affected files is lost.
- Restore the entire database from the most recent backup, and lose all changes to the database since the backup. (Recovering changes since the backup would require performing media recovery, which uses the archived redo logs.)

Implications of Running in ARCHIVELOG Mode

For most applications, running in ARCHIVELOG mode is preferable to running in NOARCHIVELOG mode because you have more flexible recovery options after a data loss. There are, however, associated costs of running in ARCHIVELOG mode:

- Space must be set aside for archiving destinations, locations on disk where the archived redo logs will be stored. These can become quite large in databases with large numbers of updates.
- The stored archived redo logs must be managed. To limit the disk space used by archived redo logs, archived redo logs can be moved to tape for longer-term storage, and older logs no longer needed to meet your recoverability goals should be deleted. (RMAN can automate most of the management of archived redo logs, by recording the location and contents of all archived redo logs, making it easy to move archived logs to tape, and identifying and deleting redo logs no longer required to meet your recoverability objectives.)
- Some performance overhead is associated with the background processes ARC0 through ARC*n* which copy filled online redo logs to the archiving destinations.

When performance requirements are extreme or disk space limitations are severe, it may be preferable to run in `NOARCHIVELOG` mode in spite of the restrictions imposed.

Deciding Whether to Use a Flash Recovery Area

It is recommended that you take advantage of the flash recovery area to store as many backup and recovery-related fileas as possible, including disk backups and archived redo logs.

Some features of Oracle database backup and recovery, such as Oracle Flashback Database, require the use of a flash recovery area. In such cases, you must create a flash recovery area, though you do not have to use it to store all recovery-related files.

Even when its use is not required, however, the flash recovery area offers a number of advantages over other on-disk backup storage methods. Backups moved to tape from the flash recovery area are retained on disk until space is needed for other required files, reducing the need to restore backups from tape. At the same time, obsolete files no longer needed to meet your recoverability goals and files backed up to tape become eligible for deletion and are deleted when space is needed, eliminating the need for DBA intervention to clear out old files.

> **See Also:** "Setting Up a Flash Recovery Area for RMAN" on page 3-11 for more about the uses and benefits of the flash recovery area.

Choosing a Backup Retention Policy

Your backup retention policy is the rule you set regarding which backups must be retained (whether on disk or other backup media) to meet your recovery and other requirements.

Backup retention policy can be based on **redundancy** or a **recovery window**. In a redundancy-based retention policy, you specify a number *n* such that you always keep at least *n* distinct backups of each file in your database. In a recovery window-based retention policy, you specify a time interval in the past (for example, one week, or one month) and keep all backups required to let you perform point-in-time recovery to any point during that window.

A backup no longer needed to satisfy the backup retention policy is said to be **obsolete**.

Implementing Backup Retention Policy with RMAN

RMAN automates the implementation of a backup retention policy, using the following commands:

- `CONFIGURE RETENTION POLICY` command lets you set the retention policy that will apply to all of your database files by default.
- `REPORT OBSOLETE` command lets you list backups currently on disk that are obsolete under the retention policy. You can also specify parameters to see which files would be obsolete under different retention policies.
- `DELETE OBSOLETE` command deletes the files which `REPORT OBSOLETE` would list as obsolete.
- `CHANGE... KEEP` lets you set a separate retention policy for specific backups, such as **long-term backups** kept for archival purposes. You can specify that a given backup must be kept until a future time, or even specify that a backup be kept forever. `CHANGE... NOKEEP` is used to let the retention policy apply to a backup previously protected by `CHANGE... KEEP`.

If you use a flash recovery area to store your backups, the database will delete obsolete backups automatically as disk space is needed for newer backups, archived logs and other files. For backups stored on disk outside a flash recovery area and for backups stored on tape, you should periodically run the `DELETE OBSOLETE` command to remove obsolete backups.

Recovery Window-Based Backup Retention Policy

A recovery window-based retention policy lets you guarantee that you can perform point-in-time recovery to any point in the past, up to a number of days that you specify. The earliest point in time to which you can recover your database under your retention policy is known as the **point of recoverability**. All backups required for recovery or point-in-time recovery back to that time will be retained.

Note that this will generally require that you keep backups older than the beginning of the recovery window. A point-in-time recovery to the beginning of the recovery window would require a restore from this backup, and then applying all changes between the backup time and the point of recoverability. For example, you might configure a recovery window of three days:

```
RMAN> CONFIGURE RETENTION POLICY TO RECOVERY WINDOW OF 3 DAYS;
```

If your last full database backup was six days ago, RMAN will keep the six-day-old backup, and all redo logs required to roll the database forward to the beginning of the recovery window three days ago, in addition to any backups and redo logs needed to recover the database to all points in time within the three day window.

A recovery window-based backup retention policy provides the most certain recoverability for your data. The disadvantage is that more careful disk space planning is required, since it may not be obvious how many backups of datafiles and archived logs must be retained to guarantee the recovery window.

Redundancy-Based Backup Retention Policy

A redundancy-based backup retention policy determines whether a backup is obsolete based on how many backups of a file are currently on disk. You might configure a redundancy level of 3:

```
RMAN> CONFIGURE RETENTION POLICY TO REDUNDANCY 3;
```

In this case, RMAN keeps three backups of each database file, and all redo logs required to recover all retained datafile backups to the current time. Any older backups will be considered obsolete.

Assume, for instance, that you make backups of a datafile every day, starting on a Monday. On Thursday, you make your fourth backup of the datafile, and the backup from Monday becomes obsolete because you have the backups from

Tuesday, Wednesday and Thursday. On Friday, the backup from Tuesday becomes obsolete, because you have the backups from Wednesday, Thursday and Friday.

Archiving Older Backups

There are several reasons to keep older backups of datafiles and archived logs:

- An older backup of datafiles and archived logs is necessary for performing point-in-time recovery to a time before your most recent backup.
- If your most recent backup is corrupt, you can still recover your database using an older backup and the complete set of archived logs since that older backup.
- You may want to keep a copy of the database for archival purposes.

To perform point-in-time recovery to a given target time earlier than your current point of recoverability, then you need a database backup that completed before the target time, as well as all of the archived logs created between the time the backup was started and the target time. For example, if you take full database backups starting at 1:00 AM on February 1 (at SCN 10000) and on February 14 (at SCN 20000), and if you decide on February 28 to use point-in-time recovery to bring your database to its state at 9:00AM February 7 (SCN 13500), then you must use the February 1 backup, plus all redo logs containing changes from between the beginning of the creation of the backup (SCN 10000) and 9:00AM February 7 (SCN 13500).

Note that point-in-time recovery to a time between backups is not an option for a database operating in `NOARCHIVELOG` mode. You can only restore your entire database from a consistent whole database backup, and re-open the database as of the time of that backup. You will lose all changes since the backup was taken.

Determining Backup Frequency

Frequent backups are essential for any recovery scheme. Base the frequency of backups on the rate or frequency of database changes such as:

- Addition and deletion of tables
- Insertions and deletions of rows in existing tables
- Updates to data within tables

The more frequently your database is updated, the more often you should perform database backups. The scenario in "Backup Scripts When Blocks Change Frequently" on page 4-29 backs up the database every week.

If database updates are relatively infrequent, then you can make whole database backups infrequently and supplement them with incremental backups (which will be relatively small because few blocks have changed). The scenario in "Backup Scripts When Few Data Blocks Change" on page 4-26 describes how to develop a backup strategy based on a single whole database backup.

See Also:

- "Backing Up to the Flash Recovery Area: Basic Scenarios" on page 4-25
- "Backing Up to the Flash Recovery Area and to Tape: Basic Scenarios" on page 4-32

Performing Backups Before and After You Make Structural Changes

There are times when you will need to take a backup of your database independent of your regular backup schedule. If you make any of the following structural changes, then perform a backup of the appropriate portion of your database immediately before and after completing the following changes:

- Create or drop a tablespace.
- Add or rename a datafile in an existing tablespace.
- Add, rename, or drop an online redo log group or member.

The part of the database that you should back up depends on your archiving mode.

Mode	Action
`ARCHIVELOG`	Make a control file backup (using RMAN or using the SQL `ALTER DATABASE` statement with the `BACKUP CONTROLFILE` option) after a structural alteration. Of course, you can back up other parts of the database as well.
`NOARCHIVELOG`	Make a consistent whole database backup immediately after the modification.

Backing Up Frequently Used Tablespaces

If you run in `ARCHIVELOG` mode, then you can back up an individual tablespace or even a single datafile. You might want to do this for one or more tablespaces that are updated much more often than the rest of your database, as is sometimes the case for the `SYSTEM` tablespace and automatic undo tablespaces.

More frequent backups of heavily-used datafiles can shorten recovery times in some situations. You may have a database where most updates are restricted to a small set of tablespaces. If you take a full database backup each Sunday, then recovery from a media failure affecting the frequently updated tablespaces on Friday requires re-applying large amounts of redo. Daily backups of the frequently-updated tablespaces reduces the amount of redo to apply without requiring a daily full database backup.

See Also: *Oracle Database Administrator's Guide* for information about managing undo tablespaces

Backing Up after NOLOGGING Operations

When a direct path load is performed to populate a database, no redo data is logged for those database changes. You cannot recover these changes after a restore from backup using conventional media recovery. Likewise, when tables and indexes are created as `NOLOGGING`, the database does not log redo data for these objects, which means that you cannot recover these objects from existing backups. Therefore, you should back up your datafiles after operations for which no redo data is logged.

Note: You can use either a full backup of your datafiles or an incremental backup. Either one will capture all changed blocks, including blocks changed by unrecoverable operations.

See Also: *Oracle Database SQL Reference* for information about the `UNRECOVERABLE` option of the `CREATE TABLE ... AS SELECT` and `CREATE INDEX` statements.

Exporting Data for Added Protection and Flexibility

Oracle database import and export utilities are used to export database objects (tables, stored procedures, and so forth) from databases to be stored as files, and re-import objects from those files. An export provides a logical-level snapshot of the exported objects at the time of the export, as a binary file that can be imported back into the source database or some other database. Consider exporting portions or all of a database for supplemental protection and flexibility in a database's backup strategy.

While useful, database exports are not a substitute for whole database backups. They cannot provide the same complete recovery advantages of physical-level backups. For example, you cannot apply archived logs to logical backups in order to update lost changes.

> **See also:** *Oracle Database Utilities* for more details about exporting and importing data for logical backup

Preventing the Backup of Online Redo Logs

Online redo logs, unlike archived logs, should never be backed up. The chief danger associated by having backups of online redo logs is that you may accidentally restore those backups without meaning to, and corrupt your database.

Online redo log backups are also not particularly useful, for the following reasons:

- If your database is in `ARCHIVELOG` mode, then the archiver is already archiving the filled redo logs automatically.
- If your database is in `NOARCHIVELOG` mode, then the only type of physical backups that you can perform are closed, consistent, whole database backups. The files in this type of backup are all consistent and do not need recovery, so the online logs are not useful after a restore from backup.

The best method for protecting the online logs against media failure is to multiplex them, with multiple log members in each group, on different disks attached to different disk controllers.

> **Note:** RMAN does not permit you to back up online redo logs. You must archive a redo log before backing it up.

Keeping Records of the Hardware and Software Configuration of the Server

During the stress of a recovery situation, it is important that you have all necessary information at your disposal. This is especially true if for some reason you need to contact Oracle Support because you run into a problem that you do not understand. You should have the following documentation about the hardware configuration:

- The name, make, and model of the machine that hosts the database
- The version and patch of the operating system
- The number of disks and disk controllers
- The disk capacity and free space

- The names of all datafiles
- The name and version of the media management software (if you use a third-party media manager)

You should also keep the following documentation about the software configuration:

- The name of the database instance (SID)
- The database identifier (DBID)
- The version and patch release of the Oracle database server
- The version and patch release of the networking software
- The method (RMAN or user-managed) and frequency of database backups
- The method of restore and recovery (RMAN or user-managed)

You should keep this information both in electronic and hardcopy form. For example, if you save this information in a text file on the network or in an email message, then if the entire system goes down, you may not have access to this data.

It is especially important to keep a record of the DBID. If you have to restore and recover your database including the loss of the SPFILE and control file, you will need the DBID during the recovery process. See "Basic Database Restore and Recovery Scenarios" on page 5-11 for details on how the DBID is used during recovery.

Validating Your Data Recovery Strategy

Practice backup and recovery techniques in a test environment before and after you move to a production system. In this way, you can measure the thoroughness of your strategies and minimize problems before they occur in a real situation. Performing test recoveries regularly ensures that your archiving, backup, and recovery procedures work. It also helps you stay familiar with recovery procedures, so that you are less likely to make a mistake in a crisis.

If you use RMAN, then run the `DUPLICATE` command to create a test database using backups of your production database. If you perform user-managed backup and recovery, then you can either create a new database, a standby database, or a copy of an existing database by using a combination of operating system and SQL*Plus commands.

See Also: *Oracle Database Backup and Recovery Advanced User's Guide* to learn about RMAN testing methods, troubleshooting SQL*Plus recovery, block media recovery, and RMAN disaster recovery

Validating RMAN Backups: BACKUP VALIDATE and RESTORE VALIDATE

The RMAN `BACKUP VALIDATE` and `RESTORE VALIDATE` commands can be a useful part of your recovery plan testing. `BACKUP VALIDATE` reads all of the specified files but does not produce any output files. All of the data blocks in the input files are validated, exactly as they are when a real backup takes place. `RESTORE VALIDATE` reads all of the backup files that would be needed to restore the specified objects, but the objects are not actually restored to disk. All of the data blocks in the backup files are validated, exactly as they are when a real restore takes place. Just as in a real restore, `RESTORE VALIDATE` automatically chooses which backup files to restore from. For example, the command `RESTORE VALIDATE DATABASE` ensures that, for every file in the database, a valid backup exists, can be read, and contains valid data.

See Also: "Validating RMAN Backups" on page 4-42 for more details on using `BACKUP VALIDATE` and `RESTORE VALIDATE`

3 Setting Up and Configuring Backup and Recovery

This chapter describes how to start RMAN and prepare the RMAN environment for implementation of your backup and recovery strategy.

This chapter includes the following topics:

Starting and Exiting RMAN: Overview

You have the following basic options for starting RMAN:

- Start the RMAN executable at the operating system command line without specifying any connection options, as in this example:

  ```
  % rman
  ```

- Start the RMAN executable at the operating system command line while connecting to a target database and, possibly, a recovery catalog, as in these examples:

  ```
  % rman TARGET /
  % rman TARGET SYS/oracle@trgt NOCATALOG
  % rman TARGET / CATALOG rman/cat@catdb
  ```

If you connect to the target database on the operating system command line, then you can begin executing commands after the RMAN prompt is displayed. If you start RMAN without connecting to the target database, then you must issue `CONNECT TARGET` command at the RMAN prompt.

To quit RMAN and terminate the program, type `EXIT` or `QUIT` at the RMAN prompt. For example:

```
RMAN> EXIT
```

Types of Database Connections

You can connect to the following types of databases.

Database	Connection
Target database	RMAN connects you to the target database, which is the database that you are backing up or recovering, with the `SYSDBA` privilege. If you do not have this privilege, then the connection fails.
Recovery catalog database	This database is optional. By default, RMAN runs in `NOCATALOG` mode.
Auxiliary database	You can connect to a standby database, duplicate database, or auxiliary instance (standby instance or tablespace point-in-time recovery instance).

Authentication for Database Connections

When connecting to a target or auxiliary database, you must have the SYSDBA privilege. You can connect as SYSDBA with a password file or with operating system authentication.

Note: You do not need to specify the SYSDBA option because RMAN uses this option implicitly and automatically.

If the target database uses password files, then you can connect using a password. Use a password file for either local or remote access. You must use a password file if you are connecting remotely as SYSDBA with a net service name.

If you connect to the database using operating system authentication, remember to set the environment variable specifying the Oracle SID. For example, to set the SID to `trgt` at the UNIX command line enter:

```
% ORACLE_SID=trgt; export ORACLE_SID
```

A SYSDBA privilege is not required when connecting to the recovery catalog. Note that you must grant the RECOVERY_CATALOG_OWNER role to the schema owner.

See Also: *Oracle Database Administrator's Guide* to learn how to authenticate users on a database

Setting Globalization Support Environment Variables for RMAN

Before invoking RMAN, set the NLS_DATE_FORMAT and NLS_LANG environment variables. These variables determine the format used for the time parameters in RMAN commands such as RESTORE, RECOVER, and REPORT.

The following example shows typical language and date format settings:

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

If you are going to use RMAN to connect to an unmounted database and mount the database later while RMAN is still connected, then set the NLS_LANG environment variable so that it also specifies the character set used by the database.

A database that is not mounted assumes the default character set, which is US7ASCII. If your character set is different from the default, then RMAN returns errors after the database is mounted. For example, if the character set is WE8DEC, you can set the NLS_LANG parameter as follows:

```
NLS_LANG=american_america.we8dec
```

NLS_LANG and NLS_DATE_FORMAT must be set for NLS_DATE_FORMAT to be used.

See Also:

- *Oracle Database Reference* for more information about the NLS_ LANG and NLS_DATE_FORMAT parameters
- *Oracle Database Globalization Support Guide*

Connecting to the Target Database

In the examples in this section, the generic values used have the following meanings.

Variable	Meaning
SYS	User with SYSDBA privileges
oracle	The password for connecting as SYSDBA specified in the target database's orapwd file
trgt	The net service name for the target database

Connecting to the Target Database from the Command Line

To connect to the target database from the operating system command line, enter the connection as in the following examples:

```
# example of operating system authentication
% rman TARGET / NOCATALOG
# example of Oracle Net authentication
% rman TARGET SYS/oracle@trgt NOCATALOG
```

You can also start RMAN without specifying NOCATALOG or CATALOG as follows:

```
# example of operating system authentication
% rman TARGET /
# example of Oracle Net authentication
% rman TARGET SYS/oracle@trgt
```

If you do not specify NOCATALOG on the command line, and if you do not specify CONNECT CATALOG after RMAN has started, then RMAN connects in NOCATALOG mode by default the first time that you run a command that requires a repository. For example:

```
% rman TARGET /
RMAN> BACKUP DATABASE;    # RMAN defaults to NOCATALOG mode
```

Connecting to the Target Database from the RMAN Prompt

Alternatively, start RMAN and connect to the target database from the RMAN prompt, as in this example:

```
% rman NOCATALOG
RMAN> CONNECT TARGET /
```

This example connects to the target database with a password file:

```
% rman NOCATALOG
RMAN> CONNECT TARGET SYS/oracle@trgt
```

Setting Up a Database for RMAN Backup

Backing up a database with RMAN is meant to be simple. Therefore, for most of the parameters required for RMAN backup, there are sensible defaults which will let you do basic backup and recovery without extensive setup or configuration.

However, when implementing an RMAN-based backup strategy, you can use RMAN more effectively if you understand the more common options available to you. Many of these can be set in the RMAN environment on a persistent basis, so that you do not have to specify the same options every time you issue a command.

The following discussion presents how RMAN's default behaviors can be changed for your backup and recovery environment and strategies, and introduces the major settings available to you and their most common possible values.

This section includes the following topics:

- Persistent Configuration Settings: Controlling RMAN Behavior
- Configuring the Default Backup Type for Disk Backups
- Configuring Disk Devices and Channels
- Configuring Tape Devices and Channels
- Configuring Compressed Backupsets as Default for Tape or Disk

- Configuring Control File and Server Parameter File Autobackup

Persistent Configuration Settings: Controlling RMAN Behavior

To simplify ongoing use of RMAN for backup and recovery, the RMAN lets you set a number of persistent configuration settings for each target database. These settings control many aspects of RMAN's behavior when working with that database, such as backup retention policy, default destinations for backups to tape or disk, default backup device type (tape or disk), and so on.

The default values for these configuration settings let you use RMAN effectively without changing any of them. However, as you develop a more advanced backup and recovery strategy, you will have to change these settings to implement that strategy. Use the RMAN `SHOW` and `CONFIGURE` commands to view and change the RMAN configuration settings.

> **See Also:** *Oracle Database Recovery Manager Reference* for `CONFIGURE` syntax

Displaying Current RMAN Configuration Settings: SHOW ALL

The `SHOW ALL` command displays the current settings of all parameters that you can set with the `CONFIGURE` command, including both those that you have altered and those which are still set to RMAN's default setting.

To show all RMAN configuration settings:

After connecting to the target database and recovery catalog (if you use one), run the RMAN `SHOW ALL` command. For example, enter the following:

```
RMAN> SHOW ALL;    # shows all CONFIGURE settings, both user-entered and default
```

Sample output for `SHOW ALL` follows:

```
using target database controlfile instead of recovery catalog
RMAN configuration parameters are:
CONFIGURE RETENTION POLICY TO REDUNDANCY 1;
CONFIGURE BACKUP OPTIMIZATION OFF; # default
CONFIGURE DEFAULT DEVICE TYPE TO 'SBT_TAPE';
CONFIGURE CONTROLFILE AUTOBACKUP ON;
CONFIGURE CONTROLFILE AUTOBACKUP FORMAT FOR DEVICE TYPE DISK TO
'/mydisk/oracle/dbs/%F';
CONFIGURE CONTROLFILE AUTOBACKUP FORMAT FOR DEVICE TYPE SBT_TAPE TO '%F'; #
default
CONFIGURE DEVICE TYPE 'SBT_TAPE' PARALLELISM 1 BACKUP TYPE TO BACKUPSET;
CONFIGURE DEVICE TYPE DISK BACKUP TYPE TO COPY PARALLELISM 1;
CONFIGURE DATAFILE BACKUP COPIES FOR DEVICE TYPE SBT_TAPE TO 1; # default
CONFIGURE DATAFILE BACKUP COPIES FOR DEVICE TYPE DISK TO 1; # default
CONFIGURE ARCHIVELOG BACKUP COPIES FOR DEVICE TYPE SBT_TAPE TO 1; # default
CONFIGURE ARCHIVELOG BACKUP COPIES FOR DEVICE TYPE DISK TO 1; # default
CONFIGURE CHANNEL DEVICE TYPE 'SBT_TAPE' PARMS  'SBT_LIBRARY=myvendor.mysbt';
CONFIGURE MAXSETSIZE TO UNLIMITED; # default
CONFIGURE ARCHIVELOG DELETION POLICY TO NONE; # default
CONFIGURE SNAPSHOT CONTROLFILE NAME TO '/mydisk/oracle/dbs/cf_snap.f';
```

The configuration is displayed as the series of RMAN commands required to re-create the configuration. You can paste the output of SHOW ALL into a command file so that you can later re-create the entire configuration. You can even run the script on a different target database, to re-create the same RMAN configuration on multiple databases.

Restoring Default RMAN Configuration Settings: CONFIGURE... CLEAR

You can return any setting you change with the CONFIGURE command to its default value by running the CONFIGURE command with the CLEAR option, as in:

```
CONFIGURE BACKUP OPTIMIZATION CLEAR;
ONFIGURE RETENTION POLICY CLEAR;
CONFIGURE CONTROLFILE AUTOBACKUP FORMAT FOR DEVICE TYPE DISK CLEAR;
```

Configuring the Default Device Type for Backups

By default, RMAN sends all backups to an operating system specific directory on disk. You can also configure RMAN to make backups to media such as tape.

After configuring an sbt (that is, tape or media management) device according to the instructions in your media management vendor documentation, you can make the media manager the default device:

```
CONFIGURE DEFAULT DEVICE TYPE TO sbt;
```

To configure disk as the default device for backups, you can either use CONFIGURE... CLEAR to restore the default setting, or explicitly configure the default device as shown:

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

Configuring the Default Backup Type for Disk Backups

You can configure backup sets or image copies as the default, using either of the following commands:

```
RMAN> CONFIGURE DEVICE TYPE DISK BACKUP TYPE TO COPY; # image copies
RMAN> CONFIGURE DEVICE TYPE DISK BACKUP TYPE TO BACKUPSET; # uncompressed
```

The default for backups to disk, as with tape, is backup set. Note that there is no analogous option for media manager devices, because RMAN can only write backups to media manager devices as backup sets.

Configuring Compressed Backupsets as Default for Tape or Disk

You can configure RMAN to use compressed backupsets by default on a particular device type, by using the `CONFIGURE DEVICE TYPE` command with the `BACKUP TYPE TO COMPRESSED BACKUPSET` option, as shown in the following examples.

The following commands let you set the default backup type to compressed backup sets for different device types:

```
RMAN> CONFIGURE DEVICE TYPE DISK BACKUP TYPE TO COMPRESSED BACKUPSET;
RMAN> CONFIGURE DEVICE TYPE sbt BACKUP TYPE TO COMPRESSED BACKUPSET;
```

To disable compression, use the `CONFIGURE DEVICE TYPE` command with arguments specifying your other desired settings, but omitting the `COMPRESSED` keyword.

Configuring Disk Devices and Channels

RMAN **channels** (connections to server sessions on the target database) perform all RMAN tasks. By default, RMAN allocates one disk channel for all operations.

The following command configures RMAN to write disk backups to the `/backup` directory (refer to "Backing Up Database Files and Archived Logs with RMAN" on page 4-5).:

```
CONFIGURE CHANNEL DEVICE TYPE DISK FORMAT '/backup/ora_df%t_s%s_s%p';
```

The format specifier `%t` is replaced with a four byte time stamp, `%s` with the backup set number, and `%p` with the backup piece number.

You can also configure an Automatic Storage Management disk group as your destination, as in the following example:

```
CONFIGURE CHANNEL DEVICE TYPE DISK FORMAT '+dgroup1';
```

Note that by configuring an explicit format for disk channels, you direct backups away from the flash recovery area, if you have configured one.

Configuring Tape Devices and Channels

Some media managers require configuration settings that you pass in by including a `PARMS` string in the `CONFIGURE` command, as follows:

```
CONFIGURE CHANNEL DEVICE TYPE sbt PARMS='ENV=mml_env_settings';
```

See the media manager documentation for details.

The following command configures two `sbt` channels for use in RMAN jobs:

```
CONFIGURE DEVICE TYPE sbt PARALLELISM 2;
```

Configuring Control File and Server Parameter File Autobackup

RMAN can be configured to automatically back up the control file and server parameter file whenever the database structure metadata in the control file changes and whenever a backup record is added. The autobackup enables RMAN to recover the database even if the current control file, catalog, and server parameter file are lost.

Because the filename for the autobackup uses a well-known format, RMAN can search for it without access to a repository, and then restore the server parameter file. After you have started the instance with the restored server parameter file, RMAN can restore the control file from an autobackup. After you mount the control file, the RMAN repository is available and RMAN can restore the datafiles and find the archived redo log.

You can enable the autobackup feature by running this command:

```
CONFIGURE CONTROLFILE AUTOBACKUP ON;
```

You can disable the feature by running this command:

```
CONFIGURE CONTROLFILE AUTOBACKUP OFF;
```

See Also:

- *Oracle Database Backup and Recovery Advanced User's Guide* for a conceptual overview of control file autobackups
- *Oracle Database Recovery Manager Reference* for `CONFIGURE` syntax

Configuring the Control File Autobackup Format

By default, the format of the autobackup file for all configured devices is the substitution variable `%F`. This variable format translates into *`c-IIIIIIIIII-YYYYMMDD-QQ`*, where:

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

You can change the default format by using the following command, where *`deviceSpecifier`* is any valid device such as `DISK` or `sbt`, and `'`*`string`*`'` must contain the substitution variable `%F` (and no other substitution variables) and is a valid handle for the specified device:

```
CONFIGURE CONTROLFILE AUTOBACKUP FORMAT
  FOR DEVICE TYPE deviceSpecifier TO 'string';
```

For example, you can run the following command:

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

The following example configures the autobackup to write to an Automatic Storage Management disk group:

```
CONFIGURE CONTROLFILE AUTOBACKUP
  FOR DEVICE TYPE DISK TO '+dgroup1';
```

To clear control file autobackup formats for a device, use the following commands:

```
CONFIGURE CONTROLFILE AUTOBACKUP FOR DEVICE TYPE DISK CLEAR;
CONFIGURE CONTROLFILE AUTOBACKUP FOR DEVICE TYPE sbt CLEAR;
```

If you have set up a flash recovery area for your database, you can direct control file autobackups to the flash recovery area by clearing the control file autobackup format for disk.

Overriding the Configured Control File Autobackup Format

The `SET CONTROLFILE AUTOBACKUP FORMAT` command, which you can specify either within a `RUN` block or at the RMAN prompt, overrides the configured autobackup format in the current session only.

You could use this command in either of the following ways:

```
RMAN> SET CONTROLFILE AUTOBACKUP FORMAT FOR DEVICE TYPE sbt TO 'controlfile_%F';
RMAN> BACKUP AS COPY DATABASE;
RMAN> RUN {
       SET CONTROLFILE AUTOBACKUP FORMAT FOR DEVICE TYPE DISK TO '/tmp/%F.bck';
       BACKUP AS BACKUPSET DEVICE TYPE DISK DATABASE;
      }
```

The first `SET CONTROLFILE AUTOBACKUP FORMAT` controls the name of the control file autobackup until the RMAN client exits, overriding any configured control file autobackup format. The `SET CONTROFILE AUTOBACKUP FORMAT` in the RUN block overrides the `SET CONTROLFILE AUTOBACKUP FORMAT` outside the RUN block for the duration of the RUN block.

The order of precedence is:

1. `SET CONTROLFILE AUTOBACKUP FORMAT` (within a `RUN` block)
2. `SET CONTROLFILE AUTOBACKUP FORMAT` (at `RMAN` prompt)
3. `CONFIGURE CONTROLFILE AUTOBACKUP FORMAT`

Setting Up a Flash Recovery Area for RMAN

As explained in "Automatic Disk-Based Backup and Recovery: The Flash Recovery Area" on page 1-16, the flash recovery area feature lets you set up a location on disk where the database can create and manage a variety of backup and recovery-related files on your behalf.

Using a flash recovery area simplifies the ongoing administration of your database by automatically naming files, retaining them as long as they are needed for restore and recovery activities, and deleting them when they are no longer needed to restore your database and space is needed for some other backup and recovery-related purpose.

Your long-term backup and recovery administration can be greatly simplified by using a flash recovery area. Use of the flash recovery area is strongly recommended. You may want to set up a flash recovery area as one of the first steps in implemeting your backup strategy.

This section outlines the functions of the flash recovery area, identifies the files stored there, explains the rules for how files are managed there, and introduces the most important configuration options.

Choosing a Location for the Flash Recovery Area

When setting up a flash recovery area, you must choose a location (a directory or Automatic Storage Management disk group) to hold the files. The flash recovery area cannot be stored on a raw file system.

You must also determine a **disk quota** for the flash recovery area, the maximum space to be used for all files stored there. You must choose a location large enough to accomodate the required disk quota. When the disk space limit is approached, Oracle can delete nonessential files to make room for new files, subject to the limitations of the retention policy.

The flash recovery area should be on a separate disk from the **database area**, where active database files such as datafiles, control files, and online redo logs are stored. Keeping the flash recovery area on the same disk as the database area exposes you to loss of both your live database files and backups in the event of a media failure.

Note: There are special considerations for choosing a location for the flash recovery area in a RAC environment. The location must be on a cluster file system, ASM or a shared directory configured through NFS. The location and disk quota must be the same on all instances.

Flash Recovery Area, Automatic Storage Management, and Oracle Managed Files

The flash recovery area is closely related to and can be used in conjunction with two other Oracle features: Oracle Managed Files and Automatic Storage Management.

Oracle Managed Files (OMF) is a service that automates naming, location, creation and deletion of database files such as control files, redo log files, datafiles and others, based on a few initialization parameters. It can simplify many aspects of the DBA's work by eliminating the need to devise your own policies for such details.

The flash recovery area is built on top of OMF, so the flash recovery area can be stored anywhere Oracle-managed files can. Oracle Managed Files can be used on top of a traditional file system supported by the host operating system (for example, VxFS or ODM).

The flash recovery area can also be used with Oracle's Automatic Storage Management (ASM). ASM consolidates storage devices into easily managed disk groups and provides benefits such as mirroring and striping without requiring a third-party logical volume manager.

Even if you choose not to use a flash recovery area, you can still use Oracle Managed Files to manage your backup files in an ASM disk group. (On a regular file system, OMF can only be used by using the flash recovery area feature.) You will lose one of the major benefits of the flash recovery area, the automatic deletion of files no longer needed to meet your recoverability goals. However, the other automatic features of OMF will still function.

Note: When storing backup files, using OMF on top of Automatic Storage Management without using a flash recovery area is supported but discouraged. It is awkward to directly manipulate files under Automatic Storage Management.

See also: *Oracle Database Administrator's Guide* for more information on Automatic Storage Management and Oracle Managed Files

Files That Can Be Stored in the Flash Recovery Area

The files in recovery area can be classified as **permanent** or **transient**. The only permanent files (assuming these are configured to be stored in the recovery area) are multiplexed copies of the current control file and online redo logs. These cannot be deleted without causing the instance to fail. All other files are transient, because Oracle will generally eventually delete these files, at some point after they become obsolete under the retention policy or have been backed up to tape. Transient files include archived redo logs, datafile copies, control file copies, control file autobackups, and backup pieces.

Note: The Oracle Flashback Database feature, which provides an convenient alternative to point-in-time recovery, generates flashback logs, which are also considered transient files and must be stored in the flash recovery area. See *Oracle Database Backup and Recovery Advanced User's Guide* for more details about Oracle Flashback Database.

Planning the Size of the Flash Recovery Area

The larger the flash recovery area is, the more useful it becomes. Ideally, the flash recovery area should be large enough to contain all of the following files:

- A copy of all datafiles

- Incremental backups, as used by your chosen backup strategy
- Online redo logs
- Archived redo logs not yet backed up to tape
- Control files
- Control file autobackups (which include copies of the control file and SPFILE)

If providing this much space is impractical, it is best to create an area large enough to keep a backup of the most important tablespaces and all the archived logs not yet copied to tape. At an absolute minimum, the flash recovery area must be large enough to contain the archived logs that have not been copied to tape.

Note: If you wish to use the Oracle Flashback Database feature, you must allow for storage required for flashback logs. See *Oracle Database Backup and Recovery Advanced User's Guide* for more details.

To determine the disk quota and current disk usage in the flash recovery area, query the view `V$RECOVERY_FILE_DEST`.

Formulas for estimating a useful flash recovery area size depend upon several factors in your backup and recovery strategy: whether few or many datafile blocks change frequently, whether you are backing up only to disk or both to disk and tape, and whether you use a redundancy-based retention policy or a recovery window-based retention policy. Specific formulas are provided for many different backup scenarios in "Backing Up to the Flash Recovery Area: Basic Scenarios" on page 4-25 and "Backing Up to the Flash Recovery Area and to Tape: Basic Scenarios" on page 4-32.

Setting Initialization Parameters for the Flash Recovery Area

To enable the flash recovery area, you must set the two initialization parameters `DB_RECOVERY_FILE_DEST_SIZE` (which specifies the disk quota, or maximum space to use for flash recovery area files for this database) and `DB_RECOVERY_FILE_DEST` (which specifies the location of the flash recovery area).

Note: .

- `DB_RECOVERY_FILE_DEST_SIZE` must be set *before* `DB_RECOVERY_FILE_DEST.`
- In a RAC database, all instances must have the same values for these parameters.

Initialization parameters can be specified by any of the following means:

- Include them initialization parameter file of the target database
- Specify them with the SQL statement `ALTER SYSTEM SET`
- Use the Database Configuration Assistant to set them

See Also:

- *Oracle Database Administrator's Guide* for details on setting and changing database initialization parameters

To find out the current flash recovery area location, query `V$RECOVERY_FILE_DEST`.

See Also: *Oracle Database SQL Reference* for `ALTER SYSTEM` syntax

There are also restrictions on some other initialization parameters when using a flash recovery area.

The DB_RECOVERY_FILE_DEST_SIZE Initialization Parameter

This parameter specifies the maximum storage in bytes of data to be used by the flash recovery area for this database. Note, however, that this value does not include certain kinds of disk overhead:

- Block 0 or the OS block header of each Oracle file is not included in this size. Allow an extra 10% for this data when computing the actual disk usage required for the flash recovery area.
- `DB_RECOVERY_FILE_DEST_SIZE` does not indicate the real size occupied on disk when the underlying filesystem is mirrored, compressed, or in some other way affected by overhead not known to Oracle. For example, if you can

configure the flash recovery area on a normal redundancy (2-way mirrored) ASM disk group, each file of X bytes occupies 2X bytes on the ASM disk group. In such a case, `DB_RECOVERY_FILE_DEST_SIZE` must be set to no more than 1/2 the size of the disks for the ASM disk group. Likewise, when using a high redundancy (three-way mirrored) ASM disk group, `DB_RECOVERY_FILE_DEST_SIZE` must be no more than 1/3 the size of the disks in the disk group, and so on.

The DB_RECOVERY_FILE_DEST Initialization Parameter

This parameter specifies a valid disk location for file creation, which can be a directory on a file system, or Automatic Storage Management disk group.

Sharing a Flash Recovery Area Among Multiple Databases Mutliple database can have the same value for `DB_RECOVERY_FILE_DEST`, if one of the following conditions is met:

- No two databases for which `DB_UNIQUE_NAME` is specified have the same value for `DB_UNIQUE_NAME`.
- For those databases where no `DB_UNIQUE_NAME` is provided, no two databases have the same value for `DB_NAME`.

For example, you might want to set up primary and standby databases to share the same `DB_RECOVERY_FILE_DEST`. These two databases will have the same `DB_NAME` but one of them must have a `DB_UNIQUE_NAME` different from `DB_NAME`.

When databases share a single flash recovery area in this fashion, the file system or ASM disk group holding the flash recovery area should be large enough to hold all of the recovery files for all of the databases. Add all the values for `DB_RECOVERY_FILE_DEST_SIZE` for the different databases, and then allow for any overhead such as mirroring or compression in allocating actual disk space, as described in "The DB_RECOVERY_FILE_DEST_SIZE Initialization Parameter" on page 3-15.

Restrictions on Initialization Parameters When Using Flash Recovery Area

Using a flash recovery area has implications for some other initialization parameters:

- You cannot use the `LOG_ARCHIVE_DEST` and `LOG_ARCHIVE_DUPLEX_DEST` parameters to specify redo log archive destinations. You must instead use the newer `LOG_ARCHIVE_DEST_`*n* parameters. See *Oracle Database Reference* for details on the semantics of the `LOG_ARCHIVE_DEST_`*n* parameters.

- LOG_ARCHIVE_DEST_10 is implicitly set to USE_DB_RECOVERY_FILE_DEST (meaning that archived redo log files will be sent to the flash recovery area) if you create a recovery area and do not set any other local archiving destinations.
- Oracle Corporation recommends that DB_RECOVERY_FILE_DEST not be the same as DB_CREATE_FILE_DEST or any of the DB_CREATE_ONLINE_LOG_DEST_*n* parameters. A warning will appear in the alert log if DB_RECOVERY_FILE_DEST is the same as any of the other parameters listed here.

Adding a Flash Recovery Area to an Existing Database

To create a flash recovery area, you can set the necessary parameters in the initialization parameter file (PFILE) and restart the database. You can also use the following steps to add a flash recovery area to an open database.

1. After you start SQL*Plus and connect to the database, set the size of the flash recovery area. For example, set it to 10 GB:

   ```
   SQL> ALTER SYSTEM SET DB_RECOVERY_FILE_DEST_SIZE = 10G SCOPE=BOTH SID='*';
   ```

   Set SCOPE to BOTH make the change both in memory and the server parameter file. Setting SID to "*" has no effect in a single-instance database; in a RAC database it causes the change to take effect across all instances.

2. Set the location of the flash recovery area. For example, if the location is the file system directory /dir1, then you can do the following:

   ```
   SQL> ALTER SYSTEM SET DB_RECOVERY_FILE_DEST = '/dir1' SCOPE=BOTH SID='*';
   ```

   If the flash recovery area location is an Automatic Storage Management disk group named disk1, for example, then you can do the following:

   ```
   SQL> ALTER SYSTEM SET DB_RECOVERY_FILE_DEST = '+disk1' SCOPE=BOTH SID='*';
   ```

Using the V$RECOVERY_FILE_DEST View

You can query the V$RECOVERY_FILE_DEST view to find out the current location, disk quota, space in use, space reclaimable by deleting files, and total number of files in the flash recovery area.

```
SQL> SELECT * FROM V$RECOVERY_FILE_DEST;
NAME            SPACE_LIMIT SPACE_USED SPACE_RECLAIMABLE NUMBER_OF_FILES
--------------  ----------- ---------- ----------------- ---------------
/mydisk/rcva     5368709120 109240320             256000              28
```

This view can help you determine whether you have allocated enough space for your flash recovery area.

Disabling the Flash Recovery Area

To disable the flash recovery area, set the `DB_RECOVERY_FILE_DEST` initialzation parameter to a null string. For example, use this SQL*Plus statement to change the parameter on a running database:

```
ALTER SYSTEM SET DB_RECOVERY_FILE_DEST='' SCOPE=BTOTH SID="*";
```

The database will no longer provide the space management features of the flash recovery area for the files stored in the old `DB_RECOVERY_FILE_DEST` location. The files will still be known to the RMAN repository, however, and available for backup and restore activities.

Configuring the Backup Retention Policy

The backup retention policy specifies which backups must be retained to meet your data recovery requirements. This policy can be based on a recovery window (the maximum number of days into the past for which you can recover) or redundancy (how many copies of each backed-up file to keep).

Use the `CONFIGURE` command to set the retention policy.

See Also:

- *Oracle Database Recovery Manager Reference* for `CONFIGURE` syntax

Configuring a Recovery Window-Based Retention Policy

The `RECOVERY WINDOW` parameter of the `CONFIGURE` command specifies the number of days between the current time and the earliest point of recoverability. RMAN does not consider any full or level 0 incremental backup as obsolete if it falls within the recovery window. Additionally, RMAN retains all archived logs and level 1 incremental backups that are needed to recover to a random point within the window.

Run the `CONFIGURE RETENTION POLICY` command at the RMAN prompt. This example ensures that you can recover the database to any point within the last week:

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

RMAN does not automatically delete backups rendered obsolete by the recovery window. Instead, RMAN shows them as OBSOLETE in the REPORT OBSOLETE output and in the OBSOLETE column of V$BACKUP_FILES. RMAN deletes obsolete files if you run the DELETE OBSOLETE command.

Configuring a Redundancy-Based Retention Policy

The REDUNDANCY parameter of the CONFIGURE RETENTION POLICY command specifies how many backups of each datafile and control file that RMAN should keep. In other words, if the number of backups for a specific datafile or control file exceeds the REDUNDANCY setting, then RMAN considers the extra backups as obsolete. The default retention policy is REDUNDANCY=1.

As you produce more backups, RMAN keeps track of which ones to retain and which are obsolete. RMAN retains all archived logs and incremental backups that are needed to recover the nonobsolete backups.

Assume that you make a backup of datafile 7 on Monday, Tuesday, Wednesday, and Thursday. You now have four backups of the datafile. If REDUNDANCY is 2, then the Monday and Tuesday backups are obsolete. If you make another backup on Friday, then the Wednesday backup becomes obsolete.

Run the CONFIGURE RETENTION POLICY command at the RMAN prompt, as in the following example:

```
CONFIGURE RETENTION POLICY TO REDUNDANCY 3;
```

Showing the Current Retention Policy

You can view the currently configured retention policy with the SHOW RETENTION POLICY command. Sample output follows:

```
RMAN> SHOW RETENTION POLICY;

RMAN configuration parameters are:
CONFIGURE RETENTION POLICY TO REDUNDANCY 3;
```

Disabling the Retention Policy

When you disable the retention policy, RMAN does not consider any backup as obsolete.

To disable the retention policy, run this command:

```
CONFIGURE RETENTION POLICY TO NONE;
```

Configuring the retention policy to `NONE` is not the same as clearing it. Clearing it returns it to its default setting of `REDUNDANCY=1`, whereas `NONE` disables it completely.

If you disable the retention policy and run `REPORT OBSOLETE` or `DELETE OBSOLETE` without passing a retention policy option to the command, RMAN issues an error because no retention policy exists to determine which backups are obsolete.

Note: If you are using a flash recovery area, then you should not run your database with the retention policy disabled. If files are never considered obsolete, then a file can only be deleted from the flash recovery area if it has been backed up to some other disk location or to a tertiary storage device such as tape. It is quite likely that all of the space in your recovery area will be used. This interferes with the normal operation of your database as described in "When Space is Not Available in the Flash Recovery Area" on page 3-21.

How Oracle Manages Disk Space in the Flash Recovery Area

Oracle does not delete eligible files from the flash recovery area until the space must be reclaimed for some other purpose. The effect is that files recently moved to tape are often still available on disk for use in recovery. The recovery area can thus serve as a kind of cache for tape. Once the flash recovery area is full, Oracle automatically deletes eligible files to reclaim space in the flash recovery area as needed.

When Files are Eligible for Deletion from the Flash Recovery Area

There are relatively simple rules governing when files become eligible for deleteion from the flash recovery area:

- Permanent files are never eligible for deletion.
- Files that are obsolete under the configured retention policy are eligible for deletion.
- Transient files that have been copied to tape are eligible for deletion.
- In a Data Guard environment, archived redo log deletion policy governs when archived redo log files can be deleted from the flash recovery area. See *Oracle*

Data Guard Concepts and Administration for details on archived redo log deletion policy.

Note: Exactly which of the eligible files will be deleted to satisfy a space request is unpredictable. The rules are likely to change between releases and are quite dependent upon your configuration. The safe and reliable way to control deletion of files from the flash recovery area is to change your retention policy. If you wish to increase the likelihood that files moved to tape are also still on disk, increase the flash recovery area quota.

When Space is Not Available in the Flash Recovery Area

If, for instance, the RMAN retention policy requires keeping a set of backups larger than the flash recovery area disk quota, or if the retention policy is set to NONE, then the flash recovery area can fill completely with no reclaimable space.

The database issues a warning alert when reclaimable space is less than 15% and a critical alert when reclaimable space is less than 3%. To warn the DBA of this condition, an entry is added to the alert log and to the `DBA_OUTSTANDING_ALERTS` table (used by Enterprise Manager). However, the database continues to consume space in the flash recovery area until there is no reclaimable space left.

When the recovery area is completely full, the error you will receive is:

```
ORA-19809: limit exceeded for recovery files
ORA-19804: cannot reclaim nnnnn bytes disk space from mmmmm limit
```

where *nnnnn* is the number of bytes required and *mmmm* is the disk quota for the flash recovery area.

The database handles a flash recovery area with insufficient reclaimable space just as it handles a disk full condition. Often, the result is an database hang, but not always. For example, if the flash recovery area is one of your mandatory redo log archiving destinations, and the database cannot archive a new log because the recovery area is full, then the archiver may, depending on your configuration, retry archiving periodically until space is freed in the recovery area. For information on how a particular feature of Oracle responds to a disk full condition, see the documentation for that feature.

See Also:

- "Resolving a Full Flash Recovery Area" on page 6-22 for more on how to address situations where the flash recovery area frequently fills to capacity and there are no files eligible for deletion

Configure Flash Recovery Area for Disk-Based Backups: Example

In this example the database is configured to store archived logs and RMAN backups to the flash recovery area. The control file and online redo log copies will still be stored in the conventional file system. The datafiles are assumed to be 3GB in size, and the retained archived redo logs should be no larger than 4GB. The backup strategy will be based on incremental backups. The control file will be automatically backed up to the flash recovery area.

The flash recovery area will be sized to 10GB, room enough for control file autobackups, a whole database level 0 incremental backup (which consists of image copies of the 3GB of datafiles), plus several incremental level 1 backups.

The parameter file should contain the following entries:

```
DB_NAME=sample
# set location for current datafiles:
DB_CREATE_FILE_DEST = '/u02/oradata/wrk_area'
# set location for control files and online redo logs:
DB_CREATE_ONLINE_LOG_DEST_1 = '/u03/oradata/wrk_area'
DB_CREATE_ONLINE_LOG_DEST_2 = '/u04/oradata/wrk_area'
# set flash recovery area location and size
DB_RECOVERY_FILE_DEST = '/u01/oradata/rcv_area'
DB_RECOVERY_FILE_DEST_SIZE = 10G
```

Because the parameter file does not set `LOG_ARCHIVE_DEST_`*n*, Oracle sends archived logs to the flash recovery area only.

Once the target database is started, the following RMAN commands configure the retention policy, backup optimization, and the control file autobackup:

```
RMAN> CONNECT TARGET SYS/orace@trgt;
RMAN> CONFIGURE RETENTION POLICY TO REDUNDANCY 1;
RMAN> CONFIGURE BACKUP OPTIMIZATION ON;
RMAN> CONFIGURE CONTROLFILE AUTOBACKUP ON;
```

You could now run a series of backup jobs to take backups to your flash recovery area.

> **See also:** "Scripting Disk-Only Backups" on page 4-26 for examples of backup jobs you could run in this environment.

Creating a Database with Multiplexed Files in the Flash Recovery Area: Scenario

Assume that you want to create a database in which the control files, datafiles, and online redo logs are Oracle managed files in a single file system directory. Additionally, you want to do the following:

- Maintain a multiplexed copy of the control file in the flash recovery area
- Maintain multiplexed copies of the online redo logs in the flash recovery area
- Archive one copy of each redo log to a file system location outside the work area and flash recovery area
- Archive one copy of each redo log to the flash recovery area
- Send RMAN backups to the flash recovery area by default

> **See Also:** *Oracle Database Backup and Recovery Advanced User's Guide* for more detailed information about file creation in the flash recovery area

To create a database with a flash recovery area:

1. Set the initialization parameters. Assume that you set the following:

```
# set DB_NAME
DB_NAME=sample
# set destination for OMF datafiles, control file and online redo logs
DB_CREATE_FILE_DEST = /u01/oradata/wrk_area/
# set log archiving destinations to a file system location
# and the flash recovery area
LOG_ARCHIVE_DEST_1 = 'LOCATION=/arc_dest1'
LOG_ARCHIVE_DEST_2 = 'LOCATION=USE_DB_RECOVERY_FILE_DEST'
# multiplexed copies of control file and online logs in flash recovery area
# rman backups also go here
DB_RECOVERY_FILE_DEST = 'LOCATION=/u01/oradata/rcv_area'
DB_RECOVERY_FILE_DEST_SIZE = 10G
```

The DB_CREATE_FILE_DEST parameter sets the default directory for all datafiles, online logs, and control files. Another copy of the control file and online logs is created in the flash recovery area.

2. After you set the initialization parameters, create the database. For example, start SQL*Plus and enter:

   ```
   SQL> CREATE DATABASE sample;
   ```

   The preceding statement creates Oracle managed datafiles in DB_CREATE_FILE_DEST. Because you did not specify a LOGFILE clause, this statement creates two online log groups. Each log group has two members, with one member in DB_CREATE_FILE_DEST and another in DB_RECOVERY_FILE_DEST. Because the CONTROL_FILES parameter was not set, Oracle creates a control file in DB_CREATE_FILE_DEST (primary) and DB_RECOVERY_FILE_DEST (multiplexed copy). On a Solaris system, the filenames might look like the following:

```
/u02/oradata/wrk_area/SAMPLE/datafile/o1_mf__system_cmr7t30p_.dbf # OMF datafile
/u02/oradata/wrk_area/SAMPLE/logfile/o1_mf_0orrm31z_.log   # log grp 1, mem 1
/u02/oradata/wrk_area/SAMPLE/logfile/o1_mf_2_2xyz16am_.log # log grp 2, mem 1
/u02/oradata/wrk_area/SAMPLE/controlfile/o1_mf_3ajeikm_.ctl # primary ctl file
/u01/oradata/rcv_area/SAMPLE/logfile/o1_mf_1_ixfvm8w9_.log  # log grp 1 mem 2
/u01/oradata/rcv_area/SAMPLE/logfile/o1_mf_2_q89tmp28_.log  # log grp 2, mem 2
/u01/oradata/rcv_area/SAMPLE/controlfile/o1_mf_6adjkid_.ctl # ctl file copy
```

3. Oracle uses LOG_ARCHIVE_DEST_1 and LOG_ARCHIVE_DEST_2 as destinations for archiving the redo logs. Archived redo log files are created in the flash recovery area because LOG_ARCHIVE_DEST_2 is configured as flash recovery area. Because you enabled a local redo log archiving destination, LOG_ARCHIVE_DEST_10 is not implicitly set to the flash recovery area.

   The archived redo log files in the flash recovery area are given Oracle-managed filenames that are *not* based on the LOG_ARCHIVE_FORMAT parameter. For example, generate an archived log:

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

   On Solaris, the preceding statement creates an archived log in the primary archiving location as well as the following flash recovery area subdirectory: /u01/oradata/rcv_area/SAMPLE/archivelog/*YYYY_MM_DD*

   where *YYYY_MM_DD* is the creation date format.

4. To create a new group of online redo logs, execute the ALTER DATABASE ADD LOGFILE statement. When no file name is specified, it creates another log file member in the flash recovery area. For example, enter the following:

   ```
   ALTER DATABASE ADD LOGFILE;
   ```

   The preceding statement creates another log group with two members: one in DB_CREATE_FILE_DEST and another in DB_RECOVERY_FILE_DEST.

Creating a Database with Only Archived Logs in the Flash Recovery Area: Scenario

Assume that you want to create a database in which the control files, datafiles, and online redo logs are Oracle managed files in a single file system directory. Additionally, you want to do the following:

- Archive each redo log to the flash recovery area (and *only* to the flash recovery area)
- Send RMAN backups to the flash recovery area by default.

> **See Also:** *Oracle Database Backup and Recovery Advanced User's Guide* for more detailed information about file creation in the flash recovery area

To create a database with a flash recovery area:

1. Set the relevant initialization parameters. Assume that you set the following:

   ```
   DB_NAME=sample
   DB_CREATE_FILE_DEST = '/u02/oradata/wrk_area'
   DB_RECOVERY_FILE_DEST = 'location=/u01/oradata/rcv_area'
   DB_RECOVERY_FILE_DEST_SIZE = 10G
   # if you set the following parameters, then the online redo logs *and*
   # current control file are located here
   DB_CREATE_ONLINE_LOG_DEST_1 = '/u03/oradata/wrk_area'
   DB_CREATE_ONLINE_LOG_DEST_2 = '/u04/oradata/wrk_area'
   ```

   The DB_CREATE_FILE_DEST parameter sets the default file system directory for the datafiles. The DB_CREATE_ONLINE_LOG_DEST_*n* parameter sets the default file system directories for the online redo logs and control files. DB_RECOVERY_FILE_DEST sets the file system directory for archived logs.

2. After you set the initialization parameters, create the database. For example, start SQL*Plus and enter:

   ```
   CREATE DATABASE sample;
   ```

No multiplexed copies of the online redo logs or control files are created in the flash recovery area.

3. Because you enabled a flash recovery area, Oracle automatically sets `LOG_ARCHIVE_DEST_10` to the flash recovery area. The filenames in the flash recovery area are given Oracle managed filenames that are *not* based on the `LOG_ARCHIVE_FORMAT` parameter. For example, generate an archived log:

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

   On Solaris, the preceding statement creates an archived log in a flash recovery area subdirectory: `/u01/oradata/rcv_area/SAMPLE/archivelog/`*`YYYY_MM_DD`*, where *`YYYY_MM_DD`* is the creation date format.

4. To create a new group of online redo logs, execute the `ALTER DATABASE ADD LOGFILE` statement. When no file name is specified, it creates another log file member in each `DB_CREATE_ONLINE_LOG_DEST_`*`n`* location (but not the flash recovery area). For example, enter the following:

   ```
   ALTER DATABASE ADD LOGFILE;
   ```

   On Solaris, the preceding statement creates one member in `/u03/oradata/wrk_area/SAMPLE/logfile` and another member in `/u04/oradata/wrk_area/SAMPLE/logfile`. On other platforms, the specific file and directory names are platform-dependent.

4 Making Backups with Recovery Manager

This chapter includes the following topics:

Overview of RMAN Backups

You generate an RMAN backup by running the BACKUP command from within the RMAN client. Often, an RMAN backup can be performed with a very simple command:

```
RMAN> BACKUP DATABASE;
```

Whenever you create a backup with RMAN, RMAN records the action in the RMAN repository. You can also record copies of files you create outside of RMAN (such as copies of datafiles created with host operating system commands) in the repository. When you attempt to restore the backups using the RESTORE command, RMAN queries the repository for information about available backups, then chooses among them to perform the restore efficiently.

RMAN backups can be taken when the database is mounted or open. (The database must at least be mounted, so that RMAN can have the use of the control file to write the RMAN repository.)

If the database is shut down normally and brought to a MOUNT state, and then a closed database backup is taken, the resulting backup is a **consistent backup**. When a database is restored from a consistent backup, it can be opened immediately. There is no redo to apply after the datafiles are restored.

If a database was not shut down normally before backup (that is, if the database was open, or exited abnormally, as in the case of a SHUTDOWN ABORT) then any backup you take is an **inconsistent backup**. When a database is restored from an inconsistent backup, Oracle must apply redo before the database can be opened. However, because inconsistent backups can be taken while the database is still available, inconsistent backups are a key part of many backup strategies.

Note: Prior to RMAN, taking online backups required that you place datafiles into backup mode using the `ALTER DATABASE/TABLESPACE BEGIN BACKUP` statement. RMAN, by contrast, requires that you **not** place a tablespace in backup mode. Do not use `ALTER DATABASE/TABLESPACE BEGIN BACKUP` before an RMAN backup.

Files That RMAN Can Back Up

RMAN can back up all database files needed for efficient recovery in the event of a failure. RMAN's BACKUP command supports backing up these types of files:

- Datafiles, and image copies of datafiles

- Control files, and image copies of control files
- Archived redo logs
- The current server parameter file
- Backup pieces, containing other backups created by RMAN

Although the database depends on other types of files for operation, such as network configuration files, password files, and the contents of the Oracle home, these files cannot be backed up with RMAN. Use some non-RMAN backup solution for any files not in the preceding list.

RMAN Backup Formats: Image Copies and Backup Sets

RMAN backups can be stored in one of two formats: as **image copies** or as **backup sets**.

Image Copies

An image copy is a bit-for-bit duplicate of a database file, identical to a copy made with an operating system command. (RMAN-created image copies are, however, recorded in the RMAN repository, unlike operating system-level file copies.) In principle, RMAN is not needed to restore a datafile from an image copy.

RMAN creates image copies when the `AS COPY` option is used with the `BACKUP` command. RMAN can create image copies of datafiles and datafile copies, control files and controlfile copies, archived redo logs, and backup pieces. RMAN supports creating image copies on disk, but not on media managers.

See Also: *Oracle Database Backup and Recovery Advanced User's Guide* for more detailed information about RMAN's handling of image copies

Backup Sets

RMAN can also store backup information in logical structures called **backup sets**. A backup set contains the data from one or more datafiles or archived redo logs, or control files or SPFILE. (Datafiles and archivelogs cannot be mixed together in the same backup set.) You can also back up existing backup sets into another backup set.

Only RMAN can create or restore from backup sets.When multiple files are backed up into the same backup set, they are read concurrently and their data is multiplexed together.

A backup set consists of one or more files called **backup pieces,** files in an RMAN-specific format. By default, a backup set consists of one backup piece. For example, you can back up ten datafiles into a single backup set containing a single backup piece (that is, one backup piece will be produced as output, and the backup piece and the backup set that contains it will be recorded in the RMAN repository). A file cannot be split across backup sets.

Backup sets are the only type of backup that RMAN supports on media manager devices such as tapes. Backup sets can also be created on disk. RMAN defaults to creating backups as backup sets on both disk and tape.

Note: The backup set is the smallest unit of a backup. RMAN only records backup sets in the repository that complete successfully. There is no such thing as a partial backup set. You cannot usefully manipulate individual backup pieces.

Unused Block Compression in Backup Sets Never-used data blocks in datafiles are never copied into backup sets, saving storage space and overhead during the backup process. Unused block compression is fundamental to how RMAN writes datafiles into backup pieces, and cannot be disabled.

Binary Compression of Backup Sets When storage space is more important to you than backup and restore times, you can use binary compression to reduce the size of your backup sets. The compression algorithm built into the Oracle server is tuned specifically for efficient compression of Oracle archived logs and datafiles, and will generally yield better compression than general-purpose compression utilities not tuned for Oracle database files.

Further, because it is integrated into Oracle, compressing backups requires only that you add the `AS COMPRESSED BACKUPSET` argument to your BACKUP command. Restoring from compressed backups requires no special action whatever.

Oracle Corporation recommends that you use RMAN's integrated binary compression instead of external compression utilities when you need to make compressed backups. For more on performance considerations when using binary compression of backup sets, see the description of the AS COMPRESSED BACKUPSET option of the BACKUP command, in *Oracle Database Recovery Manager Reference*.

Full and Incremental Datafile Backups

A full datafile backup is a backup that includes every used data block in the file. If the backup is created as a backupset, then unused block compression will cause unused blocks to be omitted. For an image copy, the entire file contents are reproduced exactly.

Note: A full backup is different from a whole database backup, which is a backup of all datafiles and the current control file.

Incremental backups can only be created for datafiles. Incremental backups of datafiles capture datafile changes on a block-by-block basis, rather than requiring the backup of all used blocks in a datafile. The resulting backup sets are generally smaller than full datafile backups, unless every block in the datafile is changed.

During media recovery, RMAN examines the restored files to determine whether it can recover them with an incremental backup. When possible, RMAN chooses incremental backups over archived logs, because applying all changes to a block at once is faster than reapplying individual changes to the block from the redo logs.

RMAN Backups and Tags

RMAN supports attaching a **tag** to a backup as a way of identifying that backup. Tags can be unique for a particular backup or set of backups taken at a given time, such as `2003_year_end`, or they can be re-used over time to identify a backup as being part of a series of backups, such as `weekly_incremental`. Many forms of the `BACKUP` command let you associate a tag with a backup, and many `RESTORE` and `RECOVER` commands let you specify which backups to use in the restore or recover operation by means of the tag provided when the backup was created.

In practice, tags are frequently used to distinguishing backups created in different strategies, especially incremental backup strategies.

Backing Up Database Files and Archived Logs with RMAN

This section contains these topics:

- Making Consistent and Inconsistent Backups with RMAN
- Making Whole Database Backups with RMAN
- Backing Up Individual Tablespaces with RMAN

- Backing Up Datafiles and Datafile Copies with RMAN
- Backing Up Control Files with RMAN
- Backing Up Server Parameter Files with RMAN
- Backing Up Archived Redo Logs with RMAN
- Using Compressed Backupsets

See Also:

- *Oracle Database Backup and Recovery Advanced User's Guide* for an overview of RMAN backups
- *Oracle Database Backup and Recovery Advanced User's Guide* for a discussion of the various RMAN backup types
- *Oracle Database Recovery Manager Reference* for BACKUP syntax

Making Consistent and Inconsistent Backups with RMAN

Consistent backups can be restored without recovery. To make a consistent backup, the database must be mounted and must not have suffered an instance failure or closed with SHUTDOWN ABORT the last time it was open. If these conditions are not met, then the backup is inconsistent. An inconsistent backup requires media recovery when it is restored, but is otherwise just as valid as a consistent backup.

You can use SQL*Plus or RMAN to start up and shut down the database. The following example connects to the target database, shuts it down cleanly, and then mounts it in preparation for a backup:

```
% rman TARGET /
RMAN> SHUTDOWN IMMEDIATE # closes database consistently
RMAN> STARTUP MOUNT # uses SPFILE
```

Making Whole Database Backups with RMAN

You can perform whole database backups with the database mounted or open. To perform a whole database backup, from the RMAN prompt, use the BACKUP DATABASE command.

The following example backs up all the datafiles as well as the control file and server parameter file (if used) to the default configured device (in this example, disk). The backup will be stored in the default backup destination with an automatically generated filename. The default is determined as follows:

- If a format is configured for the disk channel, RMAN uses the format as the default for backup storage.
- Otherwise, if a flash recovery area is in use, RMAN uses the flash recovery area as the default for backup storage..
- If there is neither a format for the disk channel nor a flash recovery area, the default location and filename format are platform-specific.

This example shows the procedure for taking a whole database backup to the default destination:

```
RMAN> BACKUP DATABASE;  # uses automatic channels to make backup
RMAN> SQL 'ALTER SYSTEM ARCHIVE LOG CURRENT'; # switches logs and archives all
logs
```

By archiving the logs immediately after the backup, you ensure that you have a full set of archived logs through the time of the backup. This guarantees that you can perform media recovery after restoring this backup.

The `FORMAT` parameter to the `BACKUP DATABASE` command lets you specify a different destination for backups and control the filename for the backup. For example, enter:

```
RMAN> BACKUP DATABASE FORMAT '/tmp/%U';  # %U generates a unique filename
```

You can also use the FORMAT argument to name an ASM disk group as backup destination, as shown in this example:

```
RMAN> BACKUP DATABASE FORMAT '+dgroup1';  # sets an ASM disk group
```

Optionally, use the `TAG` parameter to specify a backup tag. For example, enter:

```
RMAN> BACKUP DATABASE TAG = 'weekly_backup';   # gives the backup a tag
identifier
```

RMAN assigns a default tag to backups.

> **See Also:** *Oracle Database Recovery Manager Reference* for the default format description in `BACKUP ... TAG`

Backing Up Individual Tablespaces with RMAN

You can backup one or more individual tablespaces with the `BACKUP TABLESPACE` command. You can use this command when the database is mounted or open.

To back up a tablespace:

After starting RMAN, run the BACKUP TABLESPACE command at the RMAN prompt. This example backs up the users and tools tablespaces to tape, using the MAXSETSIZE parameter to specify that no backup set should be greater than 10 MB:

```
BACKUP DEVICE TYPE sbt MAXSETSIZE = 10M TABLESPACE users, tools;
```

Oracle translates the tablespace name internally into a list of datafiles.

Backing Up Datafiles and Datafile Copies with RMAN

You can back up datafiles and datafile copies when the database is mounted or open.

Backing Up Datafiles

Use the BACKUP DATAFILE command to back up individual datafiles. You can specify the datafiles by name or number.

To back up a datafile:

After starting RMAN and connecting to the target database, run the BACKUP DATAFILE command at the RMAN prompt. This example uses an sbt channel to back up datafiles 1-4 as well as a datafile copy:

```
BACKUP DEVICE TYPE sbt
  DATAFILE 1,2,3,4
  DATAFILECOPY '/tmp/system01.dbf';
```

If CONFIGURE CONTROLFILE AUTOBACKUP is ON, then RMAN writes the current control file and SPFILE to a separate autobackup piece. Otherwise, these files are automatically included in the backup set that contains datafile 1.

Backing Up Datafile Copies

Use the BACKUP DATAFILECOPY command to back up datafile copies. Datafile copies exist on disk only.

To back up a datafile copy:

While connected to the target database, run the BACKUP DATAFILECOPY command at the RMAN prompt. This example backs up datafile /tmp/system01.dbf to tape:

```
BACKUP DEVICE TYPE sbt DATAFILECOPY '/tmp/system01.dbf';
```

Backing Up Control Files with RMAN

You can back up the control file when the database is mounted or open. RMAN uses a snapshot control file to ensure a read-consistent version. If `CONFIGURE CONTROLFILE AUTOBACKUP` is `ON` (by default it is `OFF`), then RMAN automatically backs up the control file and server parameter file after every backup and after database structural changes. The control file autobackup contains metadata about the previous backup, which is crucial for disaster recovery.

If the autobackup feature is not set, then you must manually back up the control file in one of the following ways:

- Run `BACKUP CURRENT CONTROLFILE`
- Include a backup of the control file within any backup by using the `INCLUDE CURRENT CONTROLFILE` option of the `BACKUP` command
- Back up datafile `1`, because RMAN automatically includes the control file and SPFILE in backups of datafile 1

A manual backup of the control file is not the same as a control file autobackup. In manual backups, only RMAN repository data for backups within the current RMAN session is in the control file backup, and a manually backed-up control file cannot be automatically restored.

> **See Also:** *Oracle Database Backup and Recovery Advanced User's Guide* to learn more about control file autobackups

Backing Up the Current Control File Manually

After starting RMAN, run the `BACKUP CURRENT CONTROLFILE` command. This example backs up the current control file to the default disk device and assigns a tag:

```
BACKUP CURRENT CONTROLFILE TAG = mondaypmbackup;
```

If the autobackup feature is enabled, then RMAN makes two control file backups in this example: the explicit control file backup (`BACKUP CURRENT CONTROLFILE`) and the autobackup of the control file and server parameter file.

Including the Current Control File in a Backup Set

To include the current control file in a backup set, specify the `INCLUDE CURRENT CONTROLFILE` option after specifying the backup object. In this example, the default configured channel is to an `sbt` device. This command backs up tablespace `users` to tape and includes the current control file in the backup:

```
BACKUP DEVICE TYPE sbt TABLESPACE users INCLUDE CURRENT CONTROLFILE;
```

If the autobackup feature is enabled, then RMAN also creates an autobackup of the control file after the BACKUP TABLESPACE command completes.

Backing Up a Control File Copy

This example creates a control file backup with the BACKUP CONTROLFILECOPY command.

To back up a control file copy:

After starting RMAN, run the BACKUP CONTROLFILECOPY command at the RMAN prompt. This example creates the control file copy '/tmp/control01.ctl' on disk and then backs it up to tape:

```
BACKUP AS COPY CURRENT CONTROLFILE FORMAT '/tmp/control01.ctl';
BACKUP DEVICE TYPE sbt CONTROLFILECOPY '/tmp/control01.ctl';
```

Backing Up Server Parameter Files with RMAN

As explained in "Backing Up Control Files with RMAN" on page 4-9, RMAN automatically backs up the current server parameter file in certain cases. The BACKUP SPFILE command backs up the parameter file explicitly. For example:

```
BACKUP DEVICE TYPE sbt SPFILE;
```

The SPFILE that is backed up is the one currently in use by the instance. If the instance is started with a client-side initialization parameter file, then RMAN does not back up anything when this command is used.

Backing Up Archived Redo Logs with RMAN

Archived redo logs are the key to successful media recovery. Back them up regularly. You can back up logs with BACKUP ARCHIVELOG, or back up logs while backing up datafiles and control files by specifying BACKUP ... PLUS ARCHIVELOG.

Backing Up Archived Redo Log Files with BACKUP ARCHIVELOG

To back up archived redo logs, use the BACKUP ARCHIVELOG command at the RMAN prompt. This example uses a configured disk or sbt channel to back up one copy of each log sequence number for all archived redo logs:

```
BACKUP ARCHIVELOG ALL;
```

Even if your redo logs are being archived to multiple destinations and you use RMAN to back up archived redo logs, RMAN selects only one copy of the archived redo log file to include in the backup set. (Since logs with the same log sequence number are identical, there is no need to include more than one copy.)

You can also specify a range of archived redo logs by time, SCN, or log sequence number, as in the following example:

```
BACKUP ARCHIVELOG
  FROM TIME 'SYSDATE-30' UNTIL TIME 'SYSDATE-7';
```

Automatic Online Redo Log Switches During Backups of Archived Logs When taking a backup of archived redo logs that includes the most recent log (that is, a BACKUP ... ARCHIVELOG command is run without the UNTIL or SEQUENCE option) if the database is open, then before beginning the backup, RMAN will switch out of the current online redo log group, and all online redo logs that have not yet been archived, up to and including the redo log group that was current when the commadn was issued. This ensures that the backup contains all redo that was generated prior to the start of the command.

Using BACKUP ARCHIVELOG with DELETE INPUT or DELETE ALL INPUT You can specify the DELETE INPUT or DELETE ALL INPUT clauses for the BACKUP ARCHIVELOG command to delete archived logs after they are backed up, eliminating the separate step of manually deleting the archived redo logs. With DELETE INPUT, RMAN only deletes the specific copy of the archived redo log chosen for the backup set. With DELETE ALL INPUT, RMAN will delete each backed-up archived redo log file from all log archiving destinations.

For example, assume that you archive to /arc_dest1, /arc_dest2, and /arc_dest3, and you run the following command:

```
BACKUP DEVICE TYPE sbt
  ARCHIVELOG ALL
  DELETE ALL INPUT;
```

In this case RMAN backs up only one copy of each log sequence number in these directories, and then deletes *all* copies of any log that it backed up from the archiving destinations. If you had specified DELETE INPUT rather than DELETE ALL INPUT, then RMAN would only delete the specific archived redo log files that it backed up (for example, it would delete the archived redo log files in /arc_dest1 if those were the files used as the source of the backup, but it would leave the contents of the /arc_dest2 and /arc_dest3 intact) .

If you issue BACKUP ARCHIVELOG ALL or BACKUP ARCHIVELOG LIKE '...', and there are no archived redo log files to back up, then RMAN does not signal an error.

Backing Up Logs with BACKUP ... PLUS ARCHIVELOG

You can add archived redo logs to a backup of other files by using the BACKUP ... PLUS ARCHIVELOG clause. Adding BACKUP ... PLUS ARCHIVELOG causes RMAN to do the following:

1. Runs the ALTER SYSTEM ARCHIVE LOG CURRENT command.
2. Runs BACKUP ARCHIVELOG ALL. Note that if backup optimization is enabled, then RMAN skips logs that it has already backed up to the specified device.
3. Backs up the rest of the files specified in BACKUP command.
4. Runs the ALTER SYSTEM ARCHIVE LOG CURRENT command.
5. Backs up any remaining archived logs generated during the backup.

This guarantees that datafile backups taken during the command are recoverable to a consistent state.

To back up archived redo logs with BACKUP ... PLUS ARCHIVELOG:

After starting RMAN, run the BACKUP ... PLUS ARCHIVELOG command at the RMAN prompt . This example backs up the database and all archived logs:

```
BACKUP DEVICE TYPE sbt
  DATABASE PLUS ARCHIVELOG;
```

Note: If backup optimization is enabled, then RMAN skips backups of archived logs that have already been backed up to the specified device.

Using Compressed Backupsets

For any use of the BACKUP command that creates backupsets, you can take advantage of RMAN's support for binary compression of backupsets, by using the AS COMPRESSED BACKUPSET option to the BACKUP command. The resulting backupsets are compressed using an algorithm optimized for efficient compression of Oracle database files. No extra uncompression steps are required during recovery if you use RMAN's integrated compression.

This example backs up the entire database and archived logs to the configured default backup destination (disk or tape), producing compressed backupsets:

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

This example backups up several datafiles to the default device, using binary compression:

```
BACKUP AS COMPRESSED BACKUPSET DATAFILE 1,2,4;
```

Predictably, creating compressed backupsets imposes some extra CPU overhead during backup and restore, which can slow the backup process. The performance penalty can be worth paying, however, in a number of circumstances:

- If you are using disk-based backups and disk space in your flash recovery area or other disk-based backup destination is limited
- If you are performing your backups to some device over a network and reduced network bandwidth is more important than CPU usage
- If you are using some archival backup media such as CD or DVD, where reducing backup sizes saves on media costs and archival storage

Note that performance while creating compressed backupsets is CPU bound. If you have more than one CPU, you can use increased parallelism to run jobs on multiple CPUs and thus improve performance.

Note: If you are backing up to tape and your tape device performs its own compression, you should not use both RMAN backupset compression and the media manager vendor's compression. In most instances you will get better results using the media manager's compression. See the discussion of tuning RMAN's tape backup performance in *Oracle Database Backup and Recovery Advanced User's Guide* for details.

RMAN Incremental Backups

You can make incremental backups of databases, individual tablespaces or datafiles.

As with other backups, if you are in `ARCHIVELOG` mode, you can make incremental backups if the database is open; if the database is in `NOARCHIVELOG` mode, then you can only make incremental backups when the database is closed.

The goal of an incremental backup is to back up only those data blocks that have changed since a previous backup.

The primary reasons for making incremental backups part of your strategy are:

- For use in a strategy based on incrementally updated backups, where these incremental backups will be used to roll forward an image copy of the database
- To reduce the amount of time needed for daily backups
- To save network bandwidth when backing up over a network
- To get adequate backup performance when the aggregate tape bandwidth available for tape write I/Os is much less than the aggregate disk bandwidth for disk read I/Os
- To be able to recover changes to objects created with the `NOLOGGING` option. For example, direct load inserts do not create redo log entries and their changes cannot be reproduced with media recovery. They do, however, change data blocks and so are captured by incremental backups.
- To reduce backup sizes for `NOARCHIVELOG` databases. Instead of making a whole database backup every time, you can make incremental backups. Note that incremental backups of a `NOARCHIVELOG` database are only legal after a consistent shutdown.

See Also: *Oracle Database Concepts* for more information about `NOLOGGING` mode

One effective strategy is to make incremental backups to disk, and then back up these image copies to a media manager with BACKUP AS BACKUPSET. This avoids the problem of keeping the tape streaming that can occur when making incremental backups directly to tape. Because incremental backups are not as big as full backups, you can create them on disk more easily.

Incremental Backup Algorithm

Each data block in a datafile contains a system change number (SCN), which is the SCN at which the most recent change was made to the block. During an incremental backup, RMAN reads the SCN of each data block in the input file and compares it to the checkpoint SCN of the parent incremental backup. If the SCN in the input data block is greater than or equal to the checkpoint SCN of the parent, then RMAN copies the block.

Note that if you enable the block change tracking feature, RMAN can refer to the change tracking file to identify changed blocks in datafiles without scanning the full contents of the datafile. Once enabled, block change tracking does not alter how you take or use incremental backups, other than offering increased performance. See "Improving Incremental Backup Performance: Change Tracking" on page 4-23 for more details about enabling block change tracking.

Level 0 and Level 1 Incremental Backups

Incremental backups can be either level 0 or level 1. A level 0 incremental backup, which is the base for subsequent incremental backups, copies all blocks containing data, backing the datafile up into a backup set just as a full backup would. The only difference between a level 0 incremental backup and a full backup is that a full backup is never included in an incremental strategy.

A level 1 incremental backup can be either of the following types:

- A **differential backup**, which backs up all blocks changed after the most recent incremental backup at level 1 or 0
- A **cumulative backup**, which backs up all blocks changed after the most recent incremental backup at level 0

Incremental backups are differential by default.

> **Note:** Cumulative backups are preferable to differential backups when recovery time is more important than disk space, because during recovery each differential backup must be applied in succession. Use cumulative incremental backups instead of differential, if enough disk space is available to store cumulative incremental backups.

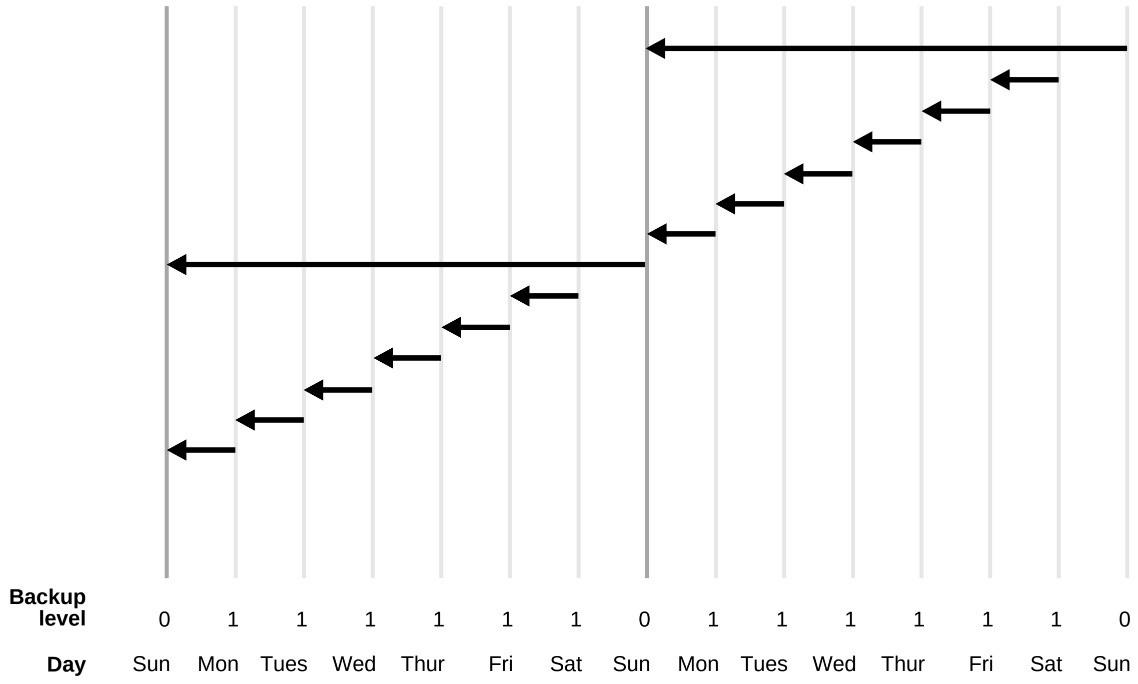
The size of the backup file depends solely upon the number of blocks modified and the incremental backup level.

Differential Incremental Backups

In a differential level 1 backup, RMAN backs up all blocks that have changed since the most recent cumulative or differental incremental backup, whether at level 1 or level 0. RMAN determines which level 1 backup occurred most recently and backs up all blocks modified after that backup. If no level 1 is available, RMAN copies all blocks changed since the level 0 backup.

If no level 0 backup is available, then the behavior depends upon the compatibility mode setting. If compatibility is >=10.0.0, RMAN copies all blocks changed since the file was created, and stores the results as a level 1 backup. In other words, the SCN at the time the incremental backup is taken is the file creation SCN. If compatibility <10.0.0, RMAN generates a level 0 backup, to be consistent with the behavior in previous releases.

Figure 4–1   Differential Incremental Backups (Default)



In the example shown in Figure 4–1, the following occurs:

- Sunday

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

- Monday - Saturday

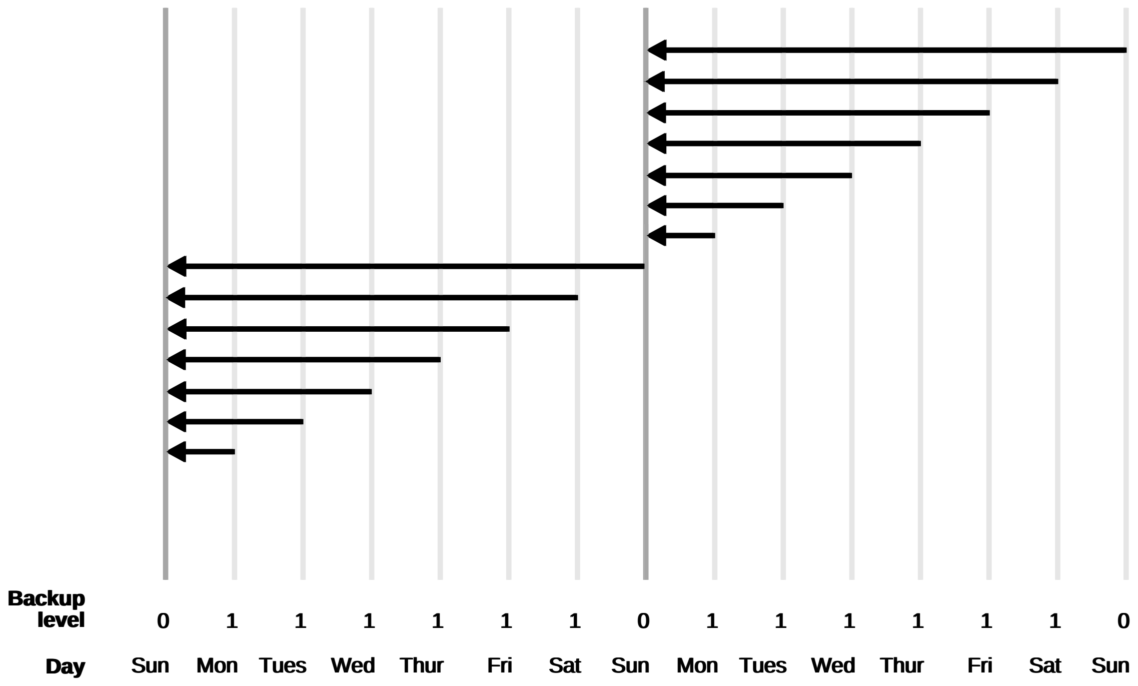
  On each day from Monday through Saturday, a differential incremental level 1 backup backs up all blocks that have changed since the most recent incremental backup at level 1 or 0. So, the Monday backup copies blocks changed since Sunday level 0 backup, the Tuesday backup copies blocks changed since the Monday level 1 backup, and so forth.

- The cycle is repeated for the next week.

Cumulative Incremental Backups

In a cumulative level 1 backup, RMAN backs up all the blocks used since the most recent level 0 incremental backup. Cumulative incremental backups reduce the work needed for a restore by ensuring that you only need one incremental backup from any particular level. Cumulative backups require more space and time than differential backups, however, because they duplicate the work done by previous backups at the same level.

Figure 4–2 Cumulative Incremental Backups



In the example shown in Figure 4–2, the following occurs:

- Sunday

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

- Monday - Saturday

  A cumulative incremental level 1 backup copies all blocks changed since the most recent level 0 backup. Because the most recent level 0 backup was created on Sunday, the level 1 backup on each day Monday through Saturday backs up all blocks changed since the Sunday backup.

- The cycle is repeated for the next week.

Basic Incremental Backup Strategy

Choose a backup scheme according to an acceptable MTTR (mean time to recover). For example, you can implement a three-level backup scheme so that a full or level 0 backup is taken monthly, a cumulative level 1 is taken weekly, and a differential level 1 is taken daily. In this scheme, you never have to apply more than a day's worth of redo for complete recovery.

When deciding how often to take full or level 0 backups, a good rule of thumb is to take a new level 0 whenever 50% or more of the data has changed. If the rate of change to your database is predictable, then you can observe the size of your incremental backups to determine when a new level 0 is appropriate. The following query displays the number of blocks written to a backup set for each datafile with at least 50% of its blocks backed up:

```
SELECT FILE#, INCREMENTAL_LEVEL, COMPLETION_TIME, BLOCKS, DATAFILE_BLOCKS
  FROM V$BACKUP_DATAFILE
  WHERE INCREMENTAL_LEVEL > 0
  AND BLOCKS / DATAFILE_BLOCKS > .5
  ORDER BY COMPLETION_TIME;
```

Compare the number of blocks in differential or cumulative backups to a base level 0 backup. For example, if you only create level 1 cumulative backups, then when the most recent level 1 backup is about half of the size of the base level 0 backup, take a new level 0.

Making Incremental Backups: BACKUP INCREMENTAL

After starting RMAN, run the `BACKUP INCREMENTAL` command at the RMAN prompt. This example makes a level 0 incrementnal backup of the database:

```
BACKUP INCREMENTAL LEVEL 0 DATABASE;
```

This example makes a differential level 1 backup of the `SYSTEM` tablespace and datafile `tools01.dbf`. It will only back up those data blocks changed since the most recent level 1 or level 0 backup:

```
BACKUP INCREMENTAL LEVEL 1
  TABLESPACE SYSTEM
  DATAFILE 'ora_home/oradata/trgt/tools01.dbf';
```

This example makes a cumulative level 1 backup of the tablespace `users`, backing up all blocks changed since the most recent level 0 backup.

```
BACKUP INCREMENTAL LEVEL = 1 CUMULATIVE
```

```
TABLESPACE users;
```

Incrementally Updated Backups: Rolling Forward Image Copy Backups

Oracle's Incrementally Updated Backups feature lets you create an image copy of a datafile, then regularly create incremental backups of your database and apply them to that image copy. The image copy is updated with all changes up through the SCN at which the incremental backup was taken. RMAN can use the resulting updated datafile in media recovery just as it would use a full image copy taken at that SCN, without the overhead of performing a full image copy of the database every day.

A backup strategy based on incrementally updated backups can help minimize time required for media recovery of your database. If you run scripts to implement this strategy daily, then at recovery time, you never have more than one day of redo to apply.

Incrementally Updated Backups: A Basic Example

To create incremental backups for use in an incrementally updated backups strategy, you must use the BACKUP... FOR RECOVER OF COPY WITH TAG form of the BACKUP command. How the command works is best understood in the context of an example script that would implement the strategy.

This script, run on a regular basis, is all that is required to implement a strategy based on incrementally updated backups:

```
RUN {
   RECOVER COPY OF DATABASE WITH TAG 'incr_update';
   BACKUP INCREMENTAL LEVEL 1 FOR RECOVER OF COPY WITH TAG 'incr_update'
       DATABASE;
   }
```

The syntax used in the script does not, however, make it clear how the strategy works. To understand the script and the strategy, it is necessary to understand the effects of these two commands when no datafile copies or incremental backups exist.

- The `BACKUP INCREMENTAL LEVEL 1... FOR RECOVER OF COPY WITH TAG`... command does not always create a level 1 incremental backup. If there is no incremental level 0 backup of an individual datafile to use with this level 1 backup, then executing this command creates a level 0 backup of the datafile with the specified tag.

Therefore, the first time the script runs, it creates the level 0 backup of the datafile needed to begin the cycle of incremental updates. In the second run and all subsequent runs, it produces level 1 incremental backups of the datafile.

- The `RECOVER COPY OF DATABASE WITH TAG`... command causes RMAN to apply any incremental level 1 backups to a set of datafile copies with the same tag. If there is no incremental backup or no datafile copy, the command generates a message but does not generate an error.

  The first time the script runs, this command has no effect, because there is neither a datafile copy nor a level 1 incremental backup.

  The second time the script runs, there is a datafile copy (created by the first `BACKUP` command), but no incremental level 1 backup, so again, the command has no effect.

  On the third run and all subsequent runs, there is a datafile copy and a level 1 incremental from the previous run, so the level 1 incremental is applied to the datafile copy, bringing the datafile copy up to the checkpoint SCN of the level 1 incremental.

Note also the following details about how this example works:

- Each time a datafile is added to the database, an image copy of the new datafile is created the next time the script runs. The time after that, the first level 1 incremental for that datafile is created, and on all subsequent runs the new datafile is processed like any other datafile.
- Tags must be used to identify the incremental backups and datafile copies created for use in this strategy, so that they do not interfere with other backup strategies you implement. (If you have multiple incremental backup strategies in effect, RMAN cannot unambiguously select incremental level 0 and level 1 backups for use in incremental backup and recover operations, unless they are tagged.)

In practice, you would schedule the example script to run once each day, possibly at midnight. On a typical night (that is, after the first two nights), when the script completed the following files would be available for a point-in-time recovery:

- An image copy of the database, as of the checkpoint SCN of the preceding run of the script, 24 hours earlier
- An incremental backup for the changes since the preceding run
- Archived redo logs including all changes between the checkpoint SCN of the image copy and the current time

If, at some point during the following 24 hours, you need to restore and recover your database from this backup, for either complete or point-in-time recovery, you can restore the datafiles from the incrementally updated datafile copies, and apply changes from the most recent incremental level 1 and the redo logs to reach the desired SCN. At most, you will have 24 hours of redo to apply, which limits how long point-in-time recovery will take.

See Also: *Oracle 2 Day DBA* to see how this technique is used in the Oracle-suggested backup strategy in Enterprise Manager.

Incrementally Updated Backups: A One Week Example

The basic example can be extended to provide fast recoverability to a window greater than 24 hours. Alter the `RECOVER COPY... WITH TAG` to perform incomplete recovery of the datafile copies to the point in time in the past where you want your window of recoverability to begin. This example shows how to maintain a seven day window:

```
RUN {
   RECOVER COPY OF DATABASE WITH TAG 'incr_update'
       UNTIL TIME 'SYSDATE - 7';
   BACKUP INCREMENTAL LEVEL 1 FOR RECOVER OF COPY WITH TAG 'incr_update'
       DATABASE;
   }
```

The effect of the script is as follows:

- On the first night the `RECOVER COPY... UNTIL TIME` statement has no effect, and the `BACKUP INCREMENTAL... FOR RECOVER OF COPY` statement creates the incremental level 0 copy.
- On the second through seventh nights, the `RECOVER COPY... UNTIL TIME` statement has no effect because `TIME 'SYSDATE - 7'` is still a time in the future. The `BACKUP INCREMENTAL... FOR RECOVER OF COPY` statement creates differetial incremental level 1 backups containing the block changes for the previous day.
- On the eighth and all subsequent nights night, the `RECOVER COPY... UNTIL TIME` statement applies the level 1 incremental from seven days ago to the copy of the database. The `BACKUP INCREMENTAL... FOR RECOVER OF COPY` statement creates an incremental backup containing the changes for the previous day.

As with the basic example, you have fast recoverability to any point in time between the SCN of the datafile copies and the present, using block changes from the incremental backups and individual changes from the redo logs. Because you have the daily level 1 incrementals, you still never need to apply more than one day of redo.

Improving Incremental Backup Performance: Change Tracking

RMAN's change tracking feature for incremental backups improves incremental backup performance by recording changed blocks in each datafile in a change tracking file. If change tracking is enabled, RMAN uses the change tracking file to identify changed blocks for incremental backup, thus avoiding the need to scan every block in the datafile.

After enabling change tracking, the first level 0 incremental backup still has to scan the entire datafile, as the change tracking file does not yet reflect the status of the blocks. Subsequent incremental backup that use this level 0 as parent will take advantage of the change tracking file.

Using change tracking in no way changes the commands used to perform incremental backups, and the change tracking files themselves generally require little maintenance after initial configuration.

Change tracking is disabled by default, because it does introduce some minimal performance overhead on your database during normal operations. However, the benefits of avoiding full datafile scans during backup are considerable, especially if only a small percentage of data blocks are changed between backups. If your backup strategy involves incremental backups, then you should enable change tracking.

One block change tracking file is created for the whole database. By default, the block change tracking file is created as an Oracle managed file in `DB_CREATE_FILE_DEST`. You can also specify the name of the block change tracking file, placing it in any location you choose.

Oracle saves enough change-tracking information to enable incremental backups to be taken using any of the 8 most recent incremental backups as its parent.

Although RMAN does not support backup and recovery of the change-tracking file itself, if the whole database or a subset needs to be restored and recovered, then recovery has no user-visible effect on change tracking. After the restore and recovery, the change tracking file is cleared, and starts recording block changes again. The next incremental backup after any recovery is able to use change-tracking data.

The size of the change tracking file is proportional to the size of the database and the number of enabled threads of redo. The size is not related to the frequency of updates to the database. The space required for block change tracking is approximately 1/30,000 the size of the data blocks to be tracked. However, to avoid overhead of allocating space as your database grows, the change tracking file size starts at 10MB, and new space is allocated in 10MB incremenents. Thus, for any database up to approximately 1TB the file size is 10MB, for up to 2TB the file size is 20MB, and so on.

Enabling and Disabling Change Tracking

You can enable or disable change tracking when the database is either open or mounted. To alter the change tracking setting, you must use SQL*Plus to connect to the target database with administrator privileges.

To store the change tracking file in the database area, set DB_CREATE_FILE_DEST in the target database. Then issue the following SQL statement to enable change tracking:

```
SQL> ALTER DATABASE ENABLE BLOCK CHANGE TRACKING;
```

You can also create the change tracking file in a location you choose yourself, using the following SQL statement:

```
SQL> ALTER DATABASE ENABLE BLOCK CHANGE TRACKING
      USING FILE '/mydir/rman_change_track.f' REUSE;
```

The REUSE option tells Oracle to overwrite any existing file with the specified name.

To disable change tracking, use this SQL statement:

```
SQL> ALTER DATABASE DISABLE BLOCK CHANGE TRACKING;
```

If the change tracking file was stored in the database area, then it is deleted when you disable change tracking.

Checking Whether Change Tracking is Enabled From SQL*Plus, you can query V$BLOCK_CHANGE_TRACKING.STATUS to determine whether change tracking is enabled, and if it is, query V$BLOCK_CHANGE_TRACKING.FILENAME to display the filename.

Moving the Change Tracking File

If you need to move the change tracking file, the ALTER DATABASE RENAME FILE command updates the control file to refer to the new location. The process outlined in this section describes how to change the location of the change tracking file while preserving its contents.

To relocate the block change tracking file:

1. If necessary, determine the current name of the change tracking file:

   ```
   SELECT filename
   FROM V$BLOCK_CHANGE_TRACKING;
   ```

2. Shut down the database. For example:

   ```
   SHUTDOWN IMMEDIATE
   ```

3. Using host operating system commands, move the change tracking file to its new location.
4. Mount the database and move the change tracking file to a location that has more space. For example:

   ```
   ALTER DATABASE RENAME FILE 'ora_home/dbs/change_trk.f' TO '/new_disk/change_
   trk.f';
   ```

5. Open the database:

   ```
   ALTER DATABASE OPEN;
   ```

If you cannot shut down the database, then you must disable change tracking and re-enable it at the new location, as in the following example:

```
ALTER DATABASE DISABLE BLOCK CHANGE TRACKING;
ALTER DATABASE ENABLE BLOCK CHANGE TRACKING USING FILE 'new_location';
```

If you choose this method, you will lose the contents of the change tracking file. Until the next time you complete a level 0 incremental backup, RMAN will have to scan the entire file.

Backing Up to the Flash Recovery Area: Basic Scenarios

For all of the following scenarios, assume that the RMAN environment and flash recovery area are configured as shown in "Configure Flash Recovery Area for Disk-Based Backups: Example" on page 3-22.

Scripting Disk-Only Backups

Backup scripts depend on the database load and the amount of disk space allocated. You can categorize the scripts based on how the database is used and the minimum amount of disk limit required to achieve the recovery window on disk.

If few database blocks change, then the incremental backup size will be significantly less than the size of the database, and the best practice is to take incremental backups, to make efficient use of disk space and other resources.

If most or all database blocks change frequently, then the size of incremental backup will be roughly as large as the size of the database, and the best practice is to periodically make a complete image copy of the database.

In the following scenarios, you do not need to back up archived logs because Oracle archives the logs to the flash recovery area. They will remain in the flash recovery area for as long as they are required by the retention policy. All database backups are also all directed to the flash recovery area.

This section contains the following topics:

- Backup Scripts When Few Data Blocks Change
- Backup Scripts When Blocks Change Frequently
- Backup Scripts When a Moderate Number of Blocks Change Weekly

Backup Scripts When Few Data Blocks Change

If most data blocks do not change each day, incremental backups will be small and you can plan your strategy around daily incremental backups.

Initial Setup The minimum recommended flash recovery area size on disk is dependent on the how frequent the backups are taken and retention policy used.

If your configured retention policy is `REDUNDANCY X` and you plan to take a backup every *Y* days, then the formula for the recommended flash recovery area disk quota is:

```
Disk Quota =
Size of X copies of database +
Size of one copy of incremental backups +
Size of (Y+1) days of archived logs
```

If you configure your retention policy to a recovery window of *X* days and execute your backup script once in *Y* days then you can calculate your required disk quota by one of two formulas. If X>=Y then the formula for the recommended flash recovery area disk quota is:

```
Disk Quota =
Size of one copy of database +
Size of (floor(X/Y) + 1) copies of incremental backups +
Size of (X * ceil(X/Y) +1) days of archived logs
```

where `ceil(X/Y)` is the smallest integer greater than or equal to X/Y and `floor(X/Y)` is the largest integer less than or equal to X/Y.

If X <Y then the recommended formula is:

```
Disk Quota =
Size of one copy of database +
Size of 2 copies of incremental backups +
Size of (Y +1) days of archived logs
```

Size your flash recovery area according to the applicable formula.

For this example, assume that the retention policy is `REDUNDANCY 1`:

```
RMAN> CONFIGURE RETENTION POLICY TO REDUNDANCY 1;
```

Also assume that backups will be performed daily.

Daily Script The daily backup script would look like this:

```
RMAN> RECOVER COPY OF DATABASE WITH TAG "whole_db_cpy";
# Make an incremental backup of the database to the flash recovery area.
RMAN> BACKUP INCREMENTAL LEVEL 1
      FOR RECOVER OF COPY WITH TAG "whole_db_copy"
      DATABASE;
```

Assume that there are no backups tagged "whole_db_copy" as of the first day of this backup scheme. The results of running this script daily are as follows:

- On the first day, the `RECOVER` statement would have no effect, and the `BACKUP... FOR RECOVER OF COPY` would create the initial level 0 database copy.
- On the second day, the `RECOVER` statement still has no effect, as there is no level 1 incremental backup to be applied to the level 0 incremental database copy. The `BACKUP` command creates the first incremental level 1 backup.
- Each day after that, the incremental level 1 from the previous day is applied over the level 0 database copy, rolling it forward to the time of the incremental

level 1 of the previous day, and a new incremental level 1 is created, containing all changes since the level 1 incremental backup of the previous day.

Assuming that this backup strategy is put into effect on February 1, Table 4–1 shows how the flash recovery area contents change over time as a result of the strategy. (Note that the flash recovery area may contain other files based on other backup activities; this table only represents files related to this strategy.)

Table 4–1 Backup Timeline for Scenario When Few Blocks Change: Version 1

Day	Script Effects	Flash Recovery Area Contents After Script
Sun Feb 1	**1.** Attempt incremental update, which has no effect because there is no level 0 backup. **2.** Create level 0 backup.	Level 0incremental image copy backup, with SCN as of Sun Feb 1
Mon Feb 2	**1.** Attempt incremental update, which has no effect because there is no level 1 incremental. **2.** Back up level 1 incremental.	Level 0 incremental image copy backup with SCN as of Sun Feb 1, level 1 incremental backup containing changes from Feb 1 through Feb. 2, archived logs from Feb 1 through today
Feb 3 and after	**1.** Perform incremental update of level 0, rolling it forward to the prcvious day. **2.** Back up level 1 incremental.	Level 0 incremental image copy backup rolled forward to previous day's level 1 SCN, level 1 incremental backup with changes for previous 24 hours, archived logs from February 1 through today. Old incremental backups and archived logs not useful for recovery of the rolled-forward level 0 backup are automatically deleted when more free space is needed in the flash recovery area.

To alter the example slightly: if you can size the flash recovery area to hold *n* days worth of archived logs and incremental backups (where *n* > 1), then you can alter the `RECOVER` line to `RECOVER COPY UNTIL TIME 'SYSDATE-`*`n`*`'`. For example, if the flash recovery area is large enough to hold three days of incremental backups, then you can change the script as follows:

```
RECOVER COPY OF DATABASE TAG "whole_db_copy" UNTIL TIME 'SYSDATE-3';
# Make an incremental backup of the database to the flash recovery area.
BACKUP INCREMENTAL LEVEL 1
      FOR RECOVER OF COPY WITH TAG "whole_db_copy"
      DATABASE;
```

Every day that you run the script, RMAN rolls forward the whole database copy to an SCN three days before the current time. Hence, the disk quota rules will preserve archived logs and incremental backups created *after* `SYSDATE-3` (because they are needed to recover the database to an SCN within the last three days.

The following table lists the set of files expected to be in the flash recovery area each day after the daily script for this strategy is run.

Backup Scripts When Blocks Change Frequently

In this scenario, typical of a CRM environment, many or most of the data blocks are updated over the course of a week.

If you use a retention policy of redundancy X, and a backup script executed once each Y days, the minimum disk quota required is determined by the following formula:

```
Disk Quota = Size of X copies of the database +
        size of Y days of archived redo logs
```

If you use a retention policy of a recovery window of X days, and the backup script is executed once each Y days, then if X>=Y, then disk quota is determined by the following formula:

```
Disk Quota = Size of 1 copy of the database +
         size of ((X * ceil(X/Y)) +1) days of archived redo logs
```

where `ceil(X/Y)` is the smallest integer greater than or equal to X/Y.

If X<Y, the disk quota is the same as when X=Y, that is,

```
Disk Quota = Size of 1 copy of the database +
         size of (Y +1) days of archived redo logs
```

For this example, assume that the retention policy is `REDUNDANCY 1`, and you are executing the backup script once each week. The minimum disk quota required for this scenario is as follows:

```
Disk Quota = Size of 1 copy of the database +
        size of 8 days of archived logs
```

This scenario requires no initial setup script. The weekly backup script is shown here:

```
# Execute once a week
# Make a full backup of the database to the flash recovery area.
BACKUP DATABASE TAG "weekly_full_bkup";
```

As shown in Table 4–2, the disk quota needs only to keep one level 0 copy of the database and one week's worth of archived logs.

Table 4–2 Backup Timeline for Scenario When Most Blocks Change

Day	Action	Contents of Flash Recovery Area
Sun Feb 1	Back up level 0.	Full backup
Sun Feb 8	Back up level 0.	Full backup from today, archived logs from Feb 1 through Feb 8
Each subsequent Sunday	Back up level 0.	Full backup from this Sunday, archived logs from the previous Sunday through this Sunday

Backup Scripts When a Moderate Number of Blocks Change Weekly

Use this strategy when roughly half the data blocks change weekly, or when the number of data blocks that change varies widely from week to week. The scripts in this strategy are "all purpose" scripts that use incrementally updated backups, starting with an image copy of the database, taking incremental level 1 backups at regular intervals, and rolling forward the existing level 0 copy.

The formula for determining space required for the flash recovery area depends upon the retention policy. For a retention policy of `REDUNDANCY X` and a backup script that runs once every Y days, the disk space required is:

```
Disk Quota = Size of X copies of the database +
             Size of 1 incremental backup +
             Size of Y+1 days of archived redo logs
```

For a recovery window of X days and a backup script executed once each Y days, if X>=Y then the space required is:

```
Disk Quota = Size of 1 copy of the database +
             Size of floor(X/Y)+1 incremental backups +
             Size of (X * ceil(X/Y)) + 1 days of archived logs
```

where `ceil(X/Y)` is the smallest integer greater than or equal to X/Y and `floor(X/Y)` is the largest integer less than or equal to X/Y.

If X< Y then the space required is the same as when X=Y:

```
Disk Quota = Size of 1 copy of the database +
             Size of 2 incremental backups +
             Size of Y+1 days of archived logs
```

For the example that follows, assume that the retention policy is `REDUNDANCY 1` and the backup script is run once each week.

Initial Setup The minimum disk quota for such a scenario is

```
Disk Quota = Size of 1 copy of the database +
             Size of 1 incremental backup +
             Size of 8 days of archived logs
```

The first week, the `BACKUP... FOR RECOVER OF COPY` command makes a level 0 incremental backup of the database to the flash recovery area. This backup will be the basis for the weekly strategy. Each subsequent week, this copy is rolled forward to the last incremental backup checkpoint time. In this way, you create an on-disk recovery window of one week.

Weekly Script After the initial setup, the weekly backup script (which, for example, you could run every Sunday night) should look like the following:

```
# Execute once a week
# Roll forward the whole copy of the database to last incremental backup SCN
RECOVER COPY OF DATABASE TAG "whole_db_cpy";
# Make an incremental backup of the database to the flash recovery area.
BACKUP INCREMENTAL LEVEL 1
      FOR RECOVER OF COPY WITH TAG "whole_db_copy"
      DATABASE;
```

Table 4–3 illustrates how the contents of the flash recovery area change with each run of the backup script, always keeping enough archived logs and backups on hand to maintain the seven day recovery window.

Table 4–3 Backup Timeline for Scenario When Moderate Number of Blocks Change

Day	Action	Contents of Flash Recovery Area After Run
Sun Feb 1	**1.** Attempt roll-forward of incremental level 0 copy of database with tag "whole_db_copy". Fail, because there is no level 0 copy. **2.** Create level 0 incremental backup.	Level 0 backup

Table 4–3 Backup Timeline for Scenario When Moderate Number of Blocks Change

Day	Action	Contents of Flash Recovery Area After Run
Sun Feb 8	**1.** Attempt roll-forward of incremental level 0 copy of database with tag "whole_db_copy" using previous week's incremental level 1 backup. Fail, because there is no level 1 backup from the previous week. **2.** Create level 1 incremental backup.	Level 0 backup from Sunday Feb 1 (not rolled forward because no level 1 exists), archived logs from Feb 1 through Feb 8
All future Sundays	**1.** Roll forward level 0 copy of database to SCN of most recent incremental level 1 backup. **2.** Create incremental level 1 backup with changes since last incremental level 1 backup SCN.	Level 0 backup rolled forward to SCN of previous Sunday's incremental level 1, archived logs from previous Sunday through today. Archived logs from prior to previous Sunday and the previous week's incremental level 1 backup are obsolete after the roll-forward of the level 0 backup, because there is no other backup to which they could be applied. Obsolete files may remain in the flash recovery area until the need for disk space causes the database to delete them or until the `DELETE OBSOLETE` command is used.

Backing Up to the Flash Recovery Area and to Tape: Basic Scenarios

This scenario describes how to use RMAN to make backups to the flash recovery area and then later move them to tape.

The scripts in this section use the flash recovery area as the log archiving destination and the destination for all disk backups. Therefore, whenever there is no space to create archived logs or backups in the flash recovery area, Oracle automatically deletes archived logs and backups that are obsolete or that have been moved to tape.

By moving backup files and archived redo logs to tape, more space becomes available in the flash recovery area for new files. The formulas for the flash recovery area disk quota are suggested minimums, and depend upon the availability of space on tape to satisfy the retention policy. You can, however, increase your on-disk recovery window and reduce the need to restore archived redo logs and backups from tape during a database restore-and-recovery scenario, by allocating a larger flash recovery area disk quota than these formulas specify.

Configuring the RMAN Environment for Disk and Tape Backups

Invoke RMAN to configure the retention policy, backup optimization, and the control file autobackup. For example:

```
CONNECT TARGET SYS/orace@trgt;
# Recovery window of 15 days ensures that the database is recoverable within 7
# days using backups on disk and tape
CONFIGURE RETENTION POLICY TO RECOVERY WINDOW OF 15 DAYS;
CONFIGURE BACKUP OPTIMIZATION ON;
# Because you do not use a recovery catalog, enable the autobackup feature
CONFIGURE CONTROLFILE AUTOBACKUP ON;
CONFIGURE DEVICE TYPE sbt PARMS='...' PARALLELISM 1; # PARMS are vendor-specific
```

Writing Backup Scripts for Disk and Tape Scenarios

As in the disk-only scenarios, the backup scripts in this section are categorized based on database workload.

Backup Scripts When Few Data Blocks Change

In this scenario, relatively few data blocks change frequently, so daily level 1 incremental backups will typically be small. The goal is to keep one level 0 incremental backup on disk, incrementally updated each day using a level 1 incremental backup, and then move all other files onto tape. This keeps flash recovery area usage to a minimum.

Initial Setup The only required setup is to create a flash recovery area with the required disk quota, and set the flash recovery area as a redo log archiving destination.

The formula for determining your disk quota depends upon the backup retenion policy. It is the same, however, whether your retention policy is based on redundancy or recovery window. Use the following formula:

```
Disk Quota = Size of 1 copy of the database
             + size of 1 day's level 1 incremental backup
             + size of (Y+1) days of archived logs
```

where Y is the number of days that elapse between executionof BACKUP RECOVERY AREA in your backup scripts.

For this example, there is one backup script, executed every day, which creates a new incremental level 1 backup with that day's changes, rolls forward the level 0 backup using the level 1 backup from the previous day, and backs up all flash recovery area files to tape. Since the script backs up the flash recovery area to tape daily, the flash recovery area must be large enough to hold a copy of the database, the daily level 1 incremental backup, and two days' worth of archived redo logs.

The on-disk recovery window is one day. To recover to a point farther back than one day, RMAN must restore backups from tape.

Daily Script After the initial setup, a typical daily RMAN backup script would look like the following:

```
# Execute each day of the week
# Roll forward the copy to most recent incremental backup SCN, which will
# be yesterday's incremental level 1 backup
RECOVER COPY OF DATABASE WITH TAG "daily_backup";
# Take incremental backups to flash recovery area (using default disk channel)
BACKUP INCREMENTAL LEVEL 1 FOR RECOVER OF COPY WITH TAG "daily_backup" DATABASE;
# Back up flash recovery area to tape
BACKUP RECOVERY AREA;
# delete obsolete backups on tape
DELETE OBSOLETE DEVICE TYPE sbt;
```

Table 4–4 describes how the contents of the flash recovery area and tape change as this script is run each day. The script rolls forward the level 0 database copy tagged as `daily_backup` to the SCN of the preceding day's incremental level 1 backup, then creates a new level 1 incremental backup with the previous day's changes. For example, when you run the script on Thursday, RMAN rolls forward the level 0 backup to the SCN of the incremental backup on Wednesday.

Table 4–4 Backup Timeline for Scenario When Few Blocks Change

Day	Action	Contents of Flash Recovery Area and SBT
Sun Feb 1	1. Attempt roll-forward of level 0 backup; fail because there is no level 0 backup yet. 2. Create level 0 incremental backup to be rolled forward on future days.	Level 0 incremental backup. Any other flash recovery area files such as archived redo logs are copied to tape and may be deleted from flash recovery area when space is needed.
Mon Feb 2	1. Attempt roll-forward of level 0 backup; fail because there is no level 1 backup yet. 2. Create level 1 incremental backup, containing changes during Feb 2. 3. Back up flash recovery area to sbt.	Level 0 backup from Feb 1, level 1 incremental backup containing changes from Feb 2. Any other flash recovery area files such as archived redo logs are copied to tape and may be deleted from flash recovery area when space is needed. Any obsolete files stored on tape are deleted.
All future days	1. Roll forward incremental level 0 backup to SCN of previous level 1 incremental backup. 2. Create level 1 incremental backup containing changes from previous day. 3. Back up flash recovery area to sbt.	Level 0 backup rolled forward to previous day, level 1 incremental backup containing changes from this day Any other flash recovery area files such as archived redo logs are copied to tape and may be deleted from flash recovery area when space is needed. Any obsolete files stored on tape are deleted.

Backup Scripts When Many Blocks Change

In this scenario, many or most of the data blocks are updated over the course of a week, as in a CRM environment. A strategy using daily incremental backups would not be recommended because the size of the incremental backups could be quite large and is hard to predict.

A better strategy in this case is to back up the database to tape weekly and the archived logs to tape every day, and to use a recovery window-based retention policy.

The elements of the strategy are as follows:

- Redo log files are archived in the flash recovery area.

- A full backup of the database is taken to the flash recovery area once each week.
- Once each day, any backup files (including archived redo logs) in the flash recovery area that is not currently stored on tape is backed up to tape using the `BACKUP RECOVERY AREA` command, and any obsolete backups on tape are deleted.

If you use a recovery window-based retention policy, all backups required to perform point-in-time recovery within the window are retained as long as they are needed. This provides the necessary protection for your data, even with limited use of disk space.

Initial Setup Create your flash recovery area with the required disk quota and set up the flash recovery area as a redo log archiving destination. The formula for determining your disk quota depends upon the frequency of backups of the flash recovery area to tape, as follows:

```
Disk Quota = Size of 1 copies of the database
           + size of (Y+1) days of archived logs
```

where *Y* is the number of days that elapse between executionof `BACKUP RECOVERY AREA` in your backup scripts.

For this example, assume that you take a full database backup once each week, and back up the flash recovery area to tape on all seven days of the week.

There are two scripts in this strategy. Use the first script at the beginning of each week (say, on Sunday) to create a full database backup, and the other script on the remaining days of the week (Monday through Saturday) to back up the flash recovery area contents (the archived redo logs for the day) to tape.

Note that both of the scripts in this strategy include the BACKUP RECOVERY AREA command, so that command is executed every day. For this example, Y=1, so the disk quota is the size of one copy of the database plus two days of archived redo logs.

Weekly Scripts This RMAN backup script is executed once each week, on Sunday:

```
# Execute only once a week
# Take copy of database to flash recovery area
BACKUP AS COPY DATABASE;
# Take backup of flash recovery area to tape
BACKUP RECOVERY AREA;
# Delete obsolete backups on tape
DELETE OBSOLETE DEVICE TYPE sbt;
EXIT;
```

Daily Script The RMAN script to be executed each day (Monday through Saturday) would look like the following:

```
# Execute 6 days/wk
# Take backup of flash recovery area to tape
BACKUP RECOVERY AREA;
# delete obsolete backups on tape
DELETE OBSOLETE DEVICE TPE sbt;
```

Because the entire flash recovery area is backed up to tape every day, the database may delete backups and archived redo log files from the flash recovery area whenever space is needed for new files. Between backups it may be difficult to predict exactly which files will still be in the flash recovery area, because files are deleted whenever space is needed.

Backup Scripts When Blocks Change Moderately

The following strategy is based on full database backups taken at regular intervals. It is useful when about half of the data blocks change during the interval between full database backups, or when the number of changed data blocks varies widely between full database backups.

The strategy is based on an incrementally updated backups-based full backup of the database, rolled forward once each week. This requires a level 0 disk copy incremental backup of the database, and a level 1 incremental created each Sunday with the previous seven days of datafile changes. After the roll-forward, the contents of the flash recovery area are backed up to tape, so the incremental level 0 copy of the database and the level 1 weekly incremental backup are available both on disk and tape.

Each day, archived redo log files accumulate in the flash recovery area. Each time the daily script runs, those redo log files are backed up to tape. After that, they may be deleted from disk if space is needed in the flash recovery area.

Initial Setup The minimum flash recovery area disk quota for this scenario is calculated with the following formula:

```
Disk Quota = Size of 1 copy of the database
           + Size of 1 level 1 incremental backup with Y days of changes
           + Size of (Y+1) days of archived redo logs
```

where *Y* is the number of days between backing up the flash recovery area to tape.

For this example, *Y*=7, so the disk quota must be at least the size of one copy of the database, one seven-day level 1 incremental backup, and eight days' worth of archived redo logs.

Weekly Script The script that implement the Sunday backup for this strategy weekly parts of the backup is as follows:

```
# Execute once a week
# Roll forward the whole copy of the database to last incremental backup SCN
RECOVER COPY OF DATABASE WITH TAG "whole_db_cpy";
# Make an incremental backup of the database to the flash recovery area.
BACKUP DEVICE TYPE DISK INCREMENTAL LEVEL 1
       FOR RECOVER OF COPY WITH TAG "whole_db_copy"
       DATABASE;
# Back up flash recovery area to tape
BACKUP RECOVERY AREA;
# delete obsolete backups on tape
DELETE OBSOLETE DEVICE TYPE sbt;
```

Daily Script The daily script for this strategy is shown here:

```
# Execute 6 days/wk
# Back up flash recovery area to tape
BACKUP RECOVERY AREA;
# delete obsolete backups on tape
DELETE OBSOLETE DEVICE TYPE sbt;
```

Table 4–5 illustrates how the scripts maintain the recovery window. Assume for the purposes of this example that the week begins on Sunday.

Table 4–5 Backup Timeline for Scenario When Moderate Number of Blocks Change

Day	Script	Action	Contents of Flash Recovery Area and Tape After Script
Week 1, Sunday	Weekly	**1.** Attempt RECOVER COPY OF DATABASE command. This command has no effect, because there is no level 0 backup of the database yet. **2.** Create level 0 backup as result of the BACKUP... FOR RECOVER OF COPY command. **3.** Back up all files in the flash recovery area to tape. **4.** Delete obsolete backups from tape.	Flash recovery area contains the level 0 incremental backup of the database, to be rolled forward each week. Tape contains a backup of the level 0 incremental backup of the database.
Week 1, Monday-Saturday	Daily	**1.** Back up flash recovery area to tape. **2.** Delete obsolete backups from tape.	Flash recovery area contains archived redo logs for the previous day, and the incremental level 0 database backup. Tape contains a backup of the level 0 incremental backup, and all redo logs for all days since Sunday of week 1. No backups are obsolete.
Week 2, Sunday	Weekly	**1.** Attempt RECOVER COPY OF DATABASE level 0 to most recent incremental SCN. This command has no effect, because there is no level 1 incremental backup to use in roll-forward. **2.** Perform level 1 incremental backup. A level 1 incremental backup is created, including changes from week 1. **3.** Back up flash recovery area to tape. **4.** Delete obsolete backups from tape.	Flash recovery area contains incremental level 0 database backup at SCN of Sunday of week 1, level 1 incremental backup with changes from week 1. Tape contains level 0 incremental backup from Sunday of week 1, level 1 incremental backup with all changes from week 1, archived redo logs from week 1. No backups are obsolete.

Table 4–5 Backup Timeline for Scenario When Moderate Number of Blocks Change

Day	Script	Action	Contents of Flash Recovery Area and Tape After Script
Week 2, Monday-Saturday	Daily	**1.** Back up flash recovery area to tape. **2.** Delete obsolete backups from tape.	Flash recovery area contains archived redo logs for the previous day, and the incremental level 0 database backup. May also contain other archived redo logs. Tape contains a backup of the level 0 incremental backup, and all redo logs for all days since Sunday of week 1.
Week 3, Sunday	Weekly	**1.** Perform RECOVER COPY OF DATABASE level 0 to most recent incremental SCN. The level 0 incremental backup is rolled forward to the beginning of week 2. **2.** Perform level 1 incremental backup. A level 1 incremental backup is created including changes from week 2. **3.** Back up flash recovery area to tape. **4.** Delete obsolete backups from tape.	Flash recovery area contains incremental level 0 database backup at SCN of Sunday from week 2, level 1 incremental backup with changes from week 2. Tape contains level 0 incremental backup rolled forward to SCN of Sunday of week 2, level 1 incremental backup with all changes from week 2, archived redo logs from week 2. Obsolete backups deleted from tape include archived redo logs from week 1, tape backup of level 0 incremental database backup from week 1.

Backup Scripts When Not Enough Disk Space for a Database Backup

If the disk quota is not sufficient to keep a copy of database, then the flash recovery area should be used *only* as an archived log destination. The disk quota rules will automatically delete these logs from the flash recovery area when they are no longer needed (because they are obsolete or because they have been backed up to tape). Back up the archived logs to tape daily, to ensure that they can be deleted from the flash recovery area when space is needed there for other files.

For this strategy, the flash recovery area should be sized large enough to hold at least two days' worth of archived redo logs, plus one day's worth of incremental backup.

This strategy is based on level 0 and level 1 incremental backups (though it does not use incrementally updated backups). This strategy uses two scripts: one to be executed once a week (for example, on Sunday), and the other to be executed every day except for the day that the first script is executed (Monday through Saturday).

The script that runs once each week creates a level 0 incremental backup on tape, containing the full contents of the database.

The script that runs each day except the first day creates level 1 incremental backups containing the changes to the database each day.

Weekly Script Here is the script to run once at the beginning of each week

```
# Execute only once a week
# backup database to tape
BACKUP DEVICE TYPE sbt INCREMENTAL LEVEL 0 DATABASE;
# delete obsolete backups on tape
DELETE OBSOLETE DEVICE TYPE sbt;
# backup recovery file destination to tape
BACKUP RECOVERY AREA;
```

Daily Script This script executes each day of the week except for the first day, for example, Monday through Saturday:

```
# backup recovery file destination to tape
BACKUP RECOVERY AREA;
# Take incremental backups to flash recovery area
BACKUP DEVICE TYPE DISK INCREMENTAL LEVEL 1 DATABASE;
```

Table 4–6 illustrates how the scripts maintain the RMAN retention policy recovery window and how the disk quota is maintained. The recovery files that exist after the execution of the script each day also exist on tape, so Oracle can delete files from the flash recovery area as needed.

Table 4–6 Backup Timeline for Limited Disk Space Scenario

Day	Script	Action	Contents of Flash Recovery Area and Tape After Script
Sun Feb 1	Once-a-week	**1.** Back up level 0 to tape. **2.** Delete obsolete from tape. **3.** Back up flash recovery area to tape.	Flash recovery area contains... Tape contains a level 0 backup of the whole database from February 1, and any archived redo logs.

Table 4–6 Backup Timeline for Limited Disk Space Scenario

Day	Script	Action	Contents of Flash Recovery Area and Tape After Script
Mon Feb 2 - Sat Feb 7	Daily	1. Back up flash recovery area to tape. 2. Back up level 1 to disk.	Flash recovery area contains... Tape contains a level 0 backup of the whole database from February 1, incremental level 1 backups for each day from Monday through the present day, and any archived redo logs for the whole week.
Sun Feb 8	Once-a-week	1. Back up level 0 to tape. 2. Delete obsolete from tape. 3. Back up flash recovery area to tape.	Flash recovery area contains... Tape contains...

Validating RMAN Backups

You can use the `VALIDATE` keyword of the `BACKUP` command to check datafiles for physical and logical corruption and confirm that all database files exist and are in the correct locations.

RMAN does not actually produce backup sets, but scans the files to determine whether they can be backed up and are not corrupted. The `BACKUP VALIDATE` command is similar to the `RESTORE VALIDATE` command. If the backup validation discovers corrupt blocks, then RMAN updates the `V$DATABASE_BLOCK_CORRUPTION` view with rows describing the corruptions. After a corrupt block is repaired, the row identifying this block is deleted from the view.

For example, you can validate that all database files and archived logs can be backed up by running a command as follows:

```
BACKUP VALIDATE DATABASE ARCHIVELOG ALL;
```

RMAN displays the same output that it would if it were really backing up the files. If RMAN cannot validate the backup of one or more of the files, then it issues an error message. For example, RMAN may show output similar to the following:

```
RMAN-00571: ===========================================================
RMAN-00569: =============== ERROR MESSAGE STACK FOLLOWS ===============
RMAN-00571: ===========================================================
RMAN-03002: failure of backup command at 08/29/2002 14:33:47
ORA-19625: error identifying file /oracle/oradata/trgt/arch/archive1_6.dbf
```

```
ORA-27037: unable to obtain file status
SVR4 Error: 2: No such file or directory
Additional information: 3
```

You cannot use the `MAXCORRUPT` or `PROXY` parameters with the `VALIDATE` option.

See Also:

- *Oracle Database Recovery Manager Reference* for `BACKUP` syntax
- *Oracle Database Backup and Recovery Advanced User's Guide* to learn how to repair corrupt blocks discovered by `BACKUP ... VALIDATE`

Overview of Querying the RMAN Repository

You can obtain information from the RMAN repository in several different ways. The following table describes the basic options.

Method	Need Catalog?	Description
`LIST` command	No	Use this command to list backups and database incarnations. The output displays those files operated on by the `CHANGE`, `CROSSCHECK`, and `DELETE` commands.
`REPORT` command	No	Use this command to find out which files need a backup, which backups are no longer needed, which files are in the schema, and so forth.
`SHOW` command	No	Use this command to show persistent RMAN configuration settings.
`PRINT SCRIPT` command	Yes	Use this command to display the names of the scripts stored in the recovery catalog.
Recovery catalog fixed views	Yes	Query these views to access the recovery catalog itself. Some information, such as the names and contents of the stored scripts, can only be obtained from the catalog views.
V$ views	No	Query these views to access records in the target database control file. RMAN obtains metadata for the recovery catalog from the control file. Some `V$` views such as `V$DATAFILE_HEADER`, `V$PROCESS`, and `V$SESSION` contain information not found in the catalog views.

The main sources of information about RMAN from within the RMAN interface are the `REPORT` and `LIST` commands. Use these commands to query the RMAN repository and determine what you have backed up as well as what you need to back up. If you are in SQL*Plus, then you can query `V$BACKUP_FILES`. Note that the database must be open to perform this query.

See Also:

- *Oracle Database Backup and Recovery Advanced User's Guide* to learn how to keep the RMAN repository up to date
- *Oracle Database Recovery Manager Reference* for `LIST` syntax
- *Oracle Database Recovery Manager Reference* for `REPORT` syntax

Listing RMAN Backups, Archived Logs, and Database Incarnations

The `LIST` command queries the recovery catalog or control file and lists the backups, archived logs, and database incarnations. You can specify these files when running the `CHANGE`, `CROSSCHECK`, and `DELETE` commands.

This section contains these topics:

- About RMAN Lists
- Listing Backups
- Listing Backups by File
- Listing Backups in Summary Mode
- Listing Backups with Restrictions
- Listing Database Incarnations

About RMAN Lists

You can control how the output is displayed by using the `BY BACKUP` and `BY FILE` options of the `LIST` command and choosing between the `SUMMARY` and `VERBOSE` options.

The primary purpose of the `LIST` command is to determine which backups are available. Note that only backups that completed successfully are stored in the repository. For example, you can list:

- Backups and proxy copies of a database, tablespace, datafile, archived redo log, or control file

- Backups that have expired
- Backups restricted by time, path name, device type, tag, or recoverability
- Incarnations of a database

Use LIST to determine what you need to back up. In particular, ensure that:

- The STATUS columns of the output tables list all backups as AVAILABLE
- All files that you need backed up are included in the output
- The backups recorded in the repository are recent

Note that the V$BACKUP_FILES also contains list information for backups.

Listing Backups

By default, RMAN lists backups by backup, which means that it serially lists each backup or proxy copy and then identifies the files included in the backup. You can also list backups by file.

By default, RMAN lists in verbose mode. You can also list backups in a summary mode if the verbose mode generates too much output.

Listing Backups by Backup

To list backups by backup, connect to the target database and recovery catalog (if you use one), and then execute the LIST BACKUP command. Specify the desired objects with the *listObjList* clause. For example, you can enter:

```
LIST BACKUP;       # lists backup sets, image copies, and proxy copies
LIST BACKUPSET;    # lists only backup sets and proxy copies
LIST COPY;         # lists only disk copies
```

Optionally, specify EXPIRED to identify backups not found during a crosscheck:

```
LIST EXPIRED BACKUP;
```

Examine the output (refer to *Oracle Database Recovery Manager Reference* for an explanation of the various column headings in the LIST output). Sample output of LIST BACKUP follows:

```
List of Backup Sets
===================

BS Key  Size       Device Type Elapsed Time Completion Time
------- ---------- ----------- ------------ ---------------
7       136M       DISK        00:00:20     04-NOV-03
```

```
        BP Key: 7   Status: AVAILABLE  Compressed: NO  Tag: TAG20031104T200759
        Piece Name: /oracle/work/RDBMS/backupset/2003_11_04/o1_mf_annnn_TAG20031104T200759_ztjxx3k8_.bkp

  List of Archived Logs in backup set 7
  Thrd Seq     Low SCN    Low Time  Next SCN   Next Time
  ---- ------- ---------- --------- ---------- ---------
  1    1       173832     21-OCT-03 174750     21-OCT-03
  1    2       174750     21-OCT-03 174755     21-OCT-03
  1    3       174755     21-OCT-03 174758     21-OCT-03

  1    37      533321     01-NOV-03 575472     03-NOV-03
  1    38      575472     03-NOV-03 617944     04-NOV-03
  1    39      617944     04-NOV-03 631495     04-NOV-03

BS Key  Type LV Size       Device Type Elapsed Time Completion Time
------- ---- -- ---------- ----------- ------------ ---------------
8       Full    2M         DISK        00:00:01     04-NOV-03
        BP Key: 8   Status: AVAILABLE  Compressed: NO  Tag: TAG20031104T200829
        Piece Name: /ade/lashdown_rdbms/oracle/dbs/c-774627068-20031104-01
  Controlfile Included: Ckp SCN: 631510       Ckp time: 04-NOV-03
  SPFILE Included: Modification time: 21-OCT-03
```

Sample output of LIST COPY follows:

```
List of Archived Log Copies
Key     Thrd Seq     S Low Time  Name
------- ---- ------- - --------- ----
37      1    37      A 01-NOV-03 /oracle/work/RDBMS/archivelog/2003_11_03/o1_mf_1_37_ztd4hl5d_.arc
38      1    38      A 03-NOV-03 /oracle/work/RDBMS/archivelog/2003_11_04/o1_mf_1_38_zthvg168_.arc
39      1    39      A 04-NOV-03 /oracle/work/RDBMS/archivelog/2003_11_04/o1_mf_1_39_ztjxwxwy_.arc
```

Listing Backups by File

Specify the desired objects with the *`listObjList`* or *`recordSpec`* clause (refer to *Oracle Database Recovery Manager Reference*). If you do not specify an object, then RMAN displays copies of all database files and archived logs. By default, RMAN lists in verbose mode, which means that it provides extensive, multiline information.

To list backups by file, connect the RMAN client to the target database and recovery catalog (if you use one), and then execute `LIST` with the `BY FILE` option, specifying the desired objects to list and options.For example, you can enter:

```
LIST BACKUP BY FILE; # shows backup sets, proxy copies, and image copies
LIST COPY BY FILE;   # shows only disk copies
```

For another example, you could specify the EXPIRED option to identify backups not found during a crosscheck:

```
LIST EXPIRED BACKUP BY FILE;
```

Examine the output (refer to *Oracle Database Recovery Manager Reference* for an explanation of the various column headings in the LIST output). Sample output follows:

```
List of Datafile Backups
========================

File Key     TY LV S Ckp SCN    Ckp Time  #Pieces #Copies Compressed Tag
---- ------- -  -- - ---------- --------- ------- ------- ---------- ---
1    5       B  F  A 631092     04-NOV-03 1       1       YES        TAG20031104T195949
     2       B  F  A 175337     21-OCT-03 1       1       NO         TAG20031021T094513
2    5       B  F  A 631092     04-NOV-03 1       1       YES        TAG20031104T195949
     2       B  F  A 175337     21-OCT-03 1       1       NO         TAG20031021T094513

... some rows omitted

List of Archived Log Backups
============================

Thrd Seq     Low SCN    Low Time  BS Key  S #Pieces #Copies Compressed Tag
---- ------- ---------- --------- ------- - ------- ------- ---------- ---
1    1       173832     21-OCT-03 7       A 1       1       NO         TAG20031104T200759
                                  1       A 1       1       NO         TAG20031021T094505
1    2       174750     21-OCT-03 7       A 1       1       NO         TAG20031104T200759
                                  1       A 1       1       NO         TAG20031021T094505
... some rows omitted
1    38      575472     03-NOV-03 7       A 1       1       NO         TAG20031104T200759
1    39      617944     04-NOV-03 7       A 1       1       NO         TAG20031104T200759


List of Controlfile Backups
===========================
CF Ckp SCN Ckp Time  BS Key  S #Pieces #Copies Compressed Tag
---------- --------- ------- - ------- ------- ---------- ---
631510     04-NOV-03 8       A 1       1       NO         TAG20031104T200829
631205     04-NOV-03 6       A 1       1       NO         TAG20031104T200432
175380     21-OCT-03 4       A 1       1       NO         TAG20031021T094639

List of SPFILE Backups
======================
Modification Time BS Key  S #Pieces #Copies Compressed Tag
----------------- ------- - ------- ------- ---------- ---
21-OCT-03         8       A 1       1       NO         TAG20031104T200829
21-OCT-03         6       A 1       1       NO         TAG20031104T200432
```

Listing Backups in Summary Mode

By default the `LIST` output is detailed, but you can also specify that RMAN display the output in summarized form. Specify the desired objects with the *`listObjectList`* or *`recordSpec`* clause. If you do not specify an object, then `LIST BACKUP` displays all backups.

After connecting to the target database and recovery catalog (if you use one), execute `LIST BACKUP`, specifying the desired objects and options. For example:

```
LIST BACKUP SUMMARY;  # lists backup sets, proxy copies, and disk copies
```

You can also specify the `EXPIRED` keyword to identify those backups that were not found during a crosscheck:

```
LIST EXPIRED BACKUP SUMMARY;
```

Sample output follows:

```
List of Backups
===============
Key     TY LV S Device Type Completion Time #Pieces #Copies Compressed Tag
------- -- -- - ----------- --------------- ------- ------- ---------- ---
1       B  A  A SBT_TAPE    21-OCT-03       1       1       NO         TAG20031021T094505
2       B  F  A SBT_TAPE    21-OCT-03       1       1       NO         TAG20031021T094513
3       B  A  A SBT_TAPE    21-OCT-03       1       1       NO         TAG20031021T094624
4       B  F  A SBT_TAPE    21-OCT-03       1       1       NO         TAG20031021T094639
5       B  F  A DISK        04-NOV-03       1       1       YES        TAG20031104T195949
6       B  F  A DISK        04-NOV-03       1       1       NO         TAG20031104T200432
7       B  A  A DISK        04-NOV-03       1       1       NO         TAG20031104T200759
8       B  F  A DISK        04-NOV-03       1       1       NO         TAG20031104T200829
```

Refer to *Oracle Database Recovery Manager Reference* for an explanation of the various column headings in the `LIST` output.

Listing Backups with Restrictions

You can specify several different conditions to narrow your `LIST` output.

After connecting to the target database and recovery catalog (if you use one), execute `LIST COPY` or `LIST BACKUP` with the *`listObjList`* or *`recordSpec`* condition. For example, enter any of the following commands:

```
# lists backups of all files in database
LIST BACKUP OF DATABASE;
# lists copy of specified datafile
LIST COPY OF DATAFILE 'ora_home/oradata/trgt/system01.dbf';
# lists specified backup set
```

```
LIST BACKUPSET 213;
# lists datafile copy
LIST DATAFILECOPY '/tmp/tools01.dbf';
```

You can also restrict the search by specifying the `maintQualifier` or RECOVERABLE clause. For example, enter:

```
# specify a backup set by tag
LIST BACKUPSET TAG 'weekly_full_db_backup';
# specify a backup or copy by device type
LIST COPY OF DATAFILE 'ora_home/oradata/trgt/system01.dbf' DEVICE TYPE sbt;
# specify a backup by directory or path
LIST BACKUP LIKE '/tmp/%';
# specify a backup or copy by a range of completion dates
LIST COPY OF DATAFILE 2 COMPLETED BETWEEN '10-DEC-2002' AND '17-DEC-2002';
# specify logs backed up at least twice to tape
LIST ARCHIVELOG ALL BACKED UP 2 TIMES TO DEVICE TYPE sbt;
```

The output depends upon the options you pass to the LIST command. For example, the following lists copies of datafile 1:

```
RMAN> list backup of datafile 1;


List of Backup Sets
===================

BS Key  Type LV Size       Device Type Elapsed Time Completion Time
------- ---- -- ---------- ----------- ------------ ---------------
2       Full    230M       SBT_TAPE    00:00:49     21-OCT-03
        BP Key: 2   Status: AVAILABLE  Compressed: NO  Tag: TAG20031021T094513
        Handle: 02f4eatc_1_1   Media: /smrdir
  List of Datafiles in backup set 2
  File LV Type Ckp SCN    Ckp Time  Name
  ---- -- ---- ---------- --------- ----
  1       Full 175337     21-OCT-03 /oracle/dbs/tbs_01.f

BS Key  Type LV Size       Device Type Elapsed Time Completion Time
------- ---- -- ---------- ----------- ------------ ---------------
5       Full    233M       DISK        00:04:30     04-NOV-03
        BP Key: 5   Status: AVAILABLE  Compressed: NO  Tag: TAG20031104T195949
        Piece Name: /oracle/work/RDBMS/backupset/2003_11_04/o1_mf_nnndf_
TAG20031104T195949_ztjxfvgz_.bkp
  List of Datafiles in backup set 5
  File LV Type Ckp SCN    Ckp Time  Name
  ---- -- ---- ---------- --------- ----
  1       Full 631092     04-NOV-03 /ade/lashdown_rdbms/oracle/dbs/tbs_01.f
```

See Also:

- *Oracle Database Recovery Manager Reference* for `listObjList` syntax
- *Oracle Database Recovery Manager Reference* for an explanation of the various columns in the `LIST` output

Listing Database Incarnations

Each time an `OPEN RESETLOGS` operation is performed on a database, this operation creates a new incarnation of the database. Database incarnations and their effect upon database recovery with RMAN are explained in more detail in *Oracle Database Backup and Recovery Advanced User's Guide*.

When performing incremental backups, RMAN can use a backup from a previous incarnation or the current incarnation as a basis for subsequent incremental backups. When performing restore and recovery, RMAN can use backups from a previous incarnation in restore and recovery operations just as it would use backups from the current incarnation, as long as all archived logs are available.

Use the `LIST INCARNATION` command to see the incarnations of your database.

To list database incarnations:

After connecting to the target database and the recovery catalog if applicable, run `LIST INCARNATION`:

```
RMAN> LIST INCARNATION;
```

If you are using a recovery catalog, and if you register multiple target databases in the same catalog, then you can distinguish them by using the `OF DATABASE` option:

```
RMAN> LIST INCARNATION OF DATABASE prod3;
```

Refer to *Oracle Database Recovery Manager Reference* for an explanation of the various column headings in the `LIST` output). Sample output follows:

```
RMAN> LIST INCARNATION OF DATABASE;

List of Database Incarnations
DB Key  Inc Key DB Name  DB ID            STATUS  Reset SCN  Reset Time
------- ------- -------- ---------------- ------  ---------- ----------
1       1       RDBMS    774627068        PARENT  1          21-OCT-03
2       2       RDBMS    774627068        CURRENT 173832     21-OCT-03
```

The preceding output indicates that a RESETLOGS was performed on database `trgt` at SCN 164378, resulting in a new incarnation. The incarnation is distinguished by incarnation key (represented in the `Inc Key` column).

Reporting on Backups and Database Schema

This section contains the following topics:

- About RMAN Reports
- Reporting on Objects Needing a Backup
- Reporting Obsolete Backups
- Reporting on the Database Schema

About RMAN Reports

To gain more detailed information from the RMAN repository, generate a report. The REPORT command can help answer questions such as the following:

- Which files need a backup?
- Which files have had unrecoverable operations performed on them?
- Which backups are obsolete and can be deleted?
- What was the physical schema of the database at some previous time?
- Which files have not been backed up recently?

Note: For the report to be accurate, the RMAN repository must be synchronized with the control file and you must have run the CHANGE, UNCATALOG, and CROSSCHECK commands recently to update the status of all backups. .

See Also: Chapter 6, "Recovery Manager Maintenance Tasks" to learn how to maintain the RMAN repository.

The information that you obtain from reports can be important for your backup and recovery strategy. In particular, run the REPORT NEED BACKUP and REPORT UNRECOVERABLE commands to ensure that the backups are available to perform recovery, and recovery can be performed within a reasonable length of time.

Reporting on Objects Needing a Backup

You can report on files that require a backup by specifying the NEED BACKUP keyword. The REDUNDANCY parameter specifies the minimum number of backups that must exist for a datafile to be considered not in need of a backup. If you do not specify the parameter, REDUNDANCY defaults to 1. The DAYS parameter indicates that recovery must begin by using logs more than *integer* days old. The INCREMENTAL parameter indicates that more than *integer* incremental backups are required for complete recovery.

Note: If you disable the retention policy, then REPORT NEED BACKUP with no other options generates an error message.

To report on files that need a backup:

1. After connecting to the target database and recovery catalog (if you use one), run CROSSCHECK commands as needed to verify the status and availability of backups known to RMAN. If files have been deleted, the RMAN repository will be out of date. This sequence of commands would crosscheck disk image copies, backupsets, and tape and proxy backups:

```
# allocate maintenance channel for crosscheck if no channels configured
ALLOCATE CHANNEL FOR MAINTENANCE DEVICE TYPE sbt;
CROSSCHECK BACKUP;  # crosschecks backup sets and proxy copies
```

2. This sequence of commands would crosscheck disk image copies only:

```
CROSSCHECK COPY;    # crosschecks only disk copies of
                    # archived redo logs, datafiles and control file
```

3. If you have a retention policy configured, then you can just run REPORT NEED BACKUP without any other options to determine which files need backups (sample output follows):

```
REPORT NEED BACKUP;

RMAN retention policy will be applied to the command
RMAN retention policy is set to redundancy 1
Report of files with less than 1 redundant backups
File #bkps Name
---- ----- -----------------------------------------------------
2    0     /oracle/oradata/trgt/undotbs01.dbf
```

4. To override the retention policy (or if you do not have a retention policy enabled), run REPORT NEED BACKUP DAYS. Any files older than the DAYS parameter value need a new backup because their backups require the specified number of DAYS worth of archived logs for recovery. For example, run:

```
REPORT NEED BACKUP DAYS = 7 DATABASE;  # needs min 7 days of logs to recover
REPORT NEED BACKUP DAYS = 30 TABLESPACE SYSTEM;
REPORT NEED BACKUP DAYS = 14 DATAFILE 'ora_home/oradata/trgt/tools01.dbf';
```

5. To determine which files need an incremental backup, specify the INCREMENTAL parameter. If complete recovery of a datafile requires more than the specified number of incremental backups, then RMAN considers it in need of a new backup. For example, enter:

```
REPORT NEED BACKUP INCREMENTAL = 1 DATABASE;
REPORT NEED BACKUP INCREMENTAL = 3 TABLESPACE SYSTEM;
REPORT NEED BACKUP INCREMENTAL = 5 DATAFILE 'ora_
home/oradata/trgt/users01.dbf';
```

See Also: *Oracle Database Recovery Manager Reference* for an explanation of the various column headings in the REPORT output

Reporting Obsolete Backups

You can report files that are obsolete, that is, superfluous, by specifying the OBSOLETE keyword. If you do not specify any other options, then REPORT OBSOLETE displays the backups that are marked obsolete by the current retention policy. By default, the retention policy is configured to REDUNDANCY of 1.

The REPORT OBSOLETE command supports the RECOVERY WINDOW and REDUNDANCY options at the command level, which have the same meanings as the options with the same names on the CONFIGURE command.

Note: As an alternative to running REPORT OBSOLETE, you can query the V$BACKUP_FILES view and check the OBSOLETE column.

To report obsolete backups:

1. After connecting to the target database and recovery catalog (if you use one), issue CROSSCHECK commands as needed to update the status of backups. Following is a possible crosscheck session:

```
# allocate maintenance channel for crosscheck if no channels configured
ALLOCATE CHANNEL FOR MAINTENANCE DEVICE TYPE sbt; # checks only on tape
CROSSCHECK BACKUP;  # crosschecks backup sets and proxy copies on tape
RELEASE CHANNEL;
```

2. Use the OBSOLETE option to identify which backups are obsolete because they are no longer needed for recovery. For example, enter:

```
# lists backups that not needed to recover the database to within last week
REPORT OBSOLETE RECOVERY WINDOW OF 7 DAYS;
# lists backups that are superfluous because more than 2 copies are on tape
REPORT OBSOLETE REDUNDANCY = 2 DEVICE TYPE sbt;
```

3. Use the ORPHAN option to list unusable backups belonging to an incarnation that is not a direct predecessor of the current incarnation. For example, enter:

```
REPORT OBSOLETE ORPHAN;
```

4. Optionally, delete obsolete backups. You can automatically delete obsolete backups by issuing DELETE OBSOLETE. For example, you can enter:

```
# delete obsolete backups displayed when you issue REPORT OBSOLETE
DELETE OBSOLETE;
# delete obsolete backups according to a specified recovery window
DELETE OBSOLETE RECOVERY WINDOW OF 7 DAYS;
# delete obsolete backups according to a specified redundancy
DELETE OBSOLETE REDUNDANCY = 2;
```

RMAN prompts for confirmation before actually deleting the files. To suppress the prompt, specify the NOPROMPT option. Specify FORCE to delete the files and remove their repository records regardless of whether the files exist.

See Also:

- "Reporting Obsolete Backups" on page 4-53 for a conceptual overview of reports of obsolete backups
- "Configuring the Backup Retention Policy" on page 3-18 for a conceptual overview

Reporting on the Database Schema

Issue REPORT SCHEMA to list the database files. If you use a recovery catalog, then you also generate historical reports of the database schema at a past time. You do not need a recovery catalog, however, to report the current schema.

To report the database schema at a specified point in time:

1. After connecting to the target database and recovery catalog (if you use one), issue REPORT SCHEMA:

   ```
   REPORT SCHEMA;
   ```

   If you use a recovery catalog, then you can use the *atClause* to specify a past time, SCN, or log sequence number:

   ```
   REPORT SCHEMA AT TIME 'SYSDATE-14';     # schema as it was two weeks ago
   REPORT SCHEMA AT SCN 1000;              # schema as it was at scn 1000
   REPORT SCHEMA AT SEQUENCE 100 THREAD 1; # schema as it was at sequence 100
   ```

2. Examine the report. For example, here is a sample output:

   ```
   REPORT SCHEMA AT SCN 1000;

   Report of database schema
   File K-bytes Tablespace           RB segs Datafile Name
   ---- ------- -------------------- ------- -------------------
   1     307200 SYSTEM               YES     /oracle/oradata/trgt/system01.dbf
   2      20480 UNDOTBS              YES     /oracle/oradata/trgt/undotbs01.dbf
   5      10240 EXAMPLE              NO      /oracle/oradata/trgt/example01.dbf
   6      10240 INDX                 NO      /oracle/oradata/trgt/indx01.dbf
   7      10240 TOOLS                NO      /oracle/oradata/trgt/tools01.dbf
   8      10240 USERS                NO      /oracle/oradata/trgt/users01.dbf
   ```

5 Performing Recovery

This chapter introduces the tools that RMAN makes available for recovering your database from backup. It includes the following topics:

Database Restore and Recovery with RMAN: Overview

The focus of this chapter is on how you use RMAN and backups created with RMAN to return your database to normal operation after the loss of one or more database files needed for its normal operation. The database files that RMAN backs up and can recover include the control file, server parameter file, datafiles and archived redo log files.

The chapter is organized as follows:

- "Preparing and Planning Database Restore and Recovery" on page 5-4 describes how to plan a restore-and-recovery operation, including how to determine which files need restore and recovery and what steps need to be taken during the recovery. It also describes how to preview the backups that RMAN will use during the restore operation, and how to verify that the backups to be used in the restore operation are valid.
- "Basic Database Restore and Recovery Scenarios" on page 5-11 presents some common database restore and recovery scenarios in their entirety. If one of these scenarios matches your situation, then you can follow the procedure outlined here. Even if the situation is not exactly the same, a review of the scenarios can help you create a plan for your own recovery process.
- "Restoring Different Types of Lost Database Files with RMAN" on page 5-14 presents a number of recovery procedures in isolation, such as how to restore a control file, an SPFILE, individual datafiles and redo logs, along with requirements in the event that you restore a particular type of file, such as steps you must take if you are restoring from a backup control file. If your exact restore and recovery scenario is not one of the basic scenarios outlined in the chapter, you can follow the processes outlined here to perform the individual tasks in your recovery plan.

The two most important RMAN commands used in database recovery are:

- RESTORE, which retrieves files from RMAN backups based on the contents of the RMAN repository
- RECOVER, which performs complete or point-in-time media recovery using available datafiles and redo logs.

Typically, you will set the state of the database appropriately for the data recovery operation to be performed, allocate or configure channels required to communicate with the disk and media manager, and then run a series of RESTORE and RECOVER commands. RMAN retrieves all needed files from backup and performs media recovery on all restored datafiles, to return your database to the desired state.

Scope and Limitations of this Chapter

This chapter introduces the techniques which will cover the most common restore and recovery scenarios. Anyone performing restore and recovery, even in complex scenarios not covered here, should be familiar with the techniques outlined in this chapter. Note, however, the following limitations on the scope of this discussion:

- Most of this chapter will focus on restore-and-recovery scenarios in which a media failure has damaged some or all of your database files, and your goal is to return your whole database to normal operation by restoring the damaged files from backup and recover all database changes in the redo log. While some more advanced types of recovery, such as point-in-time recovery of the whole database or individual tablespaces to recover from user errors, are mentioned in passing, *Oracle Database Backup and Recovery Advanced User's Guide* provides more extensive information on these techniques.

  See Also:

  - *Oracle Database Backup and Recovery Advanced User's Guide* for details on database point-in-time recovery (DBPITR)
  - *Oracle Database Backup and Recovery Advanced User's Guide* for details on tablespace point-in-time recovery (TSPITR)

- While there may be passing references to using RMAN with a recovery catalog, details on use of RMAN with a recovery catalog are covered in *Oracle Database Backup and Recovery Advanced User's Guide*.
- The discussion in this chapter will be limited to Oracle databases running in a single-instance configuration. While RMAN can perform restore and recovery of databases in Real Application Clusters and Data Guard configurations, such scenarios are beyond the scope of this manual. After reviewing this chapter for the basics of backup and recovery, refer to *Real Application Clusters Administration* and *Oracle Data Guard Concepts and Administration* for more information about using RMAN in those contexts.

Restore and Recovery with Enterprise Manager

Enterprise Manager provides access to much of the database restore and recovery functionality provided by RMAN through a set of recovery wizards, that lead the DBA through a variety of recovery procedures based on an analysis of your database, your available backups and your data recovery objectives.

Using RMAN through Enterprise Manager, you can perform the simpler restore and recovery scenarios outlined in this chapter, as well as much more sophisticated restore and recovery techniques such as point-in-time recovery and even use of the flashback features of the Oracle database, which allow for efficent repair of both media failure and user errors.

While the underlying functionality is the same, and the command-line client provides more flexibility, in many common situations, use of the Enterprise Manager interface to RMAN's restore and recovery features will be simpler than using the RMAN command line client directly.

See *Oracle 2 Day DBA* for more details on the restore and recovery features of Enterprise Manager.

Preparing and Planning Database Restore and Recovery

While RMAN makes carrying out most database restore and recovery tasks much simpler, you still have to plan your database restore and recovery actions based on which database files have been lost and what your recovery goal is.

RMAN can make most of the important decisions about the restore process for you, but you may want to preview and even override its decisions in some circumstances. For example, if you know a given backup is unavailable, due to a tape being stored offsite or a device being inaccessible, you can direct RMAN to not use that backup during the restore process.

RMAN provides tools to let you preview which backups will be used in a restore, and to validate the contents of the backups to ensure that they can be used in future restore operations.

Database Restore and Recovery Procedure: Outline

The basic procedure for performing restore and recovery with RMAN is as follows:

1. Determine which database files must be restored from backup, and which backups (which specific tapes, or specific backup sets or image copies on disk) you will use for the restore operation. The files to be restored may include the control file, SPFILE, archived redo log files, and datafiles.
2. Place the database in the state appropriate for the type of recovery that you are performing. For example, if you are recovering a single tablespace or datafile, then you can keep the database open and take the tablespace or datafile offline. Likewise, if you are restoring all datafiles, then you must shut down the database and then mount it before you can perform the restore.

3. Restore lost database files from backup with the `RESTORE` command. You may restore files to their original locations, or you may have to restore them to other locations if, for instace, a disk has failed. You may also have to update the SPFILE if you have changed the control file locations, or the control file if you have changed the locations of datafiles or redo logs.
4. Perform media recovery on restored datafiles, if any, with the `RECOVER` command.
5. Perform any final steps required to make the database available for users again. For example, restart the database if you are recovering from the loss of one copy of the control file, or bring offline tablespaces online if you were only recovering a limited number of datafiles.

This outline is intended to encompass a wide range of different scenarios. Depending upon your situation, some of the steps described may not apply. For example, you do not need to perform media recovery if the only file restored from backup is the SPFILE. You will have to devise your final recovery plan based on your particular situation.

Determining Which Database Files to Restore or Recover

It is generally obvious when the control file of your database must be restored, because the database shuts down immediately if any of the control file copies becomes inaccessible and the database cannot be started without a valid control file.

Loss of some but not all copies of your control file does not require recovery of the control file from backup. When one copy of the control file is lost, the database will automatically shut down. You can either copy an intact copy of the control file over the damaged or missing control file, or update the parameter file so that `CONTROL_FILES` does not refer to the damaged or missing control file. Once the `CONTROL_FILES` parameter references only present, intact copies of the control file, you can restart your database.

Note that if you restore the control file from backup, you must perform media recovery of the whole database and then perform an `OPEN RESETLOGS`, even if no datafiles have to be restored.

The need to restore the SPFILE is also easy to determine-- if the instance cannot read the SPFILE during startup, then you should restore it from backup.

Identifying Datafiles Requiring Media Recovery

When and how to recover depends on the state of the database and the location of its datafiles. To determine which if any files require media recovery, use the following procedure:

1. Start SQL*Plus and connect to the target database. For example, issue the following to connect to `trgt`:

   ```
   % sqlplus 'SYS/oracle@trgt AS SYSDBA'
   ```

2. Determine the status of the database by executing the following SQL query:

   ```
   SELECT STATUS FROM V$INSTANCE;
   ```

   If the status is `OPEN`, then the database is open. However, some datafiles may require media recovery.

3. Check the recovery and error columns of the `V$DATAFILE_HEADER` view to determine the status of your datafiles. Run the following SQL statements to check the datafile headers:

   ```
   COL FILE# FORMAT 999
   COL STATUS FORMAT A7
   COL ERROR FORMAT A10
   COL TABLESPACE_NAME FORMAT A10
   COL NAME FORMAT A30

   SELECT FILE#, STATUS, ERROR, RECOVER, TABLESPACE_NAME, NAME
   FROM V$DATAFILE_HEADER
   WHERE RECOVER = 'YES' OR (RECOVER IS NULL AND ERROR IS NOT NULL);
   ```

If the `ERROR` column is not `NULL` or the `RECOVER` column is not `NO`, then check for a temporary hardware or operating system problem causing the error. If there is no such problem, you must restore the file or switch to a copy.

A `NULL` value in the `RECOVER` column indicates that some hardware error prevented RMAN from reading the file's header or verifying its checksum.

If the `ERROR` column is `NULL` and the `RECOVER` column is `YES`, then the file can be recovered.

When you run the RMAN `RECOVER` command, incremental backups and archived redo logs are restored from backup as needed, and applied to the datafile to recover it to the current SCN.

Note: Because V$DATAFILE_HEADER only reads the header block of each datafile, it does not detect all problems that require the datafile to be restored. For example, you cannot tell whether a datafile contains corrupt data blocks using V$DATAFILE_HEADER.

4. If you are planning to perform complete recovery rather than point-in-time recovery, you can recover only those datafiles which require recovery, rather than the whole database. (Note that for point-in-time recovery, you must either restore and recover all datafiles or use the Oracle Flashback Database feature instead.)

   You can query V$RECOVER_FILE to list datafiles requiring recovery by datafile number with their status and error information.

   ```
   SELECT FILE#, ERROR, ONLINE, ONLINE_STATUS, CHANGE#, TIME
          FROM V$RECOVER_FILE;
   ```

   You can also perform useful joins using the datafile number and the V$DATAFILE and V$TABLESPACE views, to get the datafile and tablespace names. Use the following SQL*Plus commands to format the output of the query:

   ```
   COL DF# FORMAT 999
   COL DF_NAME FORMAT A35
   COL TBSP_NAME FORMAT A7
   COL STATUS FORMAT A7
   COL ERROR FORMAT A10
   COL CHANGE# FORMAT 99999999
   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#
   ;
   ```

   The ERROR column identifies the problem for each file requiring recovery.

   See Also: *Oracle Database Reference* for information about the V$ views

Determining your DBID

In situations requiring the recovery of your SPFILE or control file from autobackup, such as disaster recovery when you have lost all database files, you will need to determine your DBID. Your DBID should be recorded along with other basic information about your database, as recommended in "Keeping Records of the Hardware and Software Configuration of the Server" on page 2-17.

If you do not have a record of the DBID of your database, there are two places you can easily find it.

- The DBID is used in forming the filename for the control file autobackup. Locate that file, and then refer to "Configuring the Control File Autobackup Format" on page 3-10 to see where the DBID appears in the filename.
- If you have any text files that preserve the output from an RMAN session, the DBID is displayed by the RMAN client when it starts up and connects to your database. Typical output follows:

```
% rman TARGET /
Recovery Manager: Release 10.1.0.2.0 - Production

Copyright (c) 1995, 2003, Oracle.  All rights reserved.

connected to target database: RDBMS (DBID=774627068)

RMAN>
```

Previewing Backups Used in Restore Operations: RESTORE PREVIEW

The `RESTORE` command supports a `PREVIEW` option, which identifies the backups (backup sets or image copies, on disk or sequential media like tapes) required to carry out a given restore operation, based on the information in the RMAN repository. Use `RESTORE... PREVIEW` when planning your restore and recovery operation, to ensure that all required backups are available or to identify situations in which you may want to direct RMAN to use or avoid specific backups.

For example, `RESTORE... PREVIEW` can show you that RMAN will request a tape during the restore process which you know is stored offsite. You can then use the `CHANGE... UNAVAILABLE` command (described in "Marking a Backup AVAILABLE or UNAVAILABLE" on page 6-16) to set the backup status to UNAVAILABLE. If you then run `RESTORE... PREVIEW` again, RMAN will show you the backups it would use to perform a restore operation without using the unavailable backup.

Using RESTORE... PREVIEW

RESTORE ... PREVIEW can be applied to any RESTORE operation to create a detailed report of every backup to be used in the requested RESTORE operation. Here are a few examples of RESTORE commands using the PREVIEW option:

```
RESTORE DATABASE PREVIEW;
RESTORE TABLESPACE users PREVIEW;
RESTORE DATAFILE 3 PREVIEW;
RESTORE ARCHIVELOG FROM LOGSEQ 200 PREVIEW;
RESTORE ARCHIVELOG FROM TIME 'SYSDATE-7' PREVIEW;
RESTORE ARCHIVELOG FROM SCN 234546 PREVIEW;
```

RESTORE... PREVIEW output is in the same format as the output of the LIST command. See *Oracle Database Recovery Manager Reference* for details on interpreting the output of RESTORE... PREVIEW.

Using RESTORE... PREVIEW SUMMARY

If the detailed report produced by RESTORE... PREVIEW provides more information than is needed, use the RESTORE... PREVIEW SUMMARY option to suppress much of the detail about specific files used and affected by the restore process. Here are some examples of RESTORE used with the PREVIEW SUMMARY option:

```
RESTORE DATABASE PREVIEW SUMMARY;
RESTORE TABLESPACE users PREVIEW SUMMARY;
RESTORE DATAFILE 3 PREVIEW SUMMARY;
RESTORE ARCHIVELOG FROM LOGSEQ 200 PREVIEW SUMMARY;
RESTORE ARCHIVELOG FROM TIME 'SYSDATE-7' PREVIEW SUMMARY;
RESTORE ARCHIVELOG FROM SCN 234546 PREVIEW SUMMARY;
```

RESTORE... PREVIEW SUMMARY reports are in the same format as the output from the LIST SUMMARY command. See *Oracle Database Recovery Manager Reference* for details on interpreting the output of RESTORE... PREVIEW SUMMARY.

Validating the Restore of Backups: RESTORE VALIDATE

The RESTORE ... VALIDATE and VALIDATE BACKUPSET commands test whether you can restore from your backups. You can test the restore of either the entire database or individual tablespaces, datafiles, or control files. The contents of the backups are actually read to ensure that the objects to be restored can be restored from them. You have these options:

- RESTORE ... VALIDATE tests whether RMAN can restore a specific object from a backup. RMAN chooses which backups to use.

- `VALIDATE BACKUPSET` tests the validity of a backup set that you specify.

Validating with RESTORE ... VALIDATE

See Also:

- *Oracle Database Recovery Manager Reference* for `RESTORE` syntax
- *Oracle Database Recovery Manager Reference* for `VALIDATE` syntax

To validate backups with `RESTORE... VALIDATE`, the database can be mounted or open. You do *not* have to take datafiles offline when validating them.

This example illustrates validating the restore of the backup control file, `SYSTEM` tablespace, and all archived logs:

```
RESTORE CONTROLFILE VALIDATE;
RESTORE TABLESPACE SYSTEM VALIDATE;
RESTORE ARCHIVELOG ALL VALIDATE;
```

If you see error messages in the output and the following message, then RMAN cannot restore one of one of the specified files from your available backups:

```
RMAN-06026: some targets not found - aborting restore
```

If you see an error message stack and output similar to the following, for example, then RMAN encountered a problem restoring the specified file:

```
RMAN-03009: failure of restore command on c1 channel at 12-DEC-01 23:22:30
ORA-19505: failed to identify file "oracle/dbs/1fafv9gl_1_1"
ORA-27037: unable to obtain file status
SVR4 Error: 2: No such file or directory
Additional information: 3
```

If you do not see an error stack, then RMAN successfully tested restore of the specified objects from the available backups.

Validating with VALIDATE BACKUPSET

The `BACKUP VALIDATE` command requires that you know the primary keys of the backup sets that you want to validate.

To specify which backup sets to validate:

1. If you do not need to validate the whole database, then find the backups that you want to validate by running `LIST` commands, noting primary keys:

```
LIST BACKUP;
```

2. Validate the restore of the backup sets, referencing them by the primary keys. This example validates the restore of backup sets 1121 and 1122:

   ```
   VALIDATE BACKUPSET 1121,1122;
   ```

3. Check the output. If you see the `validation complete` message then RMAN successfully validated the restore of the specified backup set. For example:

   ```
   using channel ORA_DISK_1
   channel ORA_DISK_1: starting validation of archive log backupset
   channel ORA_DISK_1: restored backup piece 1
   piece handle=/oracle/dbs/0mdg9v8l_1_1 tag=TAG20020208T155604 params=NULL
   channel ORA_DISK_1: validation complete
   ```

Basic Database Restore and Recovery Scenarios

You can plan a strategy for recovering from most data losses using the process outlined in "Preparing and Planning Database Restore and Recovery" on page 5-4 and the task-specific procedures in "Restoring Different Types of Lost Database Files with RMAN" on page 5-14. However, some of the most common scenarios for database restore and recovery are presented in full here:

- Whole Database Restore and Recovery: Scenario
- Restore and Recovery of Individual Tablespaces or Datafiles: Scenario

The procedures outlined here will restore the whole database or individual tablespaces to their original locations.

To use the procedures in this section, the following requirements must be met:

- The current control file must be intact.
- You must have the complete set of archived logs and incremental backups needed for media recovery of your available datafile backups.
- For any datafiles for which you have no backup, you must have a complete set of online and archived redo logs going back to the creation of that datafile. (With a complete set of redo logs, RMAN can re-create a datafile for which there is no backup, by creating an empty datafile and then re-applying all changes since the file was created as part of the recovery process.)

If automatic channels are configured, then RMAN allocates all channels configured for the available device types according to their parallelism settings. Otherwise, you must enclose your RESTORE and RECOVER command in a RUN block, and begin by manually allocating the appropriate DISK or sbt channels. Otherwise, your RESTORE command will fail on attempting to retrieve backups from that device.

Whole Database Restore and Recovery: Scenario

In this scenario, you have a current control file and SPFILE but all datafiles are damaged or lost. You must restore and recover the whole database.

The database in this example has one read-only tablespace, history, which must be restored from backup but which does not need media recovery.

To restore and recover the database when the current control file is available:

1. After connecting to the target database, make sure the database is mounted.

```
RMAN> STARTUP MOUNT
```

2. Use the SHOW ALL command to see what channels are configured for access to backup devices. If automatic channels are *not* configured, then manually allocate one or more channels.
3. Restore the database using the RESTORE command, and recover it using the RECOVER command.
4. Examine the output to see if recovery was successful. If so, open the database.

This example performs restore and recovery of the database, using automatic channels.

```
RMAN> RESTORE DATABASE;
RMAN> RECOVER DATABASE DELETE ARCHIVELOG MAXSIZE 25MB;
```

The RECOVER DATABASE command as used here illustrates two useful options:

- DELETE ARCHIVELOG causes RMAN to delete restored log files after they have been applied to the datafiles, to save disk space.
- MAXSIZE 25MB limits space occupied by restored logs at any given moment to 25MB. This gives you more control over disk space usage by the restored logs. Note that if a single achived redo log file is larger than the specified MAXSIZE value, you will get an error. You will have to try your command again with a larger MAXSIZE value.

Read-only tablespaces may require special handling in a restore and recover operation. By default, the restore operation will skip read-only tablespaces. If a read-only tablespace is at the SCN where it became read-only after it is restored from backup, no redo will be applied to it when the rest of the database is recovered. You can force RMAN to restore any missing datafiles belonging to read-only tablespaces by using the `CHECK READONLY` option to the `RESTORE` command:

```
RMAN> RESTORE DATABASE CHECK READONLY;
RMAN> RECOVER DATABASE DELETE ARCHIVELOG;
```

If RMAN completes the recovery without error, you can open the database:

```
RMAN> ALTER DATABASE OPEN;
```

Restore and Recovery of Individual Tablespaces or Datafiles: Scenario

In this scenario, some but not all of the datafiles are damaged. You want to leave the database open so that the undamaged datafiles remain available.

Using the procedure described in "Determining Which Database Files to Restore or Recover" on page 5-5 to identify datafiles needing recovery, you discover that the damaged datafiles are from the tablespaces `users`.

The following restore and recovery procedure can be used if the database is mounted or open.

To recover a tablespace to its current location:

1. Connect to the target database and the recovery catalog database (if applicable), and make sure the database is mounted or open.
2. Take the tablespaces affected offline using `ALTER TABLESPACE ... OFFLINE IMMEDIATE` if they are not already offline.
3. Run `SHOW ALL` to see the current configuration, including configured default channels for the device where the backup is stored. If necessary, allocate one or more channels manually. (Note that if you manually allocate channels you must use a RUN block around your command.)
4. Restore the tablespace or datafile with the `RESTORE` command, and recover it with the `RECOVER` command.

This example restores and recovers the `users` tablespace, letting RMAN choose the backup to use on disk or tape:

```
RMAN> SQL 'ALTER TABLESPACE users OFFLINE IMMEDIATE';
RMAN> RESTORE TABLESPACE users;
RMAN> RECOVER TABLESPACE users;
```

If RMAN reported no errors during the recovery, then bring the tablespace back online:

```
RMAN> SQL 'ALTER TABLESPACE users ONLINE';
```

Restoring Different Types of Lost Database Files with RMAN

This section discusses how to restore the different types of database file backed up by RMAN. Once you have an overall plan for restoring the lost parts of your database, look here for details on how to execute the individual tasks in your plan.

Restoring the Control File from Backup

Loss or corruption of all copies of your control file requires restore of the control file from backup. The `RESTORE CONTROLFILE` command is used to restore the control file.

Note: After restoring the control files of your database from backup, you must perform complete media recovery as described in "Performing Media Recovery of a Database, Tablespace or Datafile" on page 5-21, and then open your database with the RESETLOGS option. The only exception is the case described in "Restore of the Control File to a New Location" on page 5-16, where you restore your control file to a location not listed in the `CONTROL_FILES` initialization parameter. In that case, you create a copy of your control file in the specified location without touching your running database.

Except as noted, the procedures in this section assume that you are not using a recovery catalog. Restore and recovery procedures for the control file when using a recovery catalog are covered in *Oracle Database Backup and Recovery Advanced User's Guide*.

RMAN can restore it to its default location (determined by rules described in the following section) or to one or more different locations of your choice, using the `RESTORE CONTROLFILE... TO` *`destination`* option.

Default Destination for Restore of the Control File

When restoring the control file, the default destination is all of the locations defined in the `CONTROL_FILES` initialization parameter. If you do not set the `CONTROL_FILES` initialization parameter, the database uses the same rules to determine the destination for the restored control file as it uses when creating a control file if the `CONTROL_FILES` parameter is not set. These rules are described in *Oracle Database SQL Reference* in the description of the `CREATE CONTROLFILE` statement.

Restore of the Control File from Control File Autobackup

If you are not using a recovery catalog, you must restore your control file from an autobackup. If you want to restore the control file from autobackup, the database must be in a NOMOUNT state. You must first set the DBID for your database, and then use the `RESTORE CONTROLFILE FROM AUTOBACKUP` command:

```
RMAN> SET DBID 320066378;
RMAN> RUN {
    SET CONTROLFILE AUTOBACKUP FORMAT
          FOR DEVICE TYPE DISK TO 'autobackup_format';
    RESTORE CONTROLFILE FROM AUTOBACKUP;
    }
```

RMAN uses the autobackup format and DBID to determine where to hunt for the control file autobackup. If one is found, RMAN restores the control file from that backup to all of the control file locations listed in the `CONTROL_FILES` initialization parameter.

For information on how to determine the correct value for *`autobackup_format`*, see the description of `CONFIGURE CONTROLFILE AUTOBACKUP FORMAT` in the entry for `CONFIGURE` In *Oracle Database Recovery Manager Reference*

See "Determining your DBID" on page 5-8 for details on how to determine your DBID.

Restore of the Control File When a Flash Recovery Area is Used

The commands used for restoring your control file are the same, whether or not you are using a flash recovery area. However, if you are using a flash recovery area, RMAN implicitly crosschecks backups and image copies listed in the control file, and catalogs any files in the flash recovery area not recorded in the restored control file. This improves the usefulness of the restored control file in the restoration of the rest of your database.

Note that tape backups are not automatically crosschecked after the restore of a control file. If you are using tape backups, then after restoring the control file and mounting the database you must crosscheck the backups on tape, as shown here:

```
RMAN> CROSSCHECK BACKUP DEVICE TYPE SBT;
```

Restore of the Control File When Using a Recovery Catalog

If you have a recovery catalog, you do not have to set the DBID or use the control file autobackup to restore the control file. You can use the `RESTORE CONTROLFILE` command with no arguments, as shown here:

```
RMAN> RESTORE CONTROLFILE;
```

The instance must be in NOMOUNT state when you perform this operation, and RMAN must be connected to the recovery catalog. The restored control file will be written to all locations listed in the `CONTROL_FILES` initialization parameter.

Restore of the Control File From a Known Location

You can restore the control file from a known control file copy using this form of the command:

```
RMAN> RESTORE CONTROLFILE from 'filename';
```

The control file copy found at the location specified by *`filename`* will be written to all locations listed in the `CONTROL_FILES` initialization parameter.

Restore of the Control File to a New Location

One way to restore the control file to one or more new locations is to change the `CONTROL_FILES` initialization parameter, and then use the `RESTORE CONTROLFILE` command with no arguments to restore the control file to the default locations. For example, if you are restoring your control file after a disk failure made some but not all `CONTROL_FILES` locations unusable, you can change `CONTROL_FILES` to replace references to the failed disk with pathnames pointing to another disk, and then run `RESTORE CONTROLFILE` with no arguments.

You can also restore the control file to any location you choose other than the `CONTROL_FILES` locations, by using the form `RESTORE CONTROLFILE TO 'filename' [FROM AUTOBACKUP]`:

```
RESTORE CONTROLFILE TO '/tmp/my_controlfile';
```

You can perform this operation with the database in NOMOUNT, MOUNT or OPEN states, because you are not overwriting any of the control files currently in use. Any existing file named '*`filename`*' is overwritten. After restoring the control file to a new location, you can then update the `CONTROL_FILES` initialization parameter to include the new location.

> **See Also:** *Oracle Database Recovery Manager Reference* for `RESTORE CONTROLFILE` syntax.

Limitations When Using a Backup Control File

Note the following requirements after restoring the control file from backup:

- After you restore your database using a backup control file, you **must** run `RECOVER DATABASE` and perform an `OPEN RESETLOGS` on the database.
- After restoring a backup control file, entries for tempfiles in locally-managed temporary tablespaces are removed. Hence, you must add new tempfiles to these tablespaces after you `OPEN RESETLOGS`. If you do not, then Oracle can display the error "`ORA-25153: Temporary Tablespace is Empty`" when attempting to sort.

For more details on restrictions on using `RESTORE CONTROLFILE` in different scenarios (such as when using a recovery catalog, or restoring from a specific backup), see the discussion of `RESTORE CONTROLFILE` in *Oracle Database Recovery Manager Reference*.

Restoring the Server Parameter File (SPFILE) from Backup

If you lose your server parameter file (SPFILE), RMAN can restore it to its default location or to a location of your choice.

Unlike the loss of the control file, the loss of your SPFILE does not cause your instance to immediately stop. Your instance may continue operating, although you will have to shut it down and restart it after restoring the SPFILE.

Note the following when restoring the SPFILE:

- If the instance is already started with the server parameter file, then you cannot overwrite the existing server parameter file.
- When the instance is started with a client-side initialization parameter file, RMAN restores the SPFILE to the default SPFILE location if the `TO` clause is not used. The default location is platform-specific (for example, *ora_home*`/dbs/spfile.ora` on Solaris).
- Restoring the SPFILE is one situation in which a recovery catalog can simplify your recovery procedure, because you can avoid the step of having to record and remember your DBID. This procedure assumes that you are not using a recovery catalog.

RMAN can also create a client-side initialization parameter file based on a backup of an SPFILE.

To restore the server parameter file:

1. If the database is up at the time of the loss of the SPFILE, connect to the target database. For example, run:

   ```
   % rman TARGET /
   ```

   If the database is not up when the SPFILE is lost, and you are not using a recovery catalog, then you must set the DBID of the target database. See "Determining your DBID" on page 5-8 for details on determining your DBID.

2. Shut down the instance and restart it without mounting. When the SPFILE is not available, RMAN starts the instance with a dummy parameter file. For example:

   ```
   RMAN> STARTUP FORCE NOMOUNT;
   ```

3. Restore the server parameter file. If restoring to the default location, then run:

   ```
   RMAN> RESTORE SPFILE FROM AUTOBACKUP;
   ```

   If restoring to a nondefault location, then you could run commands as in the following example:

   ```
   RMAN> RESTORE SPFILE TO '/tmp/spfileTEMP.ora' FROM AUTOBACKUP;
   ```

4. Restart the instance with the restored file. If restarting with a server parameter file in a nondefault location, then create a new client-side initialization parameter file with the single line `SPFILE=`*`new_location`*, where *`new_location`* is the path name of the restored server parameter file. Then, restart the instance with the client-side initialization parameter file.

   For example, create a file /tmp/init.ora which contains the single line:

   ```
   SPFILE=/tmp/spfileTEMP.ora
   ```

   Then use this RMAN command, to restart the instance based on the restored SPFILE:

   ```
   RMAN> STARTUP FORCE PFILE=/tmp/init.ora; # startup with /tmp/spfileTEMP.ora
   ```

Restore of the SPFILE from the Control File Autobackup

If you have configured control file autobackups, the SPFILE is backed up with the control file whenever an autobackup is taken.

If you want to restore the SPFILE from the autobackup, you must first set the DBID for your database, and then use the `RESTORE SPFILE FROM AUTOBACKUP` command. The procedure is similar to restoring the control file from autobackup. You must first set the DBID for your database, and then use the `RESTORE CONTROLFILE FROM AUTOBACKUP` command:

```
RMAN> SET DBID 320066378;
RMAN> RUN {
    SET CONTROLFILE AUTOBACKUP FORMAT
          FOR DEVICE TYPE DISK TO 'autobackup_format';
    RESTORE SPFILE FROM AUTOBACKUP;
    }
```

RMAN uses the autobackup format and DBID to hunt for control file autobackups, and if a control file autobackup is found, restores the SPFILE from that backup to its default location.

For information on how to determine the correct value for *`autobackup_format`*, see the description of `CONFIGURE CONTROLFILE AUTOBACKUP FORMAT` in the entry for `CONFIGURE` In *Oracle Database Recovery Manager Reference*

See "Determining your DBID" on page 5-8 for details on how to determine your DBID.

Creating a Client-Side Initialization Parameter File (PFILE) with RMAN

You can also restore the server parameter file as a client-side initialization parameter file with the `TO PFILE` *`'filename'`* clause. The filename you specify should be on a filesystem accessible from the host where the RMAN client is running. This file need not be accessible directly from the host running the instance. This command creates a PFILE called `/tmp/initTEMP.ora` on the system running the RMAN client:

```
RMAN> RESTORE SPFILE TO PFILE '/tmp/initTEMP.ora';
```

To restart the instance using the client-side PFILE, use the following command, again running RMAN on the same client machine:

```
RMAN> STARTUP FORCE PFILE='/tmp/initTEMP.ora';
```

Restoring and Recovering Datafiles and Tablespaces to a New Location

Restoring a tablespace to its original location is described in "Restore and Recovery of Individual Tablespaces or Datafiles: Scenario" on page 5-13. However, you may need to restore a datafile to a location other than its original location if, for example, the disk containing the original location of the datafiles has failed.

Restoring Datafiles from Backup to a New Location

The important step in restoring datafiles from backup to a new location is to update the control file to reflect the new locations of the datafiles. The following example shows the use of the RMAN `SET NEWNAME` command to specify the new names, and the `SWITCH` command to update the control file to start referring to the datafiles by their new names.

As with restoring datafiles from backup to their original locations, you should take the affected tablespaces offline at the start of restoring datafiles from backup to a new location.

Then, create a `RUN` block to encompass your `RESTORE` and `RECOVER` commands. For each file to be moved to a new location, use the `SET NEWNAME` command to specify the new location for that file.

Then, still within the RUN block, run the `RESTORE TABLESPACE` or `RESTORE DATAFILE` as normal. RMAN restores each datafile to the location specified with `SET NEWNAME`, rather than its original location.

After the `RESTORE` command but before the `RECOVER` command in your `RUN` block, use a `SWITCH` command to update the control file with the new filenames of the datafiles. The `SWITCH` command is equivalent to the SQL statement `ALTER DATABASE RENAME FILE`. `SWITCH DATAFILE ALL` updates the control file to reflect the new names for all datafiles for which a `SET NEWNAME` has been issued in the RUN block.

This example restores the datafiles in tablespaces `users` and `tools` to a new location, then performs recovery. Assume that the old datafiles were stored in directory `/olddisk` and the new ones will be stored in `/newdisk`.

```
RUN
{
  SQL 'ALTER TABLESPACE users OFFLINE IMMEDIATE';
  SQL 'ALTER TABLESPACE tools OFFLINE IMMEDIATE';
  # specify the new location for each datafile
  SET NEWNAME FOR DATAFILE '/olddisk/users01.dbf' TO
                           '/newdisk/users01.dbf';
  SET NEWNAME FOR DATAFILE '/olddisk/tools01.dbf' TO
                           '/newdisk/tools01.dbf';
  # to restore to an ASM disk group named dgroup, use:
  # SET NEWNAME FOR DATAFILE '/olddisk/trgt/tools01.dbf'
  #     TO '+dgroup';
  RESTORE TABLESPACE users, tools;
  SWITCH DATAFILE ALL;   # update control file with new filenames
  RECOVER TABLESPACE users, tools;
}
```

If recovery is successful, then bring the tablespaces online:

```
SQL 'ALTER TABLESPACE users ONLINE';
SQL 'ALTER TABLESPACE tools ONLINE';
```

> **See Also:** *Oracle Database Recovery Manager Reference* for `SWITCH` syntax

Performing Media Recovery of a Database, Tablespace or Datafile

Media recovery reapplies all changes from the archived and online redo logs and available incremental backups to datafiles restored from backup.

The simplest way to perform media reccovery is to use the `RECOVER DATABASE` command, with no arguments:

```
RMAN> RECOVER DATABASE;
```

You can also perform media recovery of individual tablespaces or datafiles, or skip certain tablespaces while recovering the rest of the database, as shown in the following examples:

```
RMAN> RECOVER DATABASE SKIP TABLESPACE users;

RMAN> RECOVER TABLESPACE users, tools;

RMAN> RECOVER DATAFILE '/newdisk/users01.dbf','/newdisk/tools01.dbf';

RMAN> RECOVER DATAFILE 4;
```

RMAN will restore from backup any archived redo logs required during the recovery operation. If backups are stored on a media manager, channels must be configured or allocated for use in accessing backups stored there.

One very useful option in managing disk space associated with these restored files is the `DELETE ARCHIVELOG` option, which causes the deletion of restored archived redo logs from disk once they are no longer needed for the `RECOVER` operation:

```
RMAN> RECOVER TABLESPACE users, tools DELETE ARCHIVELOG;
```

Note that when RMAN restores archived redo log files to the flash recovery area in order to perform a `RECOVER` operation, the restored logs are automatically deleted after they are applied to the datafiles, even if you do not use the `DELETE ARCHIVELOG` option.

See *Oracle Database Recovery Manager Reference* for more details on options for the `RECOVER` command.

Restore and Recover of Datafiles to a New Location

The procedure shown here is a convenient way to restore a datafile to a new location and perform media recovery on it.

```
RUN {
    SET NEWNAME FOR DATAFILE 3 to 'new_location';
    RESTORE DATAFILE 3;
    SWITCH DATAFILE 3;
    RECOVER DATAFILE 3;
}
```

If you want to store a datafile to a new Oracle Managed Files location, you can use this variation:

```
RUN {
    SET NEWNAME FOR DATAFILE 3 to NEW;
    RESTORE DATAFILE 3;
    SWITCH DATAFILE 3;
    RECOVER DATAFILE 3;
}
```

Oracle will store the restored file in an OMF location, generating a filename for it.

Point-in-Time Recovery of a Database or Tablespace

Point-in-time recovery lets you apply only those changes to your database or tablespace which occured before a particular moment in time. Use point-in-time recovery to undo unwanted changes to your database, or to recover as many changes as possible when you do not have a complete set of archived redo log files. See *Oracle Database Backup and Recovery Advanced User's Guide* for details on database point-in-time recovery, and *Oracle Database Backup and Recovery Advanced User's Guide* for details on tablespace point-in-time recovery.

Restoring Archived Redo Logs from Backup

RMAN will restore archived redo log files from backup automatically as needed to perform recovery.

However, you can restore archived redo logs manually if you wish, in order to save the time needed to restoroe these files later during the RECOVER command, or if you want to store the restored archived redo log files in some new location.

By default, RMAN restores archived redo logs with names constructed using the `LOG_ARCHIVE_FORMAT` and the `LOG_ARCHIVE_DEST_1` parameters of the target database. These parameters are combined in a platform-specific fashion to form the name of the restored archived log.

Restoring Archived Redo Logs to a New Location

You can override the default location for restored archived redo logs with the `SET ARCHIVELOG DESTINATION` command. This command manually stages archived logs to different locations while a database restore is occurring. During recovery, RMAN knows where to find the newly restored archived logs; it does not require them to be in the location specified in the initialization parameter file.

To restore archived redo logs to a new location:

1. After connecting to the target database, make sure the database is mounted or open.
2. Perform the following operations within a `RUN` block, as shown in the following example script:

   a. Specify the new location for the restored archived redo logs using `SET ARCHIVELOG DESTINATION`.

   b. Restore the archived redo logs.

   This example restores all backup archived logs to a new location:

   ```
   RUN
   {
     SET ARCHIVELOG DESTINATION TO '/oracle/temp_restore';
     RESTORE ARCHIVELOG ALL;
     # restore and recover datafiles as needed
     .
     .
     .
   }
   ```

Restoring Archived Redo Logs to Multiple Locations

You can specify restore destinations for archived logs multiple times in one RUN block, in order to distribute restored logs among several destinations. (You cannot, however specify multiple destinations simultaneously to produce multiple copies of the same log during the restore operation.) You can use this feature to manage disk space used to contain the restored logs.

This example restores 300 archived redo logs from backup, distributing them across the directories `/fs1/tmp`, `/fs2/tmp`, and `/fs3/tmp`:

```
RUN
{
  # Set a new location for logs 1 through 100.
  SET ARCHIVELOG DESTINATION TO '/fs1/tmp';
  RESTORE ARCHIVELOG FROM SEQUENCE 1 UNTIL SEQUENCE 100;
  # Set a new location for logs 101 through 200.
  SET ARCHIVELOG DESTINATION TO '/fs2/temp';
  RESTORE ARCHIVELOG FROM SEQUENCE 101 UNTIL SEQUENCE 200;
  # Set a new location for logs 201 through 300.
  SET ARCHIVELOG DESTINATION TO '/fs3/tmp';
  RESTORE ARCHIVELOG FROM SEQUENCE 201 UNTIL SEQUENCE 300;
  # restore and recover datafiles as needed
  .
  .
  .
}
```

When you issue a RECOVER command, RMAN finds the needed restored archived logs automatically across the multiple destinations to which they were restored, and applies them to the datafiles.

6 Recovery Manager Maintenance Tasks

This chapter describes how to manage the RMAN repository and some maintenance tasks related to the flash recovery area.

This chapter contains these topics:

See Also: *Oracle Database Backup and Recovery Advanced User's Guide* for more in-depth information about managing the RMAN repository when using a recovery catalog

Managing the RMAN Repository Without a Recovery Catalog

The authoritative RMAN repository is always stored in the database control file. The repository contents can also be stored in a recovery catalog database, as an adjunct to the information stored in the control file.

While RMAN is designed to work without a recovery catalog, if you choose not to use a recovery catalog, you must perform some additional administrative tasks.

> **See Also:** *Oracle Database Administrator's Guide* for a conceptual overview of the control file and more details about managing control files.

Backing Up and Restoring the Control File

If you are not using a recovery catalog, the control file is the sole storage for the RMAN repository, so it is doubly important that you protect it. Maintain alternate control files through multiplexing or operating system mirroring, and back up the control file frequently.

Configure `CONTROLFILE AUTOBACKUP` to `ON` to ensure extra protection for your control file.

So long as a control file autobackup is available, RMAN can restore the SPFILE and backup control file, and mount the database. After the control file is mounted, you can restore the remainder of the database.

Note that, if you restore a control file from autobackup, any persistent settings you set with the `CONFIGURE` command will revert to the values they had at the time of the control file autobackup. You should review the settings with the `SHOW ALL` after restoring the control file.

> **See Also:**
>
> - "Backing Up Control Files with RMAN" to learn about manual and automatic control file backups
> - *Oracle Database Backup and Recovery Advanced User's Guide* to learn how to restore a database when the current control file and recovery catalog are unavailable

Monitoring the Overwriting of Control File Records

When you do not use a recovery catalog, the control file is the sole source of information about RMAN backups. As you make backups, Oracle records these backups in the control file. To prevent the control file from growing without bound to hold RMAN repository data, records can be re-used if they are older than a threshhold you specify.

The `CONTROL_FILE_RECORD_KEEP_TIME` initialization parameter determines the minimum age in days of a record before it can be overwritten:

```
CONTROL_FILE_RECORD_KEEP_TIME = integer
```

For example, if the parameter value is `14`, then any record aged 14 days and older is a candidate for reuse. Information in an overwritten record is lost. The oldest record available for re-use will be used first.

When Oracle needs to add new RMAN repository records to the control file, but no record is older than the threshhold, Oracle attempts to expand the size of the control file. If the underlying operating system prevents the expansion of the control file (due to a disk full condition, for instance), Oracle overwrites the oldest record in the control file and logs this action in the alert log.

The default value of `CONTROL_FILE_RECORD_KEEP_TIME` is 7 days. If you are not using a recovery catalog, then set the `CONTROL_FILE_RECORD_KEEP_TIME` value to slightly longer than the oldest file that you need to keep. For example, if you back up the database once a week, then you need to keep every backup at least a week. Set `CONTROL_FILE_RECORD_KEEP_TIME` to a value such as `10` or `14`.

Caution: Regardless of whether you use a recovery catalog, never use RMAN when `CONTROL_FILE_RECORD_KEEP_TIME` is set to 0. If you do, then you may lose backup records.

Managing the Overwriting of Control File Records: Scenario

Assume the following scenario:

- You do not use a recovery catalog.
- `CONTROL_FILE_RECORD_KEEP_TIME` is set to `14`.
- All records currently in the control file are between 1 and 13 days old.
- The control file is at the maximum size permitted by the operating system.

You make a backup of the database. Because Oracle cannot expand the control file beyond the operating system file size limit, it begins overwriting records in the control file, starting with those records aged 13 days. For each record that it overwrites, it records an entry in the `alert.log` similar to the one shown here:

```
kccwnc: following controlfile record written over:
RECID #72 Recno 72 Record timestamp
07/28/00 22:15:21
Thread=1 Seq#=3460
Backup set key: stamp=372031415, count=17
Low scn: 0x0000.3af33f36
07/27/00 21:00:08
Next scn: 0x0000.3af3871b
07/27/00 23:23:54
Resetlogs scn and time
scn: 0x0000.00000001
08/05/99 10:46:44
Block count=102400 Blocksize=512
```

To guard against this type of scenario, use a recovery catalog. If you cannot use a recovery catalog, then do the following if possible:

- Store the control file in a file system rather than raw disk, so that it can expand as needed.
- Monitor the `alert.log` to make sure that Oracle is not overwriting control file records.

> **See Also:** *Oracle Database Backup and Recovery Advanced User's Guide* for a conceptual overview of control file records and how they are re-used

Interaction of Flash Recovery Area and CONTROL_FILE_RECORD_KEEP_TIME

Whe a control file record containing information about a file created in the flash recovery area is about to be reused (because the record is older than `CONTROL_FILE_RECORD_KEEP_TIME`), if the file is eligible for deletion then the database will attempt to delete the file from the flash recovery area. Otherwise, Oracle will expand the size of the control file section containing the record for this file, logging the expansion in the alert log with a message like this example:

```
kccwnc: tring to expand controlfile section nnnn for Oracle Managed Files
```

where *nnnn* is the record type number.

If Oracle is unable to expand the control file section, because the control file is at the maximum size supported under the host operating system, you will see this warning in the alert log:

```
WARNING: Oracle Managed File filename is unknown to controlfile. This is the
result of limitation in control file size that could not keep all recovery area
```

files.

This means that the control file cannot hold all flash recovery area files needed to satisfy the configured retention policy.

There are several ways to avoid or alleviate this problem:

- Use a control file of larger block size, preferably one with 32K block size. To achieve this, you must set the SYSTEM tablespace block size to be greater than or equal to the control file block size, and you need to re-create the control file after changing `DB_BLOCK_SIZE`.
- Make the files in the flash recovery area eligible for deletion, by backing them up to tertiary storage such as tape with the RMAN command `BACKUP RECOVERY AREA`, or by changing the retention policy to a shorter recovery window or lower degree of redundancy.

Maintaining the RMAN Repository in the Control File

RMAN provides several commands that enable you to check and delete records of backups as well as physically remove backups.

Crosschecking Backups

To ensure that data about backups in the recovery catalog or control file is synchronized with corresponding data on disk or in the media management catalog, perform a **crosscheck**. The `CROSSCHECK` command operates only on files that are recorded in the recovery catalog or the control file.

This section contains these topics:

- About RMAN Crosschecks
- Crosschecking Specific Backup Sets and Copies
- Crosschecking Backups of Specific Database Files

About RMAN Crosschecks

Crosschecks update outdated RMAN repository information about backups whose repository records do not match their physical status. For example, if a user removes archived logs from disk with an operating system command, the repository still indicates that the logs are on disk, when in fact they are not.

If the backup is on disk, then the `CROSSCHECK` command determines whether the header of the file is valid. If the backup is on tape, then the command simply checks that the backup exists. The possible status values for backups are AVAILABLE, UNAVAILABLE, and EXPIRED. View the status of backups in one of the following locations:

- The LIST command output
- V$BACKUP_FILES
- Various recovery catalog views such as RC_DATAFILE_COPY, RC_ARCHIVED_LOG, and so forth

Note: The CROSSCHECK command does *not* delete operating system files or remove repository records. You must use the DELETE command for these operations.

See Also:

- "Deleting Backups" on page 6-7 to learn how to delete files and update repository records
- *Oracle Database Recovery Manager Reference* for CROSSCHECK command syntax and a description of the repository status values

Crosschecking Specific Backup Sets and Copies

You can use the LIST command to report your backups and then use the CROSSCHECK command to check that these files still exist. The DELETE EXPIRED command deletes repository records for backups that fail the crosscheck.

To crosscheck specified backups:

1. Identify the desired backups that you want to check by issuing a LIST command. For example, issue:

```
LIST BACKUP;  # lists all backup sets, proxy copies, and image copies
```

2. Check whether the specified backups. For example, enter:

```
CROSSCHECK BACKUP;  # checks backup sets, proxy copies, and image copies
CROSSCHECK COPY OF DATABASE;
CROSSCHECK BACKUPSET 1338, 1339, 1340;
CROSSCHECK BACKUPPIECE TAG = 'nightly_backup';
CROSSCHECK CONTROLFILECOPY '/tmp/control01.ctl';
CROSSCHECK DATAFILECOPY 113, 114, 115;
CROSSCHECK PROXY 789;
```

If the backup is no longer available, then RMAN marks it as `EXPIRED`. If it was `EXPIRED` and is now available, then RMAN marks it `AVAILABLE`.

Crosschecking Backups of Specific Database Files

Use the `LIST` command to display backups, then use the `CROSSCHECK` command to check whether these backups exist.

To crosscheck backups of specified files:

Check for the backups of the specified database, tablespace, datafile, control file, or archived redo log. Limit the crosscheck according to the time sequence. For example, check all backups of datafile *ora_home*`/oradata/trgt/system01.dbf` over the last six months, as well as all archived logs and server parameter files:

```
# these CROSSCHECK commands use configured channels, which means that they
# always check the disk device. If you configured an sbt channel, then RMAN
# checks the sbt device, too
CROSSCHECK BACKUP OF DATAFILE "ora_home/oradata/trgt/system01.dbf"
  COMPLETED AFTER 'SYSDATE-180';
CROSSCHECK BACKUP OF ARCHIVELOG ALL SPFILE;
```

See Also: *Oracle Database Recovery Manager Reference* for `CROSSCHECK` command syntax

Deleting Backups

You can use RMAN to delete backups (backup sets and disk copies of datafiles and archived logs). RMAN deletes the specified files and removes their repository records.

This section contains these topics:

- Deleting Specified Backups
- Deleting Expired Backups
- Deleting Obsolete Backups

Note: Backups with `DELETED` status do not appear in the `LIST` command output. Query the `V$` control file views instead.

See Also:

- *Oracle Database Recovery Manager Reference* for DELETE syntax
- *Oracle Database Recovery Manager Reference* for descriptions of the recovery catalog views

Deleting Specified Backups

In general, use the DELETE command to remove backups that you no do not want to retain. This command removes the physical files, deletes the catalog records (if you use a catalog), and updates the records in the control file to status DELETED.

To delete backups and remove their repository records:

1. Run the LIST output to obtain primary keys of backups. For example:

   ```
   LIST BACKUP OF DATABASE ARCHIVELOG ALL; # lists backups of db files and logs
   LIST COPY; # lists only image copies
   LIST BACKUP; # lists everything
   ```

2. Run the DELETE command to eliminate the specified physical files and their repository records. For example:

   ```
   DELETE BACKUPPIECE 101;
   DELETE CONTROLFILECOPY '/tmp/control01.ctl';
   DELETE NOPROMPT ARCHIVELOG UNTIL SEQUENCE = 300;
   ```

3. You can also delete files with DELETE BACKUP (which deletes backup sets, proxy copies, and image copies), DELETE COPY (which deletes only image copies), or DELETE ARCHIVELOG as in these examples:

   ```
   DELETE BACKUP; # deletes all backups on disk and tape
   DELETE BACKUP OF TABLESPACE users DEVICE TYPE sbt; # delete only from tape
   DELETE COPY OF CONTROLFILE LIKE '/tmp/%';  # LIKE specifies name of the copy
   DELETE NOPROMPT ARCHIVELOG ALL
      BACKED UP 3 TIMES TO sbt; # backs up only if already backed up 3X to tape
   ```

   If you run RMAN interactively, then RMAN asks for confirmation before deleting any files. You can suppress these confirmations by using the NOPROMPT keyword:

   ```
   DELETE NOPROMPT ARCHIVELOG ALL
      BACKED UP 3 TIMES TO sbt; # backs up only if already backed up 3X to tape
   ```

Deleting Expired Backups

Use the CROSSCHECK command to determine whether backups recorded in the repository still exist on disk or tape. If RMAN cannot locate the backups, then it updates their records to EXPIRED status. You can then use the DELETE EXPIRED command to remove expired records. If the expired files still exist, then the DELETE EXPIRED command terminates with an error.

To delete expired repository records:

1. If you have not performed a crosscheck recently, then issue a CROSSCHECK command. For example, issue:

   ```
   CROSSCHECK BACKUP; # checks backup sets and copies on configured channels
   ```

2. Delete the expired backups. For example, issue:

   ```
   DELETE EXPIRED BACKUP;
   ```

Deleting Obsolete Backups

Use the DELETE OBSOLETE command to remove backups that are obsolete, that is, eligible for deletion. You can determine what qualifies a backup for obsolete status in these ways:

- By configuring a retention policy with the CONFIGURE command
- By using the options on the DELETE OBSOLETE command

The DELETE OBSOLETE command removes both the physical files, deletes the recovery catalog records (if you use a catalog), and updates the records in the target control file to status DELETED.

Deleting Backups Rendered Obsolete by the Retention Policy If you specify the DELETE OBSOLETE command with no other operands, then RMAN deletes all obsolete backups defined by the retention policy.

To delete backups made obsolete by the retention policy:

Issue a DELETE OBSOLETE command without any options to delete the objects defined as obsolete by the retention policy, for example:

```
DELETE OBSOLETE;
```

If you run RMAN interactively, then RMAN asks for confirmation before deleting any files. If you specify NOPROMPT, then RMAN does not ask for confirmation.

Deleting Backups Defined as Obsolete with the DELETE Command You can run the `DELETE OBSOLETE REDUNDANCY` or `DELETE OBSOLETE RECOVERY WINDOW` commands to delete obsolete backups. The redundancy or recovery window setting on the `DELETE` command overrides setting on the `CONFIGURE RETENTION POLICY` command.

To specify which backups are obsolete and should be deleted:

Run a `DELETE OBSOLETE` command with the `REDUNDANCY` or `RECOVERY WINDOW` options. These options define what is obsolete during the delete job. For example, enter:

```
DELETE OBSOLETE REDUNDANCY = 3;
DELETE OBSOLETE RECOVERY WINDOW OF 7 DAYS;
```

Forcing the Deletion of Backups

It is possible for the RMAN repository to indicate that an object has one status while the actual status of the object on the media is different. For example, the RMAN repository says that a backup set is `AVAILABLE` when it is in fact missing from the media management catalog. If you attempt to delete the object, then you receive a warning such as the following:

```
RMAN-06207: WARNING: 1 objects could not be deleted for DISK channel(s) due
RMAN-06208:          to mismatched status.  Use CROSSCHECK command to fix status
List of Mismatched objects
==========================
  Object Type   Filename/Handle
--------------- ---------------------------------------------------
Backup Piece    0id270ud_1_1
```

You can force RMAN to delete any object and update its status in the RMAN repository by specifying the `FORCE` keyword. RMAN ignores any I/O errors. For example:

```
DELETE FORCE NOPROMPT BACKUPSET TAG 'weekly_bkup';
```

See Also:

- *Oracle Database Backup and Recovery Advanced User's Guide* to learn about `DELETE` behavior when the backup media and repository are not synchronized
- *Oracle Database Recovery Manager Reference* for `DELETE` command syntax

Crosschecking and Deleting on Multiple RMAN Channels

This section contains these topics:

About Allocating Multiple RMAN Channels for Maintenance Commands

You can configure or manually allocate multiple maintenance channels before issuing `CROSSCHECK` or `DELETE` commands. RMAN searches for each backup on all channels that have the same device type as the channel used to create the backup. The multichannel feature is designed for use in these scenarios:

- To crosscheck or delete of all backups, both on disk and tape, with a single command;
- To make crosschecking and deleting work correctly in an Oracle Real Application Cluster (RAC) configuration in which each backup exists only on one node.

How RMAN Crosschecks and Deletes on Multiple Channels

When you configure or manually allocate multiple maintenance channels and run a `CROSSCHECK` or `DELETE` command, RMAN performs the crosscheck or delete on all channels that have the appropriate device type.

For example, assume that you prefer to manually allocate channels rather than use configured channels. You have a media manager configured, but have not yet made backups to tape. You have created only one backup of a database to disk, as follows:

```
RUN
{
  ALLOCATE CHANNEL ch1 DEVICE TYPE DISK CONNECT 'SYS/sys_pwd@node2';
  BACKUP DATABASE;
}
```

Assume that you issue the following series of commands at the RMAN prompt:

```
ALLOCATE CHANNEL FOR MAINTENANCE DEVICE TYPE DISK CONNECT 'SYS/oracle@node1';
AllOCATE CHANNEL FOR MAINTENANCE DEVICE TYPE DISK CONNECT 'SYS/oracle@node2';
ALLOCATE CHANNEL FOR MAINTENANCE DEVICE TYPE sbt;
CROSSCHECK BACKUP OF DATABASE;
```

RMAN checks the first two channels because they both have the device type of disk and finds the backup on the second channel. However, RMAN does not perform a crosscheck on the third `sbt` channel because you have not yet made backups with a media manager.

Note: The `DELETE EXPIRED` command issues warnings if any objects marked as `EXPIRED` actually exist. In rare cases, the repository can mark an object as `EXPIRED` even though the object exists. For example, a directory containing an object is corrupted at the time of the crosscheck, but is later repaired, or the media manager was not configured properly and reported some backups as not existing when they really existed.

Crosschecking Disk and Tape Channels with One Command: Example

RMAN can perform crosschecks on more than one media with a single command. Assume that you have an `sbt` channel configured as follows:

```
CONFIGURE DEVICE TYPE sbt PARALLELISM 1;
CONFIGURE DEFAULT DEVICE TYPE sbt;
```

In this case you can run the following command to perform a crosscheck on both `DISK` and `sbt`:

```
CROSSCHECK BACKUP OF DATABASE;
```

RMAN uses both the `sbt` channel and the preconfigured `DISK` channel to perform the crosscheck. Sample output follows:

```
allocated channel: ORA_SBT_TAPE_1
channel ORA_SBT_TAPE_1: sid=12 devtype=SBT_TAPE
channel ORA_SBT_TAPE_1: WARNING: Oracle Test Disk API
using channel ORA_DISK_1
crosschecked backup piece: found to be 'AVAILABLE'
backup piece handle=/oracle/dbs/16c5esv4_1_1 recid=36 stamp=408384484
crosschecked backup piece: found to be 'AVAILABLE'
backup piece handle=/oracle/dbs/c-674966176-20000915-01 recid=37 stamp=408384496
```

```
crosschecked backup piece: found to be 'AVAILABLE'
backup piece handle=12c5erb2_1_1 recid=32 stamp=408382820
crosschecked backup piece: found to be 'AVAILABLE'
backup piece handle=13c5erba_1_1 recid=33 stamp=408382829
crosschecked backup piece: found to be 'AVAILABLE'
backup piece handle=14c5erce_1_1 recid=34 stamp=408382863
crosschecked backup piece: found to be 'AVAILABLE'
backup piece handle=c-674966176-20000915-00 recid=35 stamp=408382869
```

If you do not have an automatic `sbt` channel configured, then can also manually allocate maintenance channels on disk and tape as in the following example:

```
ALLOCATE CHANNEL FOR MAINTENANCE DEVICE TYPE sbt;
CROSSCHECK BACKUP OF DATABASE;
```

Note that you do not have to manually allocate a disk channel because RMAN uses the preconfigured disk channel.

Crosschecking on Multiple Oracle Real Application Cluster Nodes: Example

This feature is useful in an Oracle Real Application Cluster (RAC) configuration in which tape backups exist on various nodes in the cluster and are only visible on the nodes where they were created.

When crosschecking on multiple nodes, it is important to allocate channels at every node where backups were created. If you omit a channel for a node, or you set the parallelism to a value less than the number of nodes, then the backups created on that node will be marked `EXPIRED` in the repository. For example, you can configure RMAN for use with RAC nodes as follows:

```
CONFIGURE DEVICE TYPE DISK PARALLELISM 2;
CONFIGURE CHANNEL 1 DEVICE TYPE DISK CONNECT 'SYS/oracle@node_1';
CONFIGURE CHANNEL 2 DEVICE TYPE DISK CONNECT 'SYS/oracle@node_2';
```

Then, crosscheck the cluster nodes with the following command:

```
CROSSCHECK BACKUP;
```

Deleting on Disk and Tape Channels with One DELETE Command: Example

You can also perform deletions on all allocated channels. In the following example, you configure an `sbt` channel:

```
CONFIGURE DEVICE TYPE sbt PARALLELISM 1;
CONFIGURE DEFAULT DEVICE TYPE TO sbt;
```

Then, you run a command to delete all backup sets from disk and tape:

```
DELETE BACKUPSET;
```

RMAN uses both the `sbt` channel and the preconfigured `DISK` channel when deleting (sample output follows). Note that RMAN prompts you for confirmation before deleting any files:

```
using channel ORA_SBT_TAPE_1
using channel ORA_DISK_1

List of Backup Pieces
BP Key  BS Key  Pc# Cp# Status      Device Type Piece Name
------- ------- --- --- ----------- ----------- ----------
388     387     1   1   AVAILABLE   SBT_TAPE    12c5erb2_1_1
397     396     1   1   UNAVAILABLE SBT_TAPE    13c5erba_1_1
424     423     1   1   AVAILABLE   SBT_TAPE    14c5erce_1_1
428     427     1   1   AVAILABLE   SBT_TAPE    c-674966176-20000915-00
433     432     1   1   AVAILABLE   DISK        /oracle/dbs/16c5esv4_1_1
437     436     1   1   AVAILABLE   DISK     /oracle/dbs/c-674966176-20000915-01

Do you really want to delete the above objects (enter YES or NO)? y
deleted backup piece
backup piece handle=/oracle/dbs/16c5esv4_1_1 recid=36 stamp=408384484
deleted backup piece
backup piece handle=/oracle/dbs/c-674966176-20000915-01 recid=37 stamp=408384496
deleted backup piece
backup piece handle=12c5erb2_1_1 recid=32 stamp=408382820
deleted backup piece
backup piece handle=13c5erba_1_1 recid=33 stamp=408382829
deleted backup piece
backup piece handle=14c5erce_1_1 recid=34 stamp=408382863
deleted backup piece
backup piece handle=c-674966176-20000915-00 recid=35 stamp=408382869
```

The following example manually allocates `DISK` and `sbt` maintenance channels and then deletes specific backup sets from both disk and tape:

```
ALLOCATE CHANNEL FOR MAINTENANCE DEVICE TYPE DISK;
ALLOCATE CHANNEL FOR MAINTENANCE DEVICE TYPE sbt;
DELETE BACKUPSET 1,2,3,4,5;
```

RMAN looks for the specified backup sets on the channels and deletes any it finds. If RMAN does not find a backup on any channel, then RMAN marks the object as deleted in the control file and deletes the recovery catalog record (if you use a recovery catalog).

Releasing Multiple Channels: Example

You can release all allocated maintenance channels by running this command:

```
RELEASE CHANNEL;
```

Deleting a Database with RMAN

You may need to remove a database from the operating system. For example, you create a test database and then no longer have a use for it.

The DROP DATABASE command requires that RMAN be connected to the target database and that the target database be mounted. The command does not require connection to the recovery catalog. If RMAN is connected to the recovery catalog, and if you specify the option INCLUDE COPIES AND BACKUPS, then RMAN also unregisters the database.

> **See Also:** *Oracle Database Backup and Recovery Advanced User's Guide* to learn how to use the SQL*Plus DROP DATABASE command

To drop the database:

1. Connect RMAN to the target database and (optionally) recovery catalog. For example:

   ```
   rman TARGET / CATALOG rman/rman@catdb
   ```

2. Catalog all backups that are associated with the database. For example, the following commands catalogs files in the flash recovery area, and then in a secondary archiving destination:

   ```
   RMAN> CATALOG START WITH '+disk1';    # all files from flash recovery area
                                         # (stored on ASM disk)
   RMAN> CATALOG START WITH '/arch_dest2';  # all files from second arch dest
   ```

3. Delete all backups and copies associated with the database. For example:

   ```
   RMAN> DELETE BACKUPSET; # deletes all backups
   RMAN> DELETE COPY; # delete all image copies (including archived logs)
   ```

4. Remove the database from the operating system (and automatically unregister it from the recovery catalog if you are connected to the catalog). For example:

```
DROP DATABASE; # delete all database files and unregister the database
```

Changing the Status of a Backup Record

This section contains the following sections:

- Marking a Backup AVAILABLE or UNAVAILABLE
- Exempting a Backup from the Retention Policy

Marking a Backup AVAILABLE or UNAVAILABLE

Run the `CHANGE ... UNAVAILABLE` command when a backup cannot be found or has migrated offsite. RMAN does not use files marked `UNAVAILABLE` in `RESTORE` or `RECOVER` commands. If the file is later found or returns to the main site, then you can mark it available again by issuing `CHANGE ... AVAILABLE`.

Note that files in the flash recovery area cannot be marked as `UNAVAILABLE`.

To mark a file's status in the repository as UNAVAILABLE or AVAILABLE:

1. Issue a `LIST` command to determine the availability status of RMAN backups. For example, issue:

```
LIST BACKUP;
```

2. Run `CHANGE` with the `UNAVAILABLE` or `AVAILABLE` keyword to update its status in the RMAN repository. For example, enter:

```
CHANGE DATAFILECOPY '/tmp/control01.ctl' UNAVAILABLE;
CHANGE COPY OF ARCHIVELOG SEQUENCE BETWEEN 1000 AND 1012 UNAVAILABLE;
CHANGE BACKUPSET 12 UNAVAILABLE;
CHANGE BACKUP OF SPFILE TAG "TAG20020208T154556" UNAVAILABLE;
CHANGE DATAFILECOPY '/tmp/system01.dbf' AVAILABLE;
CHANGE BACKUPSET 12 AVAILABLE;
CHANGE BACKUP OF SPFILE TAG "TAG20020208T154556" AVAILABLE;
```

See Also: *Oracle Database Recovery Manager Reference* for `CHANGE` command syntax

Exempting a Backup from the Retention Policy

The BACKUP ... KEEP command can create a backup that is retained for a different period of time from that specified by the configured retention policy. The backup is still a fully valid backup, however, and can be restored just as any other RMAN backup. This type of backup is called a **long-term backup**.

Note: The KEEP FOREVER clause requires the use of a recovery catalog.

Specify KEEP ... LOGS to save archived logs for a possible incomplete recovery and KEEP ... NOLOGS not to save archived logs for a possible incomplete recovery. Note that NOLOGS is not valid with an inconsistent backup.

Use the CHANGE command to alter the KEEP status of a backup that already exists. For example, you may decide that you no longer want to keep a long-term backup. The same options available for BACKUP ... KEEP are available with CHANGE ... KEEP.

Note that you cannot set KEEP attributes on files stored in the flash recovery area.

To alter the KEEP status of a backup:

Issue CHANGE ... KEEP to define a different retention period for this backup, or CHANGE ... NOKEEP to let the retention policy apply to this file.

This example allows a backup set to be marked obsolete by the retention policy:

```
CHANGE BACKUPSET 231 NOKEEP;
```

This example makes a datafile copy exempt from the retention policy for 180 days (6 months):

```
CHANGE DATAFILECOPY '/tmp/system01.dbf' KEEP UNTIL 'SYSDATE+180';
```

Cataloging Archived Logs and User-Managed Copies

You can make RMAN aware of the existence of archived logs that are not recorded in the repository as well as file and backup piece copies that are created through means other than RMAN. This section contains the following topics:

- About Cataloging Archived Logs and User-Managed Copies
- Cataloging User-Managed Datafile Copies

- Cataloging Backup Pieces
- Cataloging All Files in a Disk Location

About Cataloging Archived Logs and User-Managed Copies

The target database control file keeps records of all archived logs generated by the target database as well as all RMAN backups. The purpose of the `CATALOG` command is to add metadata to the repository when it does not have a record of files that you want RMAN to know about.

Run the RMAN `CATALOG` command when:

- You use an operating system utility to make copies of datafiles, archived logs, or backup pieces. In the case, the repository has no record of them.
- You perform recovery with a backup control file and you change the archiving destination or format during recovery. In this situation, the repository will not have information about archived logs needed for recovery. Hence, you must catalog these logs.
- You want to catalog datafile copy as a level 0 backup, thus enabling you to perform an incremental backup later by using the datafile copy as the base of an incremental backup strategy
- You want to catalog user-managed copies of Oracle7 database files created before you migrated to a higher release, or of Oracle8 and higher database files created before you started to use RMAN. These datafile copies enable you to recover the database if it crashes after migration but before you have a chance to take a backup of the migrated database.

Whenever you make a user-managed copy, for example, by using the UNIX `cp` command to copy a datafile, make sure to catalog it. When making user-managed copies, you can use the `ALTER TABLESPACE ... BEGIN/END BACKUP` statement to make datafile copies off an online tablespace. Although RMAN does not create such datafile copies, you can use the `CATALOG` command to add them to the recovery catalog so that RMAN is aware of them.

For a user-managed copy to be cataloged, it must be:

- Accessible on disk
- A complete image copy of a single file
- Either a datafile copy, control file copy, archived redo log copy, or backup piece copy

For example, if you store datafiles on mirrored disk drives, then you can create a user-managed copy by breaking the mirror. In this scenario, use the `CATALOG` command to notify RMAN of the existence of the user-managed copy after breaking the mirror. Before reforming the mirror, run a `CHANGE ... UNCATALOG` command to notify RMAN that the file copy no longer exists.

Cataloging User-Managed Datafile Copies

Use the `CATALOG` command to propagate information about user-managed copies to the RMAN repository. After the files are cataloged, you can run `LIST` or query `V$BACKUP_FILES` to confirm.

To create and catalog a user-managed copy of a datafile:

1. Make a datafile copy with an operating system utility. `ALTER TABLESPACE BEGIN`/`END BACKUP` is necessary if the database is open and the datafiles are online while the backup is in progress. This example backs up an online datafile.

```
SQL> ALTER TABLESPACE users BEGIN BACKUP;
% cp $ORACLE_HOME/oradata/trgt/users01.dbf /tmp/users01.dbf;
SQL> ALTER TABLESPACE users END BACKUP;
```

2. After connecting to the target database and, if desired, the recovery catalog, run the `CATALOG` command. For example, enter:

```
CATALOG DATAFILECOPY '/tmp/users01.dbf';
```

See Also: *Oracle Database Recovery Manager Reference* for `CATALOG` command syntax

Cataloging Backup Pieces

You can catalog backup pieces on disk. This technique is useful if you use an operating system utility to copy backup pieces from location to another on the same host, or from one host to another. You can even catalog a backup piece from a prior incarnation of the database. After a backup piece is cataloged, you can display its metadata by querying `V$BACKUP_PIECE`, `V$BACKUP_SET`, `V$BACKUP_DATAFILE`, `V$BACKUP_REDOLOG`, and `V$BACKUP_SPFILE`.

To catalog a backup piece:

1. After connecting RMAN to the target database, catalog the filenames of the backup pieces. For example:

```
CATALOG BACKUPPIECE '/disk2/09dtq55d_1_2', '/disk2/0bdtqdou_1_1';
```

Note: If you are cataloging backup pieces from prior to Oracle9*i*, you can achieve significant performance gains by cataloging the highest copy numbers first. Otherwise, RMAN must examine all pieces to determine the correct order. For example, catalog copy 3 of a piece before copy 2:

```
CATALOG BACKUPPIECE '/disk2/09dtq55d_1_3',
'/disk2/09dtq55d_1_2';
```

Backup pieces from Oracle Database Release 10*g* are not affected by this issue and can be cataloged in any order.

---

2. You can query `V$` views to verify your changes. For example:

   ```
   SELECT HANDLE FROM V$BACKUP_PIECE;
   ```

   See Also: *Oracle Database Recovery Manager Reference* for `CATALOG BACKUPPIECE` restrictions

Cataloging All Files in a Disk Location

If you use Automatic Storage Management (ASM), an Oracle Managed Files framework, or the flash recovery area, then you may need a way to recatalog files that are known to the disk management system but are no longer listed in the RMAN repository. Files can be orphaned when the intended mechanisms for tracking filenames fails due to media failure, software bug, or user error.

The `CATALOG START WITH` command enables you to search through all files in an ASM disk group, Oracle Managed Files location, or traditional file system directory and investigate those that are not recorded in the RMAN repository. If the command can catalog a file, then it will; if it cannot catalog it, then it makes its best guess about the contents of the skipped file.

The `CATALOG RECOVERY AREA` command will catalog all files in the flash recovery area. If there are any orphaned files in the recovery area, they will be added to the RMAN repository.

To catalog all files in a disk location:

After connecting RMAN to the target database, specify the disk location whose files you want to catalog. For example:

```
RMAN> CATALOG RECOVERY AREA; # catalog all files in the recovery area
```

```
RMAN> CATALOG START WITH '+disk'; # catalog all files from an ASM disk group
RMAN> CATALOG START WITH '/fs1/datafiles/'; # catalog all files in directory
```

Note: Wildcard characters are not legal in the `START WITH` clause.

Uncataloging RMAN Records

This section contains the following topics:

- About Uncataloging RMAN Records
- Removing Records for Files Deleted with Operating System Utilities

About Uncataloging RMAN Records

Run the `CHANGE ... UNCATALOG` command to perform the following actions on RMAN repository records:

- Update a backup record in the control file repository to status `DELETED`
- Delete a specific backup record from the recovery catalog (if you use one)

RMAN does not touch the specified physical files: it only alters the repository records for these files.

You can use this command when you have deleted a backup through a means other than RMAN. For example, if you delete archived redo logs with an operating system utility, then remove the record for this log from the repository by issuing `CHANGE ARCHIVELOG ... UNCATALOG`.

Removing Records for Files Deleted with Operating System Utilities

To remove catalog records for files deleted with operating system utilities, run the `CHANGE ... UNCATALOG` command.

To remove the catalog record for a backup deleted with an operating system utility:

1. Run a `CHANGE ... UNCATALOG` command for the backups that you deleted from the operating system with operating system commands. This example deletes repository references to disk copies of the control file and datafile `1`:

   ```
   CHANGE CONTROLFILECOPY '/tmp/control01.ctl' UNCATALOG;
   CHANGE DATAFILECOPY '/tmp/system01.dbf' UNCATALOG;
   ```

2. Optionally, view the relevant recovery catalog view, for example, RC_DATAFILE_COPY or RC_CONTROLFILE_COPY, to confirm that a given record was removed. For example, this query confirms that the record of copy 4833 was removed:

```
SELECT CDF_KEY, STATUS
FROM RC_DATAFILE_COPY
WHERE CDF_KEY = 4833;

CDF_KEY    STATUS
---------- ------
0 rows selected.
```

Flash Recovery Area Maintenance

While the flash recovery area is largely self-managing, there are some situations in which DBA intervention may be required.

Resolving a Full Flash Recovery Area

You have a number of choices on how to resolve a full flash recovery area when there are no files eligible for deletion:

- Make more disk space available, and increase DB_RECOVERY_FILE_DEST_SIZE to reflect the new space.
- Use the command BACKUP RECOVERY AREA, to back up the contents of the flash recovery area to a tertiary device such as tape.
- Delete unnecessary files from the flash recovery area using the RMAN delete command. (Note that if you use host operating system commands to delete files, then the database will not be aware of the resulting free space. You can run the RMAN CROSSCHECK command to have RMAN re-check the contents of the flash recovery area and identify expired files, and then use the DELETE EXPIRED command to remove missing files from the RMAN repository.)

You may also need to consider changing your backup retention policy and, if using Data Guard, consider changing your archivelog deletion policy.

See Also:

Chapter 4, "Making Backups with Recovery Manager" for more on how to decide on a retention policy, and *Oracle Data Guard Concepts and Administration* for more on archivelog deletion policy with Data Guard.

Changing the Flash Recovery Area to a New Location

If you need to move the flash recovery area of your database to a new location, you can follow this procedure:

1. Invoke SQL*Plus to change the `DB_RECOVERY_FILE_DEST` initialization parameter. For example:

   ```
   ALTER SYSTEM SET DB_RECOVERY_FILE_DEST='+disk1' SCOPE=BOTH SID='*';
   ```

   After you change this parameter, all new flash recovery area files will be created in the new location.

2. The permanent files (control files and online redo log files), flashback logs and transient files can be left in the old flash recovery area location. The database will delete the transient files from the old flash recovery area location as they become eligible for deletion.

   If you need to actually move your current permanent files, transient files, or flashback logs to the new flash recovery area, see *Oracle Database Backup and Recovery Advanced User's Guide* . The process outlined there for moving database files into and out of an ASM disk group with RMAN will also work when moving files into and out of a flash recovery area location.

   See Also: "Backing Up to the Flash Recovery Area and to Tape: Basic Scenarios" on page 4-32

Oracle will clean up transient files remaining in the old flash recovery area location as they become eligible for deletion.

Flash Recovery Area Behavior When Instance Crashes During File Creation

As a rule, the flash recovery area is self-maintaing, but when an instance crashes during the creation of a file in the flash recovery area, Oracle may leave the file in the flash recovery area. When this occurs, you will see the following error in the alert log:

```
ORA-19816: WARNING: Files may exist in location that are not known to database.
```

where *`location`* is the location of the flash recovery area.

In such a situation, you should use the RMAN command `CATALOG RECOVERY AREA` to re-catalog any such files so that they appear in the RMAN repository. If the file header of the file in question is corrupted, then you will have to delete the file manually using an operating system-level utility.

Glossary

archived redo log

A copy of one of the filled members of an online redo log group made when the database is in `ARCHIVELOG` mode. After the LGWR process fills each online redo log with redo records, the archiver process copies the log to one or more offline redo log archiving destinations. This copy is the archived redo log. Note that RMAN does not distinguish between an original archived redo log and an image copy of an archived redo log; both are considered image copies.

ARCHIVELOG mode

The mode of the database in which Oracle copies filled online redo logs to disk. Specify the mode at database creation or with the `ALTER DATABASE ARCHIVELOG` statement. Oracle automatically performs the archiving unless you execute `ALTER DATABASE ARCHIVELOG MANUAL`.

See Also: archived redo log, NOARCHIVELOG mode

archiving

The operation in which the archiver background process copies filled online redo logs to offline destinations. An offline copy of an online redo logs is called an **archived redo log**. You must run the database in `ARCHIVELOG` mode to archive redo logs.

automatic channel allocation

The persistent preconfiguration of RMAN channels. You can use the `CONFIGURE` command to specify disk and tape channels. Then, you can issue commands such as `BACKUP` and `RESTORE` at the RMAN command prompt without manually allocating channels. RMAN uses whatever preallocated channels that it needs in order to execute the commands.

automatic undo management mode

A mode of the database in which undo data is stored in a dedicated **undo tablespace**. The only undo management that you must perform is the creation of the undo tablespace. All other undo management is performed automatically.

auxiliary database

(1) A database created from target database backups with the RMAN `DUPLICATE` command.

(2) A temporary database that is restored to a new location and then started up with a new instance name during tablespace point-in-time recovery (TSPITR). A TSPITR auxiliary database contains the recovery set and auxiliary set.

See Also: **recovery set**, **auxiliary set**

auxiliary set

In TSPITR, the set of files that is not in the recovery set but which must be restored in the auxiliary database for the TSPITR set to be successful.

See Also: **auxiliary database**, **recovery set**

backup

(1) A backup of data, that is, a database, tablespace, table, datafile, control file, or archived redo log. You can make a backup by:

- Using RMAN to back up one or more datafiles, control files, or archived logs
- Making a copy either to disk or to tape using operating system utilities
- Exporting one or more tables with an Oracle export utility, called a logical or object-level backup

(2) An RMAN command that creates a backup set, proxy copy, or disk-based image copy.

See Also: **copy**, **backup set**, **multiplexing**, **RMAN**

backup, whole database

See **whole database backup**

backup and recovery

The set of concepts, procedures, and strategies involved in protecting the database against data loss due to media failure or users errors. In a wider sense, backup and recovery also involves maintenance of backups and their associated metadata.

backup control file

A backup of the control file. You can back up the control file with the RMAN `BACKUP` command or with the SQL statement `ALTER DATABASE BACKUP CONTROLFILE TO '`*`filename`*`'`

backup mode

The database mode (also called **hot backup mode**) initiated when you issue the `ALTER TABLESPACE ... BEGIN BACKUP` or `ALTER DATABASE BEGIN BACKUP` command before taking an online backup. You take a tablespace out of backup mode when you issue the `ALTER TABLESPACE ... END BACKUP` or `ALTER DATABASE END BACKUP` command.

You must use this command when you make a user-managed backup of datafiles in an online tablespace. RMAN does not require you to put the database in backup mode. Updates to tablespaces in backup mode create more than the usual amount of redo because each change causes Oracle to write the entire block rather than just the changed data to the redo log.

See Also: corrupt block, online backup

backup piece

A backup piece is a physical file in an RMAN-specific format that belongs to only one backup set. A backup set usually contains only one backup piece. The only time RMAN creates more than one backup piece in a backup set is when you limit the backup piece size using the `MAXPIECESIZE` option of the `ALLOCATE` or `CONFIGURE` command.

See Also: backup, backup set, RMAN

backup retention policy

See **retention policy**

backup set

A backup of one or more datafiles, control files, archived logs, or backup sets produced by the RMAN `BACKUP` command. A backup set is a logical grouping of one or more binary files called **backup pieces**. Backup sets are in a proprietary format and can only be restored by RMAN.

See Also: backup piece, unused block compression, multiplexing, RMAN

block change tracking

An RMAN option that improves incremental backup performance. RMAN maintains a block change tracking file in the flash recovery area. This file logs a record of changed blocks, which RMAN can then read during an incremental backup so that it does not have to perform a full scan of the datafiles.

block media recovery

The recovery of specified blocks within a datafile with the Recovery Manager `BLOCKRECOVER` command. Block media recovery leaves the affected datafiles online and restores and recovers only the damaged or corrupted blocks.

channel

A connection between RMAN and the target database. Each allocated channel starts a new Oracle server session; the session then performs backup, restore, and recovery operations. The type of channel (`DISK` or `sbt`) determines whether the Oracle server process will attempt to read or write and whether it will work through a third-party media manager.

See Also: media manager, target database

checkpoint

A data structure that defines an SCN in the redo thread of a database. Checkpoints are recorded in the control file and each datafile header, and are a crucial element of recovery.

circular reuse records

Control file records containing non-critical information used by RMAN for backups and recovery operations. These records are arranged in a logical ring. When all available record slots are full, Oracle either expands the control file to make room for a new records or overwrites the oldest record. The `CONTROL_FILE_RECORD_KEEP_TIME` initialization parameter controls how many days records must be kept before it can be overwritten. The default for `CONTROL_FILE_RECORD_KEEP_TIME` is 7 days.

See Also: noncircular reuse records

closed backup

A backup of one or more database files taken while the database is closed. Typically, closed backups are whole database backups. If you closed the database cleanly, then all the files in the backup are consistent. Otherwise, the backups are inconsistent.

See Also: consistent shutdown, consistent backup

cold backup

See **closed backup**

complete recovery

Recovery of one or more datafiles that applies all online and archived redo generated after the restored backup. Typically, you perform complete recovery when media failure damages one or more datafiles or control files. You fully recover the damaged files using all redo generated since the restored backup was taken.

See Also: **incomplete recovery**, **media recovery**

unused block compression

How RMAN conserves space by writing only used data blocks into RMAN backup sets. A newly created datafile contains many never-used blocks. When RMAN creates backup sets, it only includes blocks that have been used.

consistent backup

A **whole database backup** that you can open with the RESETLOGS option without performing media recovery. In other words, you do not need to apply redo to this backup for it to be consistent. You can only take consistent backups after you have made a **consistent shutdown** of the database. The database must not be opened until the backup has completed.

See Also: **fuzzy file**, **inconsistent backup**

consistent shutdown

A database shut down with the IMMEDIATE, TRANSACTIONAL, or NORMAL options of the SHUTDOWN statement. A database shut down cleanly does not require recovery; it is already in a consistent state.

control file autobackup

The automatic backup of the current control file that RMAN makes in the situations:

- After every BACKUP command run at the RMAN prompt
- After a BACKUP command within a RUN block that is not followed by another BACKUP command

The control file autobackup has a default filename that allows RMAN to restore it even if the control file and recovery catalog are lost. You can override the default filename if desired.

copy

To back up a bit-for-bit image of an Oracle file (Oracle datafiles, control files, and archived redo logs) onto disk. You can copy in two ways:

- Using operating system utilities (for example, the UNIX `cp` or `dd`)
- Using the RMAN `BACKUP AS COPY` command

See Also: **backup**

corrupt block

An Oracle block that is not in a recognized Oracle format, or whose contents are not internally consistent. Typically, corruptions are caused by faulty hardware or operating system problems. Oracle identifies corrupt blocks as either logically corrupt (an Oracle internal error) or media corrupt (the block format is not correct).

You can only repair a media corrupt block by recovering the block and or dropping the database object that contains the corrupt block so that its blocks are reused for another object. If media corruption is due to faulty hardware, neither solution will work until the hardware fault is corrected.

See Also: **block media recovery**

crash recovery

The automatic application of online redo records to a database after either a single-instance database crashes or all instances of an Oracle Real Applications Cluster configuration crash. Crash recovery only requires redo from the online logs: archived redo logs are not required.

See Also: **recover**

crosscheck

A check to determine whether files on disk or in the media management catalog correspond to the data in the repository and the control file. Because the **media manager** can mark tapes as expired or unusable, and because files can be deleted from disk or otherwise become corrupted, the RMAN repository can contain outdated information about backups. Run the `CROSSCHECK` command to perform a crosscheck. To determine whether you can restore a file, run `VALIDATE BACKUPSET` or `RESTORE ... VALIDATE`.

See Also: **validation**

cumulative incremental backup

An **incremental backup** that backs up all the blocks changed since the most recent backup at level 0. When recovering with cumulative incremental backups, only the most recent cumulative incremental backup needs to be applied.

See Also: **differential incremental backup**, **incremental backup**

current online redo log

The **online redo log** file in which the LGWR background process is currently logging redo records. Those files to which LGWR is not writing are called inactive.

See Also: **redo log**, **redo log groups**

database checkpoint

The thread checkpoint that has the lowest SCN. The database checkpoint guarantees that all changes in all enabled threads prior to the database checkpoint have been written to disk.

See Also: **checkpoint**

database identifier

See DBID

database point-in-time recovery (DBPITR)

The recovery of a database to a specified noncurrent time, SCN, or log sequence number.

See Also: **incomplete recovery**, **tablespace point-in-time recovery (TSPITR)**

datafile media recovery

The application of redo records to a restored datafile in order to roll it forward to a more current time. Unless you are doing **block media recovery**, the datafile must be offline while being recovery.

DBID

An internal, uniquely generated number that differentiates databases. Oracle creates this number automatically when you create the database.

differential incremental backup

A type of **incremental backup** that backs up all blocks that have changed since the most recent backup at level 1 or level 0. For example, in a differential level 1 backup RMAN determines which level 1 or level 0 backup is most recent and then backs up all blocks changed since that backup. Differential backups are the default type of incremental backup. When recovering using differential incremental backups, RMAN must apply all differential incremental level 1 backups since the restored datafile backup.

See Also: cumulative incremental backup, incremental backup

disk controller

A hardware component that is responsible for controlling one or more disk drives.

disk quota

A user-specified limit to the size of the **flash recovery area**. When the disk quota is reached, Oracle automatically deletes files that are no longer needed.

duplicate database

A database created from target database backups using the RMAN duplicate command.

See Also: auxiliary database

export

The extraction of logical data (that is, not physical files) from a database into a binary file using the Oracle export utility. You can then use the Oracle import utility to import the data into a database.

See Also: logical backups

flash recovery area

An optional disk location that you can use to store recovery-related files such as control file and online redo log copies, archived logs, flashback logs, and RMAN backups. Oracle and RMAN manage the files in the flash recovery area automatically. You can specify the **disk quota**, which is the maximum size of the flash recovery area.

flashback logs

Oracle-generated logs, similar to archived redo logs, used by the `FLASHBACK` command. Oracle can only write flashback logs to the flash recovery area. They cannot be backed up to disk.

full backup

A non-incremental RMAN backup. Note that "full" does not refer to how much of the database is backed up, but to the fact that the backup is not incremental. Consequently, you can make a full backup of one datafile.

full resynchronization

An RMAN operation that updates the **recovery catalog** with all changed metadata in the database's control file. You can initiate a full catalog **resynchronization** by issuing the RMAN command `RESYNC CATALOG`. RMAN resynchronizes as needed when executing certain commands.

fuzzy file

A datafile that contains at least one block with an SCN more recent than the checkpoint SCN in its header. For example, this situation occurs when Oracle updates a datafile that is in **backup mode**. A fuzzy file that is restored always requires recovery.

hot backup

See **online backup**

hot backup mode

See **backup mode**

image copy

A bit-for-bit **copy** of a single datafile, archived redo log file, or control file that is:

- Usable as-is to perform recovery (unlike a backup set, which uses **unused block compression** and is in an RMAN-specific format)
- Generated with the RMAN `BACKUP AS COPY` command, an operating system command such as the UNIX `cp`, or by the Oracle archiver process

inactive redo log

A redo log file that is not required for crash or instance recovery because the changes contained in its redo records have already been applied to the database. The **current online redo log** is never inactive. If you operate the database in `ARCHIVELOG` mode, the archiver process archives inactive redo log files.

See Also: online redo log, **redo log**, **redo log groups**

incarnation

A separate version of a physical database. The incarnation of the database changes when you open it with the RESETLOGS option, but you can recover backups from a prior incarnation so long as the necessary redo is available.

incomplete recovery

The recovery of a database in which you do not apply all of the changes generated since you created the restored backup. Is it the same as **database point-in-time recovery (DBPITR)**.

See Also: **complete recovery**, **media recovery**, **recover**

inconsistent backup

A backup in which some of the files in the backup contain changes that were made after the files were checkpointed. This type of backup needs recovery before it can be made consistent. Inconsistent backups are usually created by taking online database backups. You can also make an inconsistent backup by backing up datafiles while a database is closed, either:

- Immediately after the crash of an Oracle instance (or, in a RAC configuration, all instances)
- After shutting down the database using SHUTDOWN ABORT

Inconsistent backups are only useful if the database is in ARCHIVELOG mode.

See Also: **consistent backup**, **online backup**, **system change number (SCN)**, **whole database backup**

incremental backup

An RMAN backup in which only modified blocks are backed up. Incremental backups are classified by **level**. An incremental level 0 backup performs the same function as a full backup in that they both back up all blocks that have ever been used. The difference is that a full backup will not affect blocks backed up by subsequent incremental backups, whereas an incremental backup will affect blocks backed up by subsequent incremental backups.

Incremental backups at level 1 back up only blocks that have changed since previous incremental backups. Blocks that have not changed are not backed up. An incremental backup can be either a **differential incremental backup** or a **cumulative incremental backup**. A cumulative incremental backup backs up all blocks changed since the last level 1 incremental backup. A differential incremental backup backs up all blocks changed since the last level 0 or level 1 incremental backup.

instance failure

The termination of an Oracle instance due to a hardware failure, application error, or `SHUTDOWN ABORT` statement. Strictly speaking, an instance failure occurs whenever the database is not shut down consistently. Crash or instance recovery is always required after an instance failure.

instance recovery

In a RAC configuration, the application of redo data to an open database by an instance when this instance discovers that another instance has crashed.

See Also: recover

LogMiner

A utility that enables log files to be read, analyzed, and interpreted by means of SQL statements

See Also: archived redo log

log sequence number

A number that uniquely identifies a set of redo records in a redo log file. When Oracle fills one online redo log file and switches to a different one, Oracle automatically assigns the new file a log sequence number.

See Also: log switch, redo log

log switch

The point at which LGWR stops writing to the active redo log file and switches to the next available redo log file. LGWR switches when either the active log file is filled with redo records or you force a switch manually.

See Also: redo log

logical backups

Backups in which an Oracle export utility extracts database data and then saves it to a binary file at the operating system level. You can then import the data back into a database with the corresponding Oracle import utility.

long-term backup

A backup that you want to exclude from a backup retention policy, but want to record in the recovery catalog. Typically, long-term backups are snapshots of the database that you may want to use in the future for report generation.

Mean Time To Recover (MTTR)

The desired time required to perform instance or media recovery on the database. A variety of factors influence MTTR for media recovery, including the speed of detection, the method used to perform media recovery, and the size of the database.

media failure

A physical problem that arises when Oracle fails in its attempt to write or read a file that is required to operate the database. Disk failure can affect a variety of files, including the datafiles, redo log files, and control files. Because the database instance cannot continue to function properly, it cannot write to the datafiles.

See Also: media recovery

media manager

A utility provided by a third party vendor that is capable of actions such as loading, labelling and unloading sequential media such as tape drives. Media managers also allow you to configure media expiration and recycling, and may also have the ability to control an ATL (automated tape library).

media recovery

The application of redo or incremental backups to a restored backup datafile or individual data block to bring it to a specified time. Datafile media recovery always begins at the lowest SCN recorded in the datafile headers.

When performing media recovery, you can recover a database, tablespace, datafile, or set of blocks within a datafile. In `ARCHIVELOG` mode, you have the choice of **complete recovery** or **incomplete recovery**. In `NOARCHIVELOG` mode, the only option is typically to restore from the most recent backup without applying redo.

See Also: block media recovery, recover

mirroring

Maintaining identical copies of data on one or more disks. Typically, mirroring is performed on duplicate hard disks at the operating system level, so that if one of the disks becomes unavailable, the other disk can continue to service requests without interruptions. When mirroring files, Oracle writes once while the operating system writes to multiple disks; when **multiplexing** files, Oracle writes the same data to multiple files.

multiplexing

- **online redo logs**

  The automated maintenance of more than one identical copy of the online redo log.

- **control file**

  The automated maintenance of more than one identical copy of a database's control file.

- **backup set**

  The RMAN technique of reading database files *simultaneously* from the disks and then writing the blocks to the *same* backup piece.

- **archived redo logs**

  The Oracle archiver process is able to archive multiple copies of a redo log.

See Also: **mirroring**

NOARCHIVELOG mode

The mode of the database in which Oracle does not require filled online redo logs to be archived before they can be overwritten. Specify the mode at database creation or change it with the `ALTER DATABASE NOARCHIVELOG` command. Note that running in `NOARCHIVELOG` mode severely limits the possibilities for recovery of lost data.

See Also: **archived redo log**, **ARCHIVELOG mode**

noncircular reuse records

Control file records containing critical information needed by the Oracle database. These records do not change often and cannot be overwritten. Some examples of information in non-circular reuse records include datafiles and online redo logs.

See Also: **circular reuse records**

offline backup

A backup of a tablespace or datafile made when the tablespace or datafile is offline and the database open. Run the `ALTER TABLESPACE OFFLINE` statement to take a tablespace offline, and the `ALTER DATABASE DATAFILE ... OFFLINE` statement to take an individual datafile offline.

offline normal

When a tablespace is taken offline normal, it is taken offline using the `ALTER TABLESPACE ... OFFLINE NORMAL` statement. The datafiles in the tablespace are checkpointed and do not require recovery before being brought online. If a tablespace is not taken offline normal, then its datafiles must be recovered before being brought online.

online backup

A backup of one or more datafiles taken while a database is open and the datafiles are online. When you make a user-managed backup while the database is open, you must put the tablespaces in **backup mode** by issuing an `ALTER TABLESPACE BEGIN BACKUP` command. (`ALTER DATABASE BEGIN BACKUP` is an alternative when your database has many tablespaces, all of which need to be put in backup mode.) When you make an RMAN backup while the database is open, however, you should not put the tablespaces in backup mode.

online redo log

The online redo log is a set of two or more files that record all changes made to Oracle datafiles and control files. Whenever a change is made to the database, Oracle generates a redo record in the redo buffer. The LGWR process flushes the contents of the redo buffer into the online redo log.

The **current online redo log** is the one being written to by LGWR. When LGWR gets to the end of the file, it performs a **log switch** and begins writing to a new log file. If you run the database in `ARCHIVELOG` mode, then the archiver process or processes copy the redo data into an **archived redo log**.

See Also: **archived redo log**

online redo log group

The Oracle online redo log consists of two or more online redo log groups. Each group contains one or more identical online redo log members. An **online redo log member** is a physical file on the operating system containing the redo records.

online redo log member

A physical online redo log file within an **online redo log group**. Each log group must have one or more members. Each member of a group is identical.

operating system backups

See **user-managed backups**

operating system backup and recovery

See **user-managed backup and recovery**

Oracle Flashback Database

The return of the whole database to a prior consistent SCN by means of the RMAN `FLASHBACK` command or SQL*Plus `FLASHBACK` statement. A database flashback is different from traditional media recovery because it does not involve the restore of physical files, instead restoring your current datafiles to past states using saved images of changed data blocks. The flashback database process uses **flashback logs** and archived redo logs.

Oracle-managed file

A database file managed by the Oracle Managed Files feature.

Oracle Managed Files (OMF)

A feature of the Oracle database which manages the creation, naming and deletion of Oracle database files within dedicated areas of disk, to minimize the need for DBAs to concern themselves with such specifics.

orphaned backups

Backups that are unusable because they belong to incarnations of the database that are not direct ancestors of the current incarnation.

parallel recovery

A form of recovery in which several processes simultaneously apply changes from redo log files. Instance and media recovery can be parallelized automatically with the `RECOVERY_PARALLELISM` initialization parameter or options to the SQL*Plus `RECOVER` command.

parallelization

Allocating multiple channels for RMAN backup and recovery operations.

partial resynchronization

A type of **resynchronization** in which RMAN transfers data about archived logs, backup sets, and datafile copies from the target control file to the **recovery catalog**.

password files

A file created by the `ORAPWD` command. A database must use password files if you wish to connect using the `SYSDBA` or `SYSOPER` roles over a network. For a more comprehensive explanation, see the *Oracle Database Administrator's Guide*.

physical schema

The datafiles, control files, and redo logs in a database at a given time. Issue the RMAN `REPORT SCHEMA` command to obtain a list of tablespaces and datafiles.

pluggable tablespace

See **transportable tablespace**

proxy copy

A backup in which the **media manager** takes over the transfer of data between the media storage device and disk during RMAN backup and restore operations.

recover

To recover a database file or a database is typically to perform media recovery, crash recovery or instance recovery. Can also be used generically, as in "recover your data," to refer to reconstructing or re-creating lost data by any means.

See Also: **recover**

recovery

When used to refer to a database file or a database, the application of redo data or incremental backups to database files in order to reconstruct lost changes. The three types of recovery are **instance recovery**, **crash recovery**, and **media recovery**. Oracle performs the first two types of recovery automatically using online redo records; only media recovery requires you to restore a backup and issue commands.

See Also: **complete recovery**, **incomplete recovery**

recovery catalog

A set of Oracle tables and views used by RMAN to store information about Oracle databases. RMAN uses this data to manage the backup, restore, and recovery of Oracle databases. The recovery catalog is optional. If you do not use a recovery catalog, RMAN uses the target control file as the sole repository of metadata.

See Also: **recovery catalog database**

recovery catalog database

An Oracle database that contains a recovery catalog schema. You should not store the recovery catalog in the target database.

Recovery Manager (RMAN)

A utility that backs up, restores, and recovers Oracle databases. You can use it with or without the central information repository called a **recovery catalog**. If you do not use a recovery catalog, RMAN uses the database's control file to store information necessary for backup and recovery operations. You can use RMAN in conjunction with a media manager to back up files to tertiary storage.

See Also: **backup piece**, **backup set**, **copy**, **media manager**, **recovery catalog**

recovery set

One or more tablespaces that are being recovered to an earlier point in time during **tablespace point-in-time recovery (TSPITR)**. After TSPITR, all database objects in the recovery set have been recovered to the same point in time.

See Also: **auxiliary set**

recovery window

A recovery window is a period of time in a retention policy bounded by the current time and the earliest point of recoverability. The point of recoverability is the end time for a hypothetical point-in-time recovery, that is, the point to which you must be able to recover following a media failure. A retention policy states that you must have enough backups and archived redo logs to be able to recover to any point between the current time and the point of recoverability.

See Also: **retention policy**

redo log

A redo log can be either an **online redo log** or an **archived redo log**. The online redo log is a set of two or more redo log groups that records all changes made to Oracle datafiles and control files. An archived redo log is a copy of an online redo log that has been written to an offline destination.

See Also: **archived redo log**, **online redo log**

redo log groups

Each online redo log member (which corresponds to an online redo log file) belongs to a group. A group has one or more identical members. A multiplexed redo log is a redo log in which the redo groups have multiple members.

redo thread

The redo generated by an instance. If the database runs in a single instance configuration, then the database has only one thread of redo.

redundancy set

A set of backups enabling you to recover from the failure or loss of any Oracle database file.

registration

In RMAN, the execution of a `REGISTER DATABASE` command in order to record the existence of a target database in the recovery catalog. A target database is uniquely identified in the catalog by its DBID. You can register more than one database in the same catalog, and also register the same database in multiple catalogs.

See Also: **DBID**

repository

The RMAN metadata about backup and recovery operations on the target database. Either the control file or the recovery catalog can function as the RMAN repository.

See Also: **recovery catalog**

RESETLOGS option

A method for opening a database that archives any current online redo logs (if using ARCHIVELOG mode), resets the log sequence number to 1, clears the online redo logs, and begins a new database incarnation. A database must be opened with the RESETLOGS keyword after incomplete recovery or recovery with a backup control file.

restore

The replacement of a lost or damaged file with a backup. You can restore files either with commands such as UNIX `cp` or the RMAN `RESTORE` command.

resynchronization

The operation that updates the recovery catalog with current information from the target database control file. You can initiate a **full resynchronization** of the catalog by issuing a `RESYNC CATALOG` command. A **partial resynchronization** transfers information to the recovery catalog about archived redo logs, backup sets and datafile copies. RMAN performs resynchronizations automatically when needed.

retention policy

A user-defined policy for determining how long backups and archived logs need to be retained for media recovery. You can define a retention policy in terms of backup redundancy or a **recovery window**. For example, if you need to recover the database to any point within the last 7 days, then at least one backup made 8 days ago or earlier must be retained. Also, you must retain all of the archived redo logs needed to perform the recovery.

RMAN

See **Recovery Manager (RMAN)**

rollback segments

Database segments that record the before-images of changes to the database.

rolling back

The use of rollback segments to undo uncommitted transactions applied to the database during the **rolling forward** stage of **recover**.

rolling forward

The application of redo records or incremental backups to datafiles and control files in order to recover changes to those files.

See Also: recover, **rolling back**

SBT

System Backup to Tape

snapshot control file

A copy of a database's control file taken in an operating system specific location by Recovery Manager. RMAN uses the snapshot control file to read a consistent version of a control file when either resynchronizing the recovery catalog or backing up the control file.

standby database

A copy of a production database that you can use for disaster protection.

stored script

A sequence of RMAN commands stored in the **recovery catalog**.

system change number (SCN)

A stamp that defines a committed version of a database at a point in time. Oracle assigns every committed transaction a unique SCN.

SYSTEM tablespace

The `SYSTEM` tablespace differs from other tablespaces in that all datafiles contained in the tablespace must be online for Oracle to function. If a media failure affects one of the datafiles in `SYSTEM`, then you must mount the database and recover.

tablespace point-in-time recovery (TSPITR)

The recovery of one or more non-`SYSTEM` tablespaces to a noncurrent time. You can use either RMAN or user-managed methods to perform TSPITR.

target database

In RMAN, the database that you are backing up or restoring.

tempfile

A file that belongs to a temporary tablespace, and is created with the `TEMPFILE` option. Temporary tablespaces cannot contain permanent database objects such as tables, and are typically used for sorting. Because tempfiles cannot contain permanent objects, RMAN does not back them up.

time-based recovery

The incomplete recovery of database files to a noncurrent time. Time-based recovery is also known as **point-in-time recovery**.

See Also: incomplete recovery, **media recovery**, **recover**

transportable tablespace

A feature that transports a set of tablespaces from one database to another, or from one database to itself. Transporting or "plugging" a tablespace into a database is like creating a tablespace with preloaded data.

undo tablespace

A dedicated tablespaces that stores only undo information when the database is run in **automatic undo management mode**.

user-managed backups

Backups made using a non-RMAN method, for example, using an operating system utility. For example, you can make a user-managed backup by running the `cp` command on UNIX or the `copy` command on Windows. A user-managed backup is also called an **operating system backups**.

user-managed backup and recovery

A backup and recovery strategy for an Oracle database that does not use RMAN. This term is equivalent to **operating system backup and recovery**. You can back up and restore database files using operating system utilities (for example, the `cp` command in UNIX), and recover using the SQL*Plus `RECOVER` command.

validation

A test that checks whether a backup can be restored. RMAN scans the backups and looks at the checksum to verify that the contents can be successfully restored.

See Also: **crosscheck**, **media manager**, **recovery catalog**

whole database backup

A backup of the control file and all datafiles that belong to a database.

See Also: **backup**

Index