Oracle9*i*

Backup and Recovery Concepts

Release 1 (9.0.1)

July 2001

Part No. A90133-02

Oracle9*i* Backup and Recovery Concepts, Release 1 (9.0.1)

Part No. A90133-02

Primary Author: Lance Ashdown

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

Contents

Send Us Your Comments

Oracle9*i* Backup and Recovery Concepts, Release 1 (9.0.1)

Part No. A90133-02

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 gives a basic conceptual overview of Oracle 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 *Oracle9i Database Concepts* and *Oracle9i Database Administrator's Guide*
- The operating system environment under which you are running Oracle

Organization

This document contains:

Chapter 1, "Backup and Recovery Overview"
This chapter gives a brief overview of Oracle backup and recovery.

Chapter 2, "Backup Principles"
This chapter describes the basic principles of RMAN and operating system backups.

Chapter 3, "Recovery Principles"
This chapter describes the basic principles of crash, instance, and media recovery.

Chapter 4, "Backup and Recovery Strategies"
This chapter gives general recommendations for a backup and recovery strategy.

"Glossary"
This chapter gives definitions for common backup and recovery terms.

Related Documentation

For more information, see these Oracle resources:

- *Oracle9i Recovery Manager User's Guide and Reference*
- *Oracle9i User-Managed Backup and Recovery Guide*
- *Oracle9i Database Utilities*
- `http://www.oracle.com/database/recovery`

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

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

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

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

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

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

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

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

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

Conventions

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

- Conventions in Text
- Conventions in Code Examples

Conventions in Text

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

Convention	Meaning	Example
Bold	Bold typeface indicates terms that are defined in the text or terms that appear in a glossary, or both.	The C datatypes such as **ub4**, **sword**, or **OCINumber** are valid. When you specify this clause, you create an **index-organized table**.

Convention	Meaning	Example
Italics	Italic typeface indicates book titles, emphasis, syntax clauses, or placeholders.	*Oracle9i Database Concepts* You can specify the *parallel_clause*. Run `U`*`old_release`*`.SQL` where *old_release* refers to the release you installed prior to upgrading.
UPPERCASE monospace (fixed-width font)	Uppercase monospace typeface indicates elements supplied by the system. Such elements include parameters, privileges, datatypes, RMAN keywords, SQL keywords, SQL*Plus or utility commands, packages and methods, as well as system-supplied column names, database objects and structures, user names, and roles.	You can specify this clause only for a `NUMBER` column. You can back up the database using the `BACKUP` command. Query the `TABLE_NAME` column in the `USER_TABLES` data dictionary view. Specify the `ROLLBACK_SEGMENTS` parameter. Use the `DBMS_STATS.GENERATE_STATS` procedure.
lowercase monospace (fixed-width font)	Lowercase monospace typeface indicates executables and sample user-supplied elements. Such elements include computer and database names, net service names, and connect identifiers, as well as user-supplied database objects and structures, column names, packages and classes, user names and roles, program units, and parameter values.	Enter `sqlplus` to open SQL*Plus. 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.

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 ])`

Convention	Meaning	Example
{ }	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	`CREATE TABLE ... AS subquery;`
	■ That you can repeat a portion of the code	`SELECT col1, col2, ... , col`*`n`*` FROM employees;`
. . .	Vertical ellipsis points indicate that we have omitted several lines of code not directly related to the example.	

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

`Italics`	Italicized text indicates variables for which you must supply particular values.	`CONNECT SYSTEM/`*`system_password`*

`UPPERCASE`	Uppercase typeface indicates elements supplied by the system. We show these terms in uppercase in order to distinguish them from terms you define. Unless terms appear in brackets, enter them in the order and with the spelling shown. However, because these terms are not case sensitive, you can enter them in lowercase.	`SELECT last_name, employee_id FROM employees;`
		`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.	`SELECT last_name, employee_id FROM employees;`
		`sqlplus hr/hr`

Documentation Accessibility

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

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

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

1 Backup and Recovery Overview

This chapter introduces concepts that are fundamental to backup and recovery. It is intended as a general overview. Subsequent chapters explore backup and recovery concepts in greater detail.

This chapter includes the following topics:

- Backup and Recovery: Basic Concepts
- Errors and Failures Requiring Recovery
- Data Structures Used for Database Recovery
- Database Archiving Modes
- Oracle's Backup and Recovery Solutions

Backup and Recovery: Basic Concepts

In general, **backup and recovery** refers to the various strategies and procedures involved in protecting your database against data loss and reconstructing the data should that loss occur. The reconstructing of data is achieved through **media recovery**, which refers to the various operations involved in restoring, rolling forward, and rolling back a **backup** of database files.

This section contains these topics:

- Oracle Backups: Basic Concepts
- Oracle Recovery: Basic Concepts

Oracle Backups: Basic Concepts

A **backup** is a copy of data. This copy can include important parts of the database such as the control file and datafiles. A backup is a safeguard against unexpected data loss and application errors. If you lose the original data, then you can reconstruct it by using a backup.

Backups are divided into **physical backups** and **logical backups**. Physical backups, which are the primary concern in a backup and recovery strategy, are copies of physical database files. You can make physical backups with either the Recovery Manager (RMAN) utility or operating system utilities. In contrast, logical backups contain logical data (for example, tables and stored procedures) extracted with the Oracle Export utility and stored in a binary file. You can use logical backups to supplement physical backups.

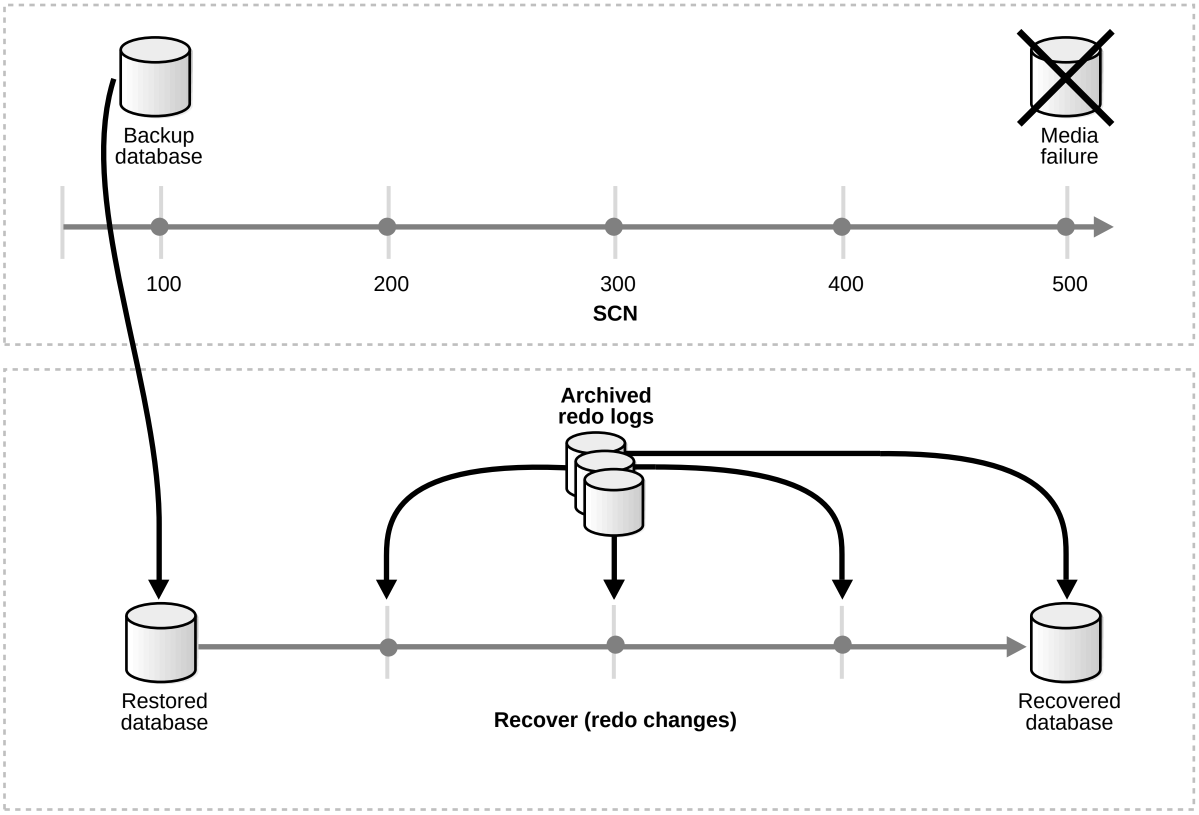
Oracle Recovery: Basic Concepts

To **restore** a physical backup of a datafile or control file is to reconstruct it and make it available to the Oracle database server. To **recover** a restored datafile is to update it by applying **archived redo logs** and **online redo logs**, that is, records of changes made to the database after the backup was taken. If you use RMAN, then you can also recover restored datafiles with **incremental backups**, which are backups of a datafile that contain only blocks that changed after a previous incremental backup.

After the necessary files are restored, **media recovery** must be initiated by the user. Media recovery can use both archived redo logs and online redo logs to recover the datafiles. If you use SQL*Plus, then you can run the `RECOVER` command to perform recovery. If you use RMAN, then you run the RMAN `RECOVER` command to perform recovery.

Figure 1–1 illustrates the basic principle of backing up, restoring, and performing media recovery on a database.

Figure 1–1 Restoring and Recovering a Database



Unlike media recovery, Oracle performs **crash recovery** and **instance recovery** automatically after an instance failure. Crash and instance recovery recover a database to its transaction-consistent state just before instance failure. By definition, **crash recovery** is the recovery of a database in a single-instance configuration or an Oracle Real Application Clusters configuration in which all instances have crashed. In contrast, **instance recovery** is the recovery of one failed instance by a live instance in an Oracle Real Application Clusters configuration.

Crash and instance recovery involve two distinct operations: **rolling forward** the current, online datafiles by applying both committed and uncommitted transactions contained in online redo records, and then **rolling back** changes made in uncommitted transactions to their original state. Because crash and instance recovery are automatic, this manual will not discuss these operations.

See Also:

- *Oracle9i Recovery Manager User's Guide and Reference*
- *Oracle9i User-Managed Backup and Recovery Guide* to learn how to make operating system backups and perform recovery using SQL*Plus.
- *Oracle9i Database Performance Guide and Reference* to learn about tuning crash and instance recovery

Errors and Failures Requiring Recovery

Several problems can halt the normal operation of an Oracle database or affect database I/O operations. The following sections describe the most common types of problems. For some of these problems, crash and instance recovery occur automatically and require no action on the part of the database administrator. For other problems, administrator-initiated media recovery is required.

This section contains these topics:

- Media Failure
- User Error
- Database Instance Failure
- Statement Failure
- Process Failure
- Network Failure

Media Failure

An error can occur when trying to write or read an file on disk that is required to operate an Oracle database. This occurrence is called **media failure** because there is a physical problem reading or writing to files on the storage medium.

A common example of media failure is a disk head crash that causes the loss of all database files on a disk drive. All files associated with a database are vulnerable to a disk crash, including datafiles, control files, online redo logs, and archived logs.

The appropriate recovery from a media failure depends on the files affected. Media failure is the primary concern of a backup and recovery strategy, because it typically requires restoring some or all database files and the application of redo during recovery.

See Also:

- *Oracle9i Recovery Manager User's Guide and Reference* for a discussion of backup and recovery methods using RMAN
- *Oracle9i User-Managed Backup and Recovery Guide* for a discussion of backup and recovery methods using operating system utilities

How Media Failures Affect Database Operation

Media failures can affect one or all types of files necessary for the operation of an Oracle database, including datafiles, online redo log files, and control files. Also, media failures can affect archived redo logs stored on disk.

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. Typically, multiplexed files are stored on separate disks.

If a media failure damages a single disk, and if you have 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 once Oracle 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, Oracle discovers it cannot read a datafile and an operating system error is returned to the application, along with an Oracle error indicating that the file cannot be found, cannot be opened, or cannot be read. Oracle continues to run, but the error is returned each time an unsuccessful read occurs. At the next checkpoint, a write error will occur when Oracle attempts to write the file header as part of the standard checkpoint process.

If Oracle discovers that it cannot write to a datafile, and if Oracle is in ARCHIVELOG mode, then Oracle returns an error in the database writer trace file and takes the datafile offline automatically. Only the datafile that cannot be written to is taken offline; the tablespace containing that file remains online.

If the datafile that cannot be written to is in the **SYSTEM tablespace**, then the file is not taken offline. Instead, an error is returned and Oracle shuts down the instance. The reason for this exception is that all files in the SYSTEM tablespace must be online in order for Oracle to operate properly. For the same reason, the undo tablespaces (if in **automatic undo management mode**) or the datafiles of a tablespace containing active rollback segments (if in **manual undo management mode**) must remain online.

If Oracle cannot write to a datafile, and Oracle is not archiving the filled online redo log files, then the database writer background process fails and the current instance fails. If the problem is temporary (for example, the disk controller was powered off), then crash or instance recovery usually can be performed using the online redo log files, in which case the instance can be restarted. However, if a datafile is permanently damaged and archiving is not used, then you must restore the entire database using the most recent **consistent backup**.

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

As an administrator, you can do little to prevent user errors such as accidentally dropping a table. Often, user error can be reduced by increased training on database and application principles. You can also avoid user errors by administering privileges correctly so that users are able to do less potential damage. Furthermore, by planning an effective recovery scheme ahead of time, you can ease the work necessary to recover from user errors.

Typically, a user error such as a dropped table requires either re-entering the lost changes manually (if a record of them exists), importing the dropped object (if an export file exists), or performing incomplete recovery either of an individual tablespaces (called **tablespace point-in-time recovery (TSPITR)**) or of the entire database.

Database Instance Failure

Database **instance failure** occurs when a problem prevents an Oracle database instance from continuing to run. An instance failure can result from a hardware problem, such as a power outage, or a software problem, such as an operating system crash. Instance failure also results when you issue a SHUTDOWN ABORT or STARTUP FORCE statement.

Mechanics of Instance and Crash Recovery

When one or more instances fail, Oracle automatically recovers the lost changes associated with the instance or instances. Crash or instance recovery consists of the following steps:

1. Rolling forward to recover data that has not been recorded in the datafiles, yet has been recorded in the online redo log, including changes to undo blocks. This phase is called **cache recovery**.
2. Opening the database. Instead of waiting for all transactions to be rolled back before making the database available, Oracle allows the database to be opened as soon as cache recovery is complete. Any data that is not locked by unrecovered transactions is immediately available.
3. Marking all transactions systemwide that were active at the time of failure as DEAD and marking the rollback or undo segments containing these transactions as PARTLY AVAILABLE.
4. Rolling back dead transactions as part of SMON recovery. This phase is called **transaction recovery**.
5. Resolving any pending distributed transactions undergoing a two-phase commit at the time of the instance failure.
6. As new transactions encounter rows locked by dead transactions, they can automatically roll back the dead transaction to release the locks. If you are using Fast-Start Recovery, then only the data block is immediately rolled back, as opposed to the entire transaction.

See Also:

- *Oracle9i Real Application Clusters Administration* for a discussion of instance recovery
- *Oracle9i Database Performance Guide and Reference* for a discussion of Fast-Start Recovery and instance recovery tuning

Statement Failure

Statement failure occurs when there is a logical failure in the handling of a statement in an Oracle program. For example, assume that all extents of a table (in other words, the number of extents specified in the `MAXEXTENTS` parameter of the `CREATE TABLE` statement) are allocated, and are completely filled with data. A valid `INSERT` statement cannot insert a row because no space is available. Therefore, the statement fails.

If a statement failure occurs, then the Oracle software or operating system returns an error. A statement failure usually requires no recovery steps: Oracle automatically corrects for statement failure by rolling back any effects of the statement and returning control to the application. The user can simply re-execute the statement after the problem indicated by the error message is corrected. For example, if insufficient extents are allocated, then the DBA needs to allocate more extents so that the user's statement can execute.

Process Failure

A process failure is a failure in a user, server, or background process of a database instance such as an abnormal disconnect or process termination. When a process failure occurs, the failed subordinate process cannot continue work, although the other processes of the database instance can continue.

The Oracle background process PMON detects aborted Oracle processes. If the aborted process is a user or server process, then PMON resolves the failure by rolling back the current transaction of the aborted process and releasing any resources that this process was using. Recovery of the failed user or server process is automatic. If the aborted process is a background process, then the instance usually cannot continue to function correctly. Therefore, you must shut down and restart the instance.

Network Failure

When your system uses networks such as local area networks and phone lines to connect client workstations to database servers, or to connect several database servers to form a distributed database system, network failures such as aborted phone connections or network communication software failures can interrupt the normal operation of a database system. For example:

- A network failure can interrupt normal execution of a client application and cause a process failure to occur. In this case, the Oracle background process `PMON` detects and resolves the aborted server process for the disconnected user process, as described in the previous section.
- A network failure can interrupt the two-phase commit of a distributed transaction. After the network problem is corrected, the Oracle background process `RECO` of each involved database automatically resolves any distributed transactions not yet resolved at all nodes of the distributed database system.

Data Structures Used for Database Recovery

Several structures of an Oracle database safeguard data against possible failures. This section introduces each of these structures and its role in database recovery.

This section contains these topics:

- Redo Logs
- Rollback and Undo Segments
- Control Files

Redo Logs

The **online redo log**, present for every Oracle database, records all changes made in an Oracle database. The online redo log of a database consists of at least two **redo log files** that are separate from the datafiles (which actually store a database's data). As part of recovery from an instance or media failure, Oracle applies the appropriate changes in the database's redo log to the datafiles, which update database data to the instant that the failure occurred.

Online Redo Log Groups and Members

Every database must have at least two **online redo log groups**. Each redo log group contains at least one **online redo log member**, which is a physical file containing the redo records. If you configure a group to contain multiple members, then you are **multiplexing** the online redo logs. The multiplexed members of the group contain identical redo data but use different filenames.

Oracle uses and reuses these files in a circular fashion to record database changes. The log file that Oracle is currently writing to is called the **current online redo log**.

The background process LGWR records all changes made through the associated instance to the current online redo log files. Each redo record contains both the old and the new values. Oracle also records the old value to an undo block located either in a rollback segment (if running in manual undo management mode) or in a dedicated undo tablespace (if running in automatic undo management mode).

Archived Redo Logs

Optionally, you can configure an Oracle database to archive copies of the online redo logs after they fill. This type of log is called an **archived redo log**. An archived log is uniquely identified by its **redo thread** number and **log sequence number**. By **archiving** filled online redo log files, older redo log data is preserved for operations such as media recovery, while the preallocated online redo log files continue to be reused to store the most current database changes.

Datafiles that were restored from backup, or were not closed by a **clean shutdown**, may not be completely up to date. During recovery, datafiles must be updated by applying the changes in the archived and online redo logs.

Rollback and Undo Segments

You can operate the database in either of two mutually exclusive modes: **manual undo management mode**, or **automatic undo management mode**. In the first case, you must create and manage **rollback segments**. In the case of automatic undo management, you create an **undo tablespace** that contains system-managed **undo segments**. Rollback and undo segments are used for a number of functions in the operation of an Oracle database. In general, these segments store the "before image" of data that has been changed by uncommitted transactions.

Among other things, the information in a rollback or undo segment is used during database recovery to undo any uncommitted changes applied from the redo log to the datafiles. Therefore, if database recovery is necessary, then the data is in a consistent state after the rollback segments are used to remove all uncommitted data from the datafiles.

Control Files

In general, the control files of a database store the status of the physical structure of the database. The control file is absolutely crucial to database operation. It contains (but is not limited to) the following types of information:

- Database information (**RESETLOGS SCN and time stamp**)
- Progress of the **thread checkpoint**
- Tablespace and datafile records (filenames, datafile checkpoints, read / write status, offline ranges)
- Redo threads (current online redo log)
- Log records (sequence numbers, SCN range in each log)
- RMAN backup and copy records
- Block corruption information

Status information in the control file such as the database checkpoints, current online redo log file, and the **datafile header** checkpoints for the datafiles guides Oracle during crash, instance, or media recovery.

Database Archiving Modes

A database can operate in two distinct modes: **NOARCHIVELOG mode** (media recovery disabled) or **ARCHIVELOG mode** (media recovery enabled). The database mode has a profound impact on your backup and recovery strategy.

This section contains these topics:

- NOARCHIVELOG Mode
- ARCHIVELOG Mode

NOARCHIVELOG Mode

If a database is used in `NOARCHIVELOG` mode, then the archiving of the online redo log is disabled. Information in the control file indicates that archiving is not required for filled groups. Therefore, as soon as a filled group becomes inactive, the group is available for reuse by the LGWR process.

`NOARCHIVELOG` mode protects a database only from instance failure, not from media failure. Only the most recent changes made to the database, stored in the groups of the online redo log, are available for crash or instance recovery. These changes are sufficient for crash or instance recovery because Oracle will not overwrite an online log that may be needed until its changes have been recorded in the datafiles. However, it will not be possible to perform media recovery by applying archived redo logs.

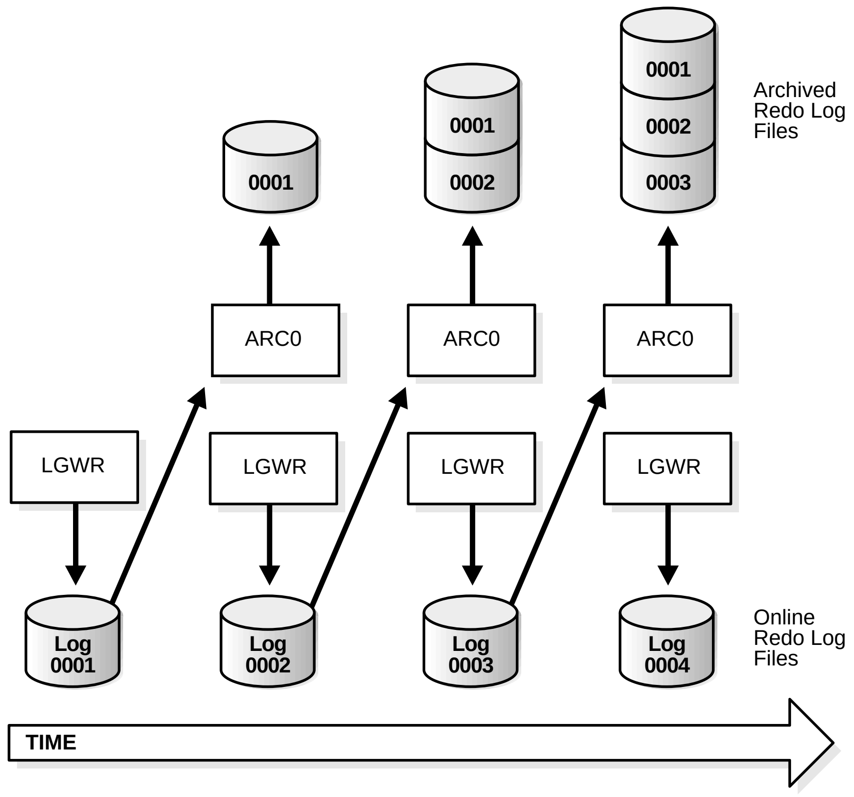
ARCHIVELOG Mode

If an Oracle database operates in `ARCHIVELOG` mode, then the archiving of the online redo log is enabled. Information in a database control file indicates that a group of filled online redo log files cannot be reused by LGWR until the group has been archived.

Figure 1–2 illustrates how the database's online redo log files are used in `ARCHIVELOG` mode and how the archived redo log is generated by the process archiving the filled groups (for example, `ARC0` in this illustration).

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

Figure 1–2 Online Redo Log File Use in ARCHIVELOG Mode



Automatic Archiving and the Archiver Background Processes

You can configure an instance to have an additional background process, the archiver (ARC*n*), which automatically archives each group of online redo log files after it becomes an **inactive redo log**. Automatic archiving frees you from having to keep track of, and archive, filled groups manually. For this convenience alone, automatic archiving is the choice of most database systems that run in ARCHIVELOG mode. For heavy workloads, such as bulk loading of data, multiple archiver processes can be configured by setting the initialization parameter LOG_ARCHIVE_MAX_PROCESSES.

If you request automatic archiving at instance startup by setting the LOG_ARCHIVE_START initialization parameter, then Oracle starts the number of

ARC*n* processes specified by `LOG_ARCHIVE_MAX_PROCESSES` during instance startup. Otherwise, the ARC*n* processes are not started when the instance starts up.

You can interactively start or stop automatic archiving at any time. If automatic archiving was not specified to start at instance startup, and if you subsequently start automatic archiving, then Oracle creates the ARC*n* background processes. ARC*n* then remains for the duration of the instance, even if automatic archiving is temporarily turned off and on again, although the number of ARC*n* processes can be changed dynamically by setting `LOG_ARCHIVE_MAX_PROCESSES` with the `ALTER SYSTEM` statement.

The archiver always archives groups in order, beginning with the lowest **log sequence number**. The archiver automatically archives filled groups as they become inactive. A record of every automatic archival is written in the ARC*n* trace file by the archiver process. Each entry shows the time the archive started and stopped.

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

Manual Archiving

If a database runs in `ARCHIVELOG` mode, then you can manually archive the filled groups of inactive online redo log files, as necessary, whether or not automatic archiving is enabled or disabled. If automatic archiving is disabled, then you must manually archive filled groups.

For most database systems, automatic archiving is best because you do not have to watch for a group to become inactive and available for archiving. Furthermore, if automatic archiving is disabled and manual archiving is not performed fast enough, then database operation can be suspended temporarily whenever the log writer is forced to wait for an inactive group to become available for reuse.

The manual archiving option is provided so that you can:

- Archive a group when automatic archiving has been stopped because of a problem (for example, the offline storage device specified as the archived redo log destination has experienced a failure or become full)

- Archive a group in a non-standard fashion (for example, archive one group to one offline storage device, the next group to a different offline storage device, and so on)
- Rearchive a group if the original archived version is lost or damaged

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

Oracle's Backup and Recovery Solutions

You have two methods for performing Oracle backup and recovery: **Recovery Manager (RMAN)** and **user-managed backup and recovery**. RMAN is a utility automatically installed with the database that can back up any Oracle8 or later database. RMAN uses server sessions on the database to perform the work of backup and recovery. RMAN has its own syntax and is accessible either through a command-line interface or though the Oracle Enterprise Manager GUI. RMAN comes with an API that allows it to function with a third-party **media manager**.

One of the principal advantages of RMAN is that it obtains and stores metadata about its operations in the control file of the production database. You can also set up an independent **recovery catalog**, which is a schema that contains metadata imported from the control file, in a separate **recovery catalog database**. RMAN performs the necessary record keeping of backups, archived logs, and so forth using the metadata, so restore and recovery is greatly simplified.

An alternative method of performing recovery is to use operating system commands for backups and SQL*Plus for recovery. This method, also called **user-managed backup and recovery**, is fully supported by Oracle Corporation, although use of RMAN is highly recommended because it is more robust and greatly simplifies administration.

Whether you use RMAN or user-managed methods, you can supplement your physical backups with logical backups of schema objects made using the Export utility. The utility writes data from an Oracle database to binary operating system files. You can later use Import to restore this data into a database.

See Also:

- *Oracle9i Recovery Manager User's Guide and Reference* for complete information on using RMAN for backup and recovery
- *Oracle9i User-Managed Backup and Recovery Guide* to learn how to make backups with operating system utilities and perform recovery with SQL*Plus

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 Oracle databases of release 8.0 or higher, then RMAN is an appropriate choice. If you manage some Oracle7 and some post-Oracle7 releases, then you must use a non-RMAN method of backing up the Oracle7 databases.

Table 1–1 describes the version and system requirements for the Oracle 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 `cp`)
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

Feature	Recovery Manager	Operating System	Export
Closed database backups	Supported. Requires instance to be mounted.	Supported.	Not supported.
Open database backups	Do not use BEGIN/END BACKUP statements.	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 writes to V$BACKUP_CORRUPTION and V$COPY_CORRUPTION.	Not supported.	Supported. Identifies corrupt blocks in the export log.
Automatic backup	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 in the recovery catalog and in the control file, or exclusively in the target control file.	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.
Backs up initialization parameter file and password files	Not supported.	Supported.	Not supported.
Operating system independent language	Supported (uses PL/SQL interface).	Not supported.	Supported.

2
Backup Principles

This chapter introduces database concepts that are fundamental to backing up a database.

This chapter includes the following topics:

Physical and Logical Backups

Backups of Oracle data are either physical or logical.

This section contains these topics:

- Physical Backups
- Logical Backups

Physical Backups

In contrast to logical backups, **physical backups** are backups of physical database files: datafiles and control files. If you run the database in `ARCHIVELOG` mode, then the database also generates archived redo logs. You can back up the datafiles, control files, and archived redo logs. Backups of online redo logs are not supported, as explained in "Avoiding the Backup of Online Redo Logs" on page 4-10.

Physical backups are divided into two categories: image copies and backups in a proprietary format. An image copy is an exact duplicate of a datafile, control file, or archived log. You can create image copies of physical files with operating system utilities or the RMAN `COPY` command, and you can restore them as-is without performing additional processing by using either operating system utilities or the RMAN `RESTORE` command.

Note: Unlike operating system copies, the RMAN `COPY` command validates the blocks in the file and records the copy in the repository.

The RMAN `BACKUP` command generates a backup set, which is a logical object containing one or more backup pieces. Each backup piece is a physical file in a proprietary, binary format. You must use RMAN to restore a backup set.

Logical Backups

In contrast to physical backups, **logical backups** are exports of schema objects into a binary file. Import and Export are utilities used to move Oracle data in and out of Oracle schema. Export writes data from an Oracle database to binary operating system files. These export files store information about schema objects, for example, tables and stored procedures. Import is a utility that reads export files and restores the corresponding data into an existing database.

Although Import and Export are designed for moving Oracle data, you can also use them as a supplemental method of protecting data in an Oracle database. You should not use Import and Export as the sole method of backing up your data.

> **See Also:** *Oracle9i Database Utilities* to learn more about logical backups

Whole Database and Partial Database Backups

This section includes these topics:

- Whole Database Backups
- Tablespace Backups
- Datafile Backups
- Control File Backups

> **See Also:** *Oracle9i Database Utilities* for information about logical backups

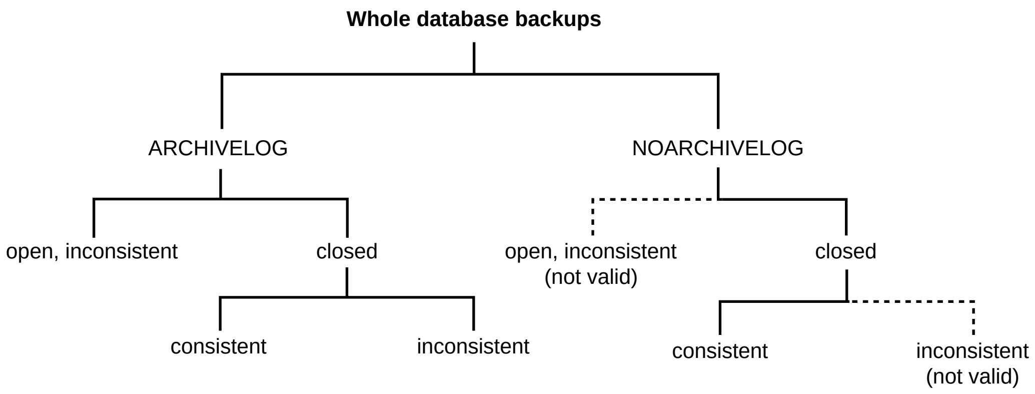
Whole Database Backups

A **whole database backup** includes backups of the current control file along with all datafiles. Whole database backups are the most common type of backup.

Whole database backups do not require you to operate the database in a specific archiving mode. Before performing whole database backups, however, be aware of the implications of backing up in `ARCHIVELOG` and `NOARCHIVELOG` modes (refer to "Database Archiving Modes" on page 1-11).

Figure 2–1 illustrates the valid configuration options given the type of backup that is performed.

Figure 2–1 Whole Database Backup Options



A whole database backup is either a **consistent backup** or an **inconsistent backup**. Whether or not a backup is consistent determines whether you need to apply redo logs after restoring the backup.

You can make backups of the entire database with the following methods:

- An operating system utility that makes a separate copy of each individual datafile in the database as well as the current control file
- The RMAN `BACKUP DATABASE` command
- The RMAN `COPY DATAFILE` command run against each datafile in the database, and the `COPY CURRENT CONTROLFILE` command run against the control file

Tablespace Backups

A tablespace backup is a backup of the datafiles that constitute the tablespace. For example, if tablespace `users` contains datafiles `2`, `3`, and `4`, then a backup of tablespace `users` backs up these three datafiles.

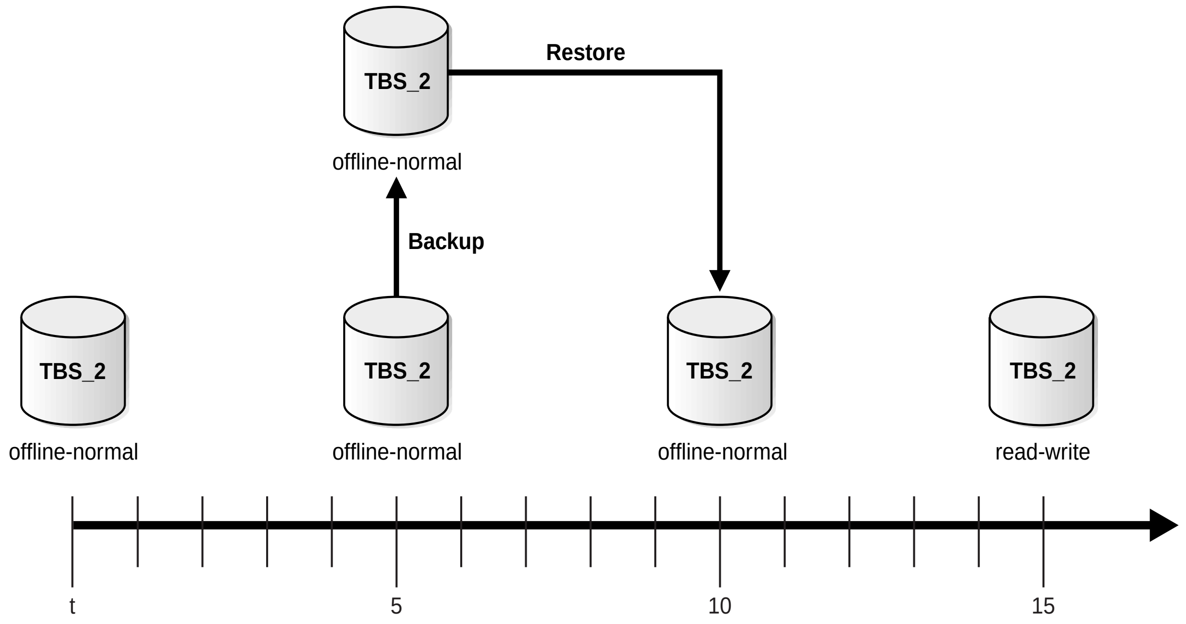
Tablespace backups, whether online or offline, are valid only if the database is operating in `ARCHIVELOG` mode. The reason is that redo is required to make the restored tablespace consistent with the other tablespaces in the database.

The only time a tablespace backup is valid for a database in `NOARCHIVELOG` mode is when the tablespace is currently read-only or offline-normal. These cases are exceptions because no redo is required to recover them.

For example, take the scenario depicted in Figure 2–2:

1. You take a tablespace offline normal at some time during day *t*.
2. You make a backup of the tablespace at day *t + 5*.
3. You restore the tablespace at day *t + 10* with the backup made at day *t + 5*.
4. You make the tablespace read / write at day *t* + 15.

Figure 2–2   Tablespace Backups in NOARCHIVELOG Mode



Because there were no changes to the offline tablespace between *t + 5* and *t + 10*, no media recovery is needed. If you make the tablespace read / write at *t + 15* and then subsequently attempt to restore the *t + 5* backup, however, Oracle requires media recovery for the changes after *t + 15*. Hence, you can only open the database if all necessary redo is located in the online redo logs.

You can make backups of an individual tablespace with the following methods:

- An operating system utility that makes a separate copy of each datafile in the tablespace
- The RMAN `BACKUP TABLESPACE` command
- The RMAN `COPY DATAFILE` command run against each datafile in the tablespace

> **See Also:** *Oracle9i Recovery Manager User's Guide* to learn how to make RMAN backups and copies, and *Oracle9i Recovery Manager Reference* for `BACKUP` and `COPY` syntax

Datafile Backups

A datafile backup is a backup of a single datafile. Datafile backups, which are not as common as tablespace backups, are valid in `ARCHIVELOG` databases. The only time a datafile backup is valid for a database in `NOARCHIVELOG` mode is if:

- Every datafile in a tablespace is backed up. You cannot restore the database unless all datafiles are backed up.
- The datafiles are read-only or offline-normal.

You can make backups of an individual datafile using these methods:

- An operating system utility
- The RMAN `BACKUP DATAFILE` command
- The RMAN `COPY DATAFILE` command, which produces a **datafile copy**

> **See Also:** *Oracle9i Recovery Manager User's Guide* to learn how to make RMAN backups and copies, and *Oracle9i Recovery Manager Reference* for `BACKUP` and `COPY` syntax

Control File Backups

Backing up the control file is a crucial aspect of backup and recovery. Without an accessible control file, you cannot mount or open the database.

If you use RMAN as your backup and recovery solution, and if you run the `CONFIGURE CONTROLFILE AUTOBACKUP` command, then RMAN automatically backs up the control file whenever you run backup and copy jobs. This backup is called a **control file autobackup**. Because the autobackup uses a default filename, RMAN can restore this backup even if the RMAN repository is unavailable. Hence, this feature is extremely useful in a disaster recovery scenario.

You can make manual backups of the control file by using the following methods:

- The RMAN `BACKUP CURRENT CONTROLFILE` creates an RMAN-specific backup of the control file, and the `COPY CURRENT CONTROLFILE` command creates an image copy of the control file.
- The SQL statement `ALTER DATABASE BACKUP CONTROLFILE` makes a binary backup of the control file.
- The SQL statement `ALTER DATABASE BACKUP CONTROLFILE TO TRACE` exports the control file contents to a SQL script file. You can use the script to create a new control file. Trace file backups have one major disadvantage: they contain no records of archived redo logs, offline ranges for datafiles, and RMAN backups and copies. For this reason, binary backups are preferable.

See Also: *Oracle9i Recovery Manager User's Guide* to learn how to make RMAN backups and copies, and *Oracle9i User-Managed Backup and Recovery Guide* to learn how to make user-managed control file backups

Archived Redo Log Backups

Archived redo logs are essential for recovering an inconsistent backup. The only way to recover an inconsistent backup without archived logs is to use RMAN incremental backups. To be able to recover a backup through the most recent log, every log generated between these two points must be available. In other words, you cannot recover from log 100 to log 200 if log 173 is missing. If log 173 is missing, then you must halt recovery at log 172 and open the database with the `RESETLOGS` option.

Because archived redo logs are essential to recovery, you should back them up regularly. If you use a media manager, then back them up regularly to tape.

You can make backups of archived logs by using the following methods:

- An operating system utility
- The RMAN `BACKUP ARCHIVELOG` command
- The RMAN `BACKUP ... PLUS ARCHIVELOG` or `BACKUP ... PLUS ARCHIVELOG` commands
- The RMAN `COPY ARCHIVELOG` command

See Also: *Oracle9i Recovery Manager User's Guide* to learn how to back up archived logs, and *Oracle9i User-Managed Backup and Recovery Guide* to learn how to make user-managed archived log backups

Consistent and Inconsistent Backups

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

Whether you make consistent or inconsistent backups depends on a number of factors. If your database must be open and available all the time, then inconsistent backups are your only option. If there are recurring periods of minimal use, then you may decide to take regular consistent backups of the whole database and supplement them with online backups of often-used tablespaces.

Consistent Backups

A consistent backup of a database or part of a database is a backup in which all read/write datafiles and control files have been checkpointed with respect to the same **system change number (SCN)**. In addition, every online, read/write datafile is not a **fuzzy file**, that is, does not contain changes beyond the SCN in the **datafile header**. Oracle determines whether a restored backup is consistent by checking the datafile headers against the datafile header information contained in the control file.

Oracle makes the control files and datafiles consistent to the same SCN during a database **thread checkpoint**. The only tablespaces in a consistent backup that are allowed to have older SCNs are read-only and offline normal tablespaces, which are still consistent with the other datafiles in the backup because no changes have been made to them. If the checkpoint SCN in the datafile header matches the **offline-start SCN** in the control file, then Oracle knows that the datafile needs no recovery.

The important point is that you can open the database after restoring a consistent whole database backup *without applying redo* because the data is already consistent: no action is required to make the data in the restored datafiles correct. Hence, you can restore a year-old consistent backup of your database without performing media recovery and without Oracle performing instance recovery.

The only way to make a consistent whole database backup is to shut down the database with the NORMAL, IMMEDIATE, or TRANSACTIONAL options and make the backup while the database is closed. If a database is not shut down cleanly, for example, an instance fails or you issue a SHUTDOWN ABORT statement, then the database's datafiles are always inconsistent—unless the database is a **read-only database**. Instance recovery will be required at open time.

A consistent whole database backup is the only valid backup option for databases operating in NOARCHIVELOG mode, because otherwise redo will need to be applied to create consistency. In NOARCHIVELOG mode, Oracle does not archive the redo logs, and so the required redo logs may not exist on disk.

Inconsistent Backups

An inconsistent backup is a backup in which all read / write datafiles and control files have not been checkpointed with respect to the same SCN. For example, one read / write datafile header may contain an SCN of 100 while other read / write datafile headers contain an SCN of 95 or 90. Oracle cannot open the database until all of these header SCNs are consistent, that is, until all changes recorded in the online redo logs have been applied to the datafiles on disk.

If the database must be up and running 24 hours a day, 7 days a week, then you have no choice but to perform inconsistent backups of a whole database. For example, a backup of an offline tablespace in an open database is inconsistent with other tablespaces because portions of the database are being modified and written to disk while the backup of the tablespace is progressing. The datafile headers for the online and offline datafiles may contain inconsistent SCNs. You must run your database in ARCHIVELOG mode to make online backups of online datafiles.

If you run the database in ARCHIVELOG mode, then you can construct a whole database backup using backups of online datafiles taken at different times. For example, if your database contains seven tablespaces, and if you back up the control file as well as a different tablespace each night, then in a week you will back up all tablespaces in the database as well as the control file. You can consider this staggered backup as a whole database backup.

Inconsistent Closed Backups

You have the option of making inconsistent closed backups if a database is backed up after a system crash or SHUTDOWN ABORT. This type of backup is valid if the database is running in ARCHIVELOG mode, because both online and archived redo logs are available to make the backup consistent.

Caution: Oracle strongly recommends that you do not make inconsistent, closed database backups in NOARCHIVELOG mode.

If you run the database in NOARCHIVELOG mode, then only back it up when you have closed it cleanly with the IMMEDIATE, NORMAL, or TRANSACTIONAL options. Inconsistent whole database backups of databases running in NOARCHIVELOG mode are usable *only if* the redo logs containing the changes made prior to the backup are available when you restore it—an unlikely occurrence.

The reason that NOARCHIVELOG inconsistent backups are not recommended is that the datafile headers of the backed up files contain different SCNs (a normal shutdown guarantees the consistency of these SCNs), and because the database is in NOARCHIVELOG mode, no archived redo logs are available to apply the lost changes. For this reason, RMAN does not allow you to back up a database that has been running in NOARCHIVELOG mode and shut down abnormally because the backup is not usable for recovery.

The basic guideline is: *if you run your database in* NOARCHIVELOG *mode, always have a backup that is usable without performing any recovery*. This aim is defeated if you need to apply redo from logs to recover a backup.

Archiving Unarchived Redo Log Files

After an **online backup** or inconsistent **closed backup**, always ensure that you have the redo necessary to recover the backup by archiving the unarchived redo logs. When the database is open, run the following SQL statement to force Oracle to switch out of the current log and archive it as well as all other unarchived logs:

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

When the database is mounted, open, or closed, you can run the following SQL statement to force Oracle to archive all noncurrent redo logs:

```
ALTER SYSTEM ARCHIVE LOG ALL;
```

When the database is mounted, open, or closed, you can run the following SQL statement to archive a specific group, where *integer* is the number of the group:

```
ALTER SYSTEM ARCHIVE LOG GROUP integer;
```

Backing Up the Archived Logs and the Control File

After open or inconsistent closed backups, Oracle recommends backing up all archived logs produced during the backup, and then backing up the control file after the backup completes. If you do not have all archived redo logs produced during the backup, then you cannot recover the backup because you do not have all the redo records necessary to make it consistent. Having a control file backup generated after the completion of the database backup is helpful when using RMAN because the control file contains a record of the backup (in V$BACKUP_SET).

Online and Offline Backups

This section contains these topics:

- Backups of Online Tablespaces and Datafiles
- Backups of Offline Tablespaces and Datafiles

Backups of Online Tablespaces and Datafiles

You can back up all or specified datafiles of an online tablespace while the database is open, but only when the database runs in ARCHIVELOG mode. In this case, Oracle can write changes to the online datafiles while the backup is occurring. A backup of online datafiles is called an **online backup**.

One danger in making online backups is the possibility of inconsistent data within a block. For example, assume that either RMAN or an operating system utility reads the entire block while database writer is in the middle of updating the block. In this case, RMAN or the copy utility may read the old data in the top half of the block and the new data in the bottom top half of the block. In this case, the block is a **fractured block**, meaning that the data contained in this block is not consistent.

During an RMAN backup, an Oracle server session reads the datafiles, not an operating system utility. The server session reads whole Oracle blocks and determines whether the block is fractured by comparing the header and footer of each block. If the session detects a fractured block, then it rereads the block until it gets a consistent picture of the data.

When you back up an individual datafile or online tablespace with an operating system utility (rather than with RMAN), you must use a different method to handle fractured blocks. You must first place the online tablespace in **backup mode** with the ALTER TABLESPACE BEGIN BACKUP statement. As a result, Oracle stops recording checkpoints to the tablespace's datafiles. You must put a tablespace in backup mode to make user-managed backups of datafiles in an online, read / write tablespace. After an online backup is completed, Oracle advances the file header to the current database checkpoint, but only after you run the ALTER TABLESPACE

`...` END BACKUP or ALTER DATABASE END BACKUP statement to take the tablespace out of backup mode.

When you restore a datafile from an operating system backup, the datafile header has a record of the most recent datafile checkpoint that occurred *before* the online tablespace backup, not any that occurred *during* it. As a result, Oracle asks for the appropriate set of redo log files to apply should recovery be needed.

Backups of Offline Tablespaces and Datafiles

An **offline backup** is performed while the tablespace or datafile is offline. You can take tablespaces offline with the ALTER TABLESPACE OFFLINE statement by using any of three different options: NORMAL, TEMPORARY, or IMMEDIATE. Taking an offline backup with the NORMAL option ensures that after the backup is complete, you do not have to perform recovery to bring the tablespace or datafile back online. In this way, you can perform necessary backups on datafiles and tablespaces without ever having to shut down the database or perform recovery.

RMAN and User-Managed Backups

This section contains these topics:

- RMAN Backups
- User-Managed Backups

RMAN Backups

RMAN backups are stored in a different format from user-managed backups. You generate an RMAN backup by running the BACKUP command from within the RMAN interface, as in the following example:

```
RMAN> BACKUP DATABASE;
```

The BACKUP command generates either a **backup set** or a **proxy copy** and writes it to the operating system or a third-party media manager (if used). A backup set is a logical construction composed of one or more backup pieces. A **backup piece** is a file in a proprietary format composed of the blocks from one or more input datafiles, control files, or archived redo logs. For example, you can back up 5 datafiles into 1 backup set containing 1 backup piece, which causes RMAN to intermingle the blocks from the different datafiles into a single file.

The format of a backup piece is "proprietary" in the sense that only RMAN can generate backup sets, and only RMAN can restore them. A proxy copy is a special type of RMAN backup whose data transfer is managed by a third-party media vendor. You must use the RMAN interface to create and restore proxy copies.

In contrast to the BACKUP command, the RMAN COPY command generates a datafile, control file, or archived log image copy that can be restored by an operating system utility. An image copy is an exact duplicate of the input file. For example, this command copies datafile 1 to df1.copy on the operating system:

```
RMAN> COPY DATAFILE 1 TO 'df1.copy';
```

The COPY command only copies to disk. However, you can use the BACKUP command to back up image copies to tape.

Whenever you use RMAN to make a backup or copy, it records the action in the target database control file. If you use a recovery catalog, then RMAN pulls the metadata from the control file into the catalog. When you want to restore the backups or copies, run the RESTORE command. RMAN queries the metadata and then chooses among the available backups and copies and restores them.

> **See Also:** *Oracle9i Recovery Manager User's Guide* to learn how to make RMAN backups, and *Oracle9i Recovery Manager Reference* for BACKUP syntax

User-Managed Backups

You must use operating system utilities to make user-managed backups. The available commands are operating system specific. For example, on a UNIX system you can back up a datafile using dd as follows:

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

On Windows NT, you can back up a datafile by pressing CTRL+C and then CTRL+V, by dragging and dropping, or by running a COPY command at the Command Prompt as in the following example:

```
C:\> COPY df1.dbf F:\BACKUP\df1.dbf
```

One major difference between user-managed backups and RMAN backups is that in the former there is no automatic metadata record of the backup. In other words, you must manually keep records of what you back up and where you back it up.

> **See Also:** *Oracle9i User-Managed Backup and Recovery Guide* to learn how to make backups using operating system utilities

3 Recovery Principles

This chapter introduces the structures that are used during database recovery. The topics in this chapter include:

- Types of Oracle Recovery
- Redo Application During Recovery
- Complete and Incomplete Media Recovery
- RMAN and User-Managed Restore and Recovery

See Also:

- *Oracle9i Recovery Manager User's Guide and Reference* for information about RMAN restore and recovery
- *Oracle9i User-Managed Backup and Recovery Guide* to learn how to perform user-managed restore and recovery

Types of Oracle Recovery

This section contains these topics:

- Instance and Crash Recovery
- Media Recovery

Instance and Crash Recovery

Crash recovery is used to recover from a failure either when a single-instance database crashes or all instances of an Oracle Real Application Clusters database crashes. Instance recovery refers to the case where a surviving instance recovers a failed instance in an Oracle Real Application Clusters database.

The goal of crash and instance recovery is to restore the data block changes located in the cache of the dead instance and to close the redo thread that was left open. Instance and crash recovery use only online redo log files and current online datafiles. Oracle recovers the **redo thread**s of the dead instances together.

Crash and instance recovery have the following shared characteristics:

- Redo the changes using the current online datafiles (as left on disk after the crash or `SHUTDOWN ABORT`)
- Use only the online redo logs and never require the use of the archived logs
- Have a recovery time governed by the number of dead instances, amount of redo generated in each dead redo thread since the last checkpoint, and by user-configurable factors such as the number and size of redo log files, checkpoint frequency, and the parallel recovery setting

Oracle performs this recovery automatically on two occasions:

- At the first database open after the crash of a single-instance database or all instances of an Oracle Real Applications Cluster database (crash recovery). After a normal shutdown, Oracle sets a flag in the control file. Oracle performs crash recovery automatically at startup upon detecting that an instance did not set a flag in the control file indicating a normal shutdown.
- When some but not all instances of an Oracle Real Application Clusters configuration fail (instance recovery). The recovery is performed automatically by a surviving instance in the configuration.

The important point is that in both crash and instance recovery Oracle applies the redo automatically: no user intervention is required to supply redo logs. However, you can set parameters in the database server that can tune the duration of instance and crash recovery performance. Also, you can tune the rolling forward and rolling back phases of instance recovery separately. Finally, you can tune checkpointing so that recovery time is optimized.

> **See Also:** *Oracle9i Database Performance Guide and Reference* for a discussion of instance recovery mechanics as well as instructions for tuning instance and crash recovery

Media Recovery

Media recovery is divided into the following types:

- Datafile media recovery
- Block media recovery

Typically, the term "media recovery" refers to recovery of datafiles. Block media recovery is a more specialized operation that you can only perform with RMAN.

Datafile Media Recovery

Datafile media recovery is used to recover from a lost or damaged current datafile or control file. It is also used to recover changes that were lost when a tablespace went offline without the `OFFLINE NORMAL` option. Datafile media recovery and instance recovery have in common the requirement to repair database integrity. However, these types of recovery differ with respect to their additional features. Media recovery has the following characteristics:

- Applies needed changes using restored backups of damaged datafiles.
- Can use archived logs as well as the online logs.
- Requires explicit invocation by a user.
- Does not detect media failure (that is, the need to restore a backup) automatically. After a backup has been restored, however, detection of the need to recover it through media recovery *is* automatic.
- Has a recovery time governed solely by user policy (for example, frequency of backups, parallel recovery parameters) rather than by Oracle internal mechanisms.

The database cannot be opened if any of the online datafiles needs media recovery, nor can a datafile that needs media recovery be brought online until media recovery has been executed. The following scenarios necessitate 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 Oracle) without the `OFFLINE NORMAL` option.

Unless the database is not open by any instance, datafile media recovery can only operate on offline datafiles. You can initiate datafile media recovery before opening a database even when crash recovery would have sufficed. If so, crash recovery still runs automatically at database open.

Note that when a file requires media recovery, you *must* perform media recovery even if all necessary changes are contained in the online logs. In other words, you must still run recovery even though the archived logs are not needed. Media recovery may find nothing to do — and signal the "no recovery required" error — if invoked for files that do not need recovery.

Block Media Recovery

Block media recovery is a technique for restoring and recovering individual data blocks while all database files remain online and available. If corruption is limited to only a few blocks among a subset of database files, then block media recovery may be preferable to datafile recovery.

The interface to block media recovery is provided by RMAN. If you do not already use RMAN as your principal backup and recovery solution, then you can still perform block media recovery by cataloging into the RMAN repository the necessary user-managed datafile and archived redo log backups.

> **See Also:** *Oracle9i Recovery Manager User's Guide* to learn how to catalog user-managed datafile and archived log backups and to perform block media recovery

Redo Application During Recovery

Media recovery proceeds through the application of redo data to the datafiles. Whenever a change is made to a datafile, the change is first recorded in the online redo logs. Media recovery selectively applies the changes recorded in the online and archived redo logs to the restored datafile to roll it forward.

This section contains these topics:

- About Redo Application
- Cache Recovery
- Transaction Recovery

About Redo Application

Database buffers in the buffer cache in the SGA are written to disk only when necessary, using a least-recently-used (LRU) algorithm. Because of the way that the database writer process uses this algorithm to write database buffers to datafiles, datafiles may contain some data blocks modified by uncommitted transactions and some data blocks missing changes from committed transactions.

Two potential problems can result if an instance failure occurs:

- Data blocks modified by a transaction might not be written to the datafiles at commit time and may only appear in the redo log. Therefore, the redo log contains changes that must be reapplied to the database during recovery.
- After the roll forward phase, the datafiles may contain changes that had not been committed at the time of the failure. These uncommitted changes must be rolled back to ensure transactional consistency. These changes were either saved to the datafiles before the failure, or introduced during the roll forward phase.

To solve this dilemma, two separate steps are generally used by Oracle for a successful recovery of a system failure: rolling forward with the redo log (cache recovery) and rolling back with the rollback or undo segments (transaction recovery).

Cache Recovery

The **online redo log** is a set of operating system files that record all changes made to any database buffer, including data, index, and rollback segments, *whether the changes are committed or uncommitted.* All changes to Oracle blocks are recorded in the online log.

The first step of recovery from an instance or disk failure is called **cache recovery** or **rolling forward**, and involves reapplying all of the changes recorded in the redo log to the datafiles. Because rollback data is also recorded in the redo log, rolling forward also regenerates the corresponding rollback segments

Rolling forward proceeds through as many redo log files as necessary to bring the database forward in time. Rolling forward usually includes online redo log files (instance recovery or media recovery) and may include archived redo log files (media recovery only).

After rolling forward, the data blocks contain all committed changes. They may also contain uncommitted changes that were either saved to the datafiles before the failure, or were recorded in the redo log and introduced during cache recovery.

Transaction Recovery

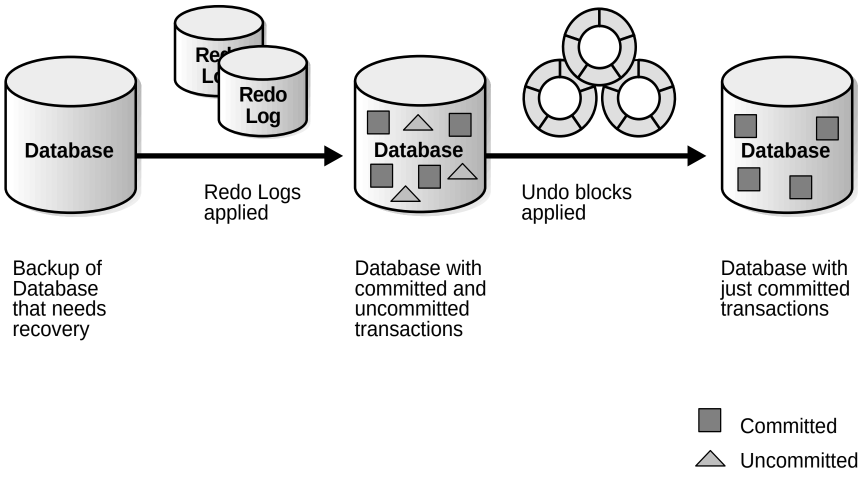
You can run Oracle in either **manual undo management mode** or **automatic undo management mode**. In manual mode, you must create and manage **rollback segments** to record the before-image of changes to the database. In automatic undo management mode, you create one or more undo tablespaces. These undo tablespaces contain undo segments similar to traditional rollback segments. The main difference is that Oracle manages the undo for you.

Undo blocks (whether in rollback segments or automatic undo tablespaces) record database actions that should be undone during certain database operations. In database recovery, the undo blocks roll back the effects of uncommitted transactions previously applied by the rolling forward phase.

After the roll forward, any changes that were not committed must be undone. Oracle applies undo blocks to roll back uncommitted changes in data blocks that were either written before the crash or introduced by redo application during cache recovery. This process is called **rolling back** or **transaction recovery**.

Figure 3–1 illustrates rolling forward and rolling back, the two steps necessary to recover from any type of system failure.

Figure 3–1 Basic Recovery Steps: Rolling Forward and Rolling Back


Database
Redo Log
Redo Log
Database
Database
Redo Logs applied
Undo blocks applied
Backup of Database that needs recovery
Database with committed and uncommitted transactions
Database with just committed transactions
Committed
Uncommitted

Oracle can roll back multiple transactions simultaneously as needed. All transactions systemwide that were active at the time of failure are marked as dead.

Instead of waiting for SMON to roll back dead transactions, new transactions can recover blocking transactions themselves to get the row locks they need.

Complete and Incomplete Media Recovery

Media recovery updates a backup to either to the current or to a specified noncurrent time. When performing media recovery, you can recover the whole database, a tablespace, or a datafile. In any case, you always use a restored backup to perform the recovery.

This section contains the follow topics:

- Complete Recovery
- Incomplete Recovery

Complete Recovery

Complete recovery involves using redo data or incremental backups combined with a backup of a database, tablespace, or datafile to update it to the most current point in time. It is called *complete* because Oracle applies *all* of the redo changes contained in the archived and online logs to the backup. Typically, you perform complete media recovery after a media failure damages datafiles or the control file.

You can perform complete recovery on a database, tablespace, or datafile. If you are performing complete recovery on the whole database, then whether you are using RMAN or SQL*Plus you must:

- Mount the database
- Ensure that all datafiles you want to recover are online
- Restore a backup of the whole database or the files you want to recover
- Apply online or archived redo logs, or a combination of the two

If you are performing complete recovery on a tablespace or datafile, then you must:

- Take the tablespace or datafile to be recovered offline if the database is open
- Restore a backup of the datafiles you want to recover
- Apply online or archived redo logs, or a combination of the two

Incomplete Recovery

Incomplete recovery uses a backup to produce a noncurrent version of the database. In other words, you do not apply all of the redo records generated after the most recent backup. You usually perform incomplete recovery of the whole database in the following situations:

- Media failure destroys some or all of the online redo logs.
- A user error causes data loss, for example, a user inadvertently drops a table.
- You cannot perform complete recovery because an archived redo log is missing.
- You lose your current control file and must use a backup control file to open the database.

To perform incomplete media recovery, you must restore all datafiles from backups created prior to the time to which you want to recover and then open the database with the `RESETLOGS` option when recovery completes. The `RESETLOGS` operation creates a new **incarnation** of the database—in other words, a database with a new stream of log sequence numbers starting with log sequence 1.

Tablespace Point-in-Time Recovery

The tablespace point-in-time recovery (TSPITR) feature enables you to recover one or more tablespaces to a point-in-time that is different from the rest of the database. TSPITR is most useful when you want to:

- Recover from an erroneous drop or truncate table operation
- Recover a table that has become logically corrupted
- Recover from an incorrect batch job or other DML statement that has affected only a subset of the database
- Recover one independent schema to a point different from the rest of a physical database (in cases where there are multiple independent schemas in separate tablespaces of one physical database)
- Recover a tablespace on a very large database (VLDB) rather than restore the whole database from a backup and perform a complete database roll-forward

See Also: *Oracle9i User-Managed Backup and Recovery Guide* to learn how to perform user-managed TSPITR, and *Oracle9i Recovery Manager User's Guide* to learn how to perform TSPITR with RMAN.

Media Recovery Options

Because you are not completely recovering the database to the most current time, you must tell Oracle when to terminate recovery. You can perform the following types of media recovery.

Type of Recovery	Function
Time-based recovery	Recovers the data up to a specified point in time.
Cancel-based recovery	Recovers until you issue the `CANCEL` statement (not available when using Recovery Manager).
Change-based recovery	Recovers until the specified SCN.
Log sequence recovery	Recovers until the specified log sequence number (only available when using Recovery Manager).

RMAN and User-Managed Restore and Recovery

You have a choice between two basic methods for recovering physical files. You can:

- Use the RMAN utility to restore and recover the database
- Restore backups by means of operating system utilities, and then recover by executing the SQL*Plus `RECOVER` command

Whichever method you choose, you can recover a database, tablespace, or datafile. Before performing media recovery, you need to determine which datafiles to recover. Often you can use the fixed view `V$RECOVER_FILE`. This view lists all files that require recovery and explains the error that necessitates recovery.

See Also: *Oracle9i Recovery Manager User's Guide* for more about using `V$` views in a recovery scenario

RMAN Restore and Recovery

The basic RMAN recovery commands are `RESTORE` and `RECOVER`. Use `RESTORE` to restore datafiles from backup sets or from image copies on disk, either to their current location or to a new location. You can also restore backup sets containing archived redo logs. Use the RMAN `RECOVER` command to perform media recovery and apply archived logs or incremental backups.

RMAN automates the procedure for recovering and restoring your backups and copies. For example, run the following commands from within RMAN to restore and recover the database to its current time:

```
SHUTDOWN IMMEDIATE; # shuts down database
STARTUP MOUNT; # starts and mounts database
RESTORE DATABASE; # restores all datafiles
RECOVER DATABASE; # recovers database using all available redo
ALTER DATABASE OPEN; # reopens the database
```

> **See Also:** *Oracle9i Recovery Manager User's Guide and Reference* to learn how to restore and recovery using RMAN

User-Managed Restore and Recovery

If you do not use RMAN, then you can restore backups with operating system utilities and then run the SQL*Plus `RECOVER` command to recover the database. You should follow these basic steps:

1. After identifying which files are damaged, place the database in the appropriate state for restore and recovery. For example, if some but not all datafiles are damaged, then take the affected tablespaces offline while the database is open.
2. Restore the files with an operating system utility. If you do not have a backup, it is sometimes possible to perform recovery if you have the necessary redo logs dating from the time when the datafiles were first created and the control file contains the name of the damaged file.

   If you cannot restore a datafile to its original location, then relocate the restored datafile and change the location in the control file.
3. Restore any necessary archived redo log files.
4. Use the SQL*Plus `RECOVER` command to recover the datafile backups.

For example, assume that you lose the `/oracle/dbs/users1.dbf` datafile, which is contained in the `users` tablespace, to a media failure. Also, assume that you have a backup called `/dsk2/backup/users1.dbf` on a separate disk drive. You discover that the datafile is missing because a query returns an error saying that this file (and only this file) is missing.

Your first step is to take the `users` tablespace offline. For example, you run this SQL statement:

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

Then, you restore the backup of `users1.dbf` using an operating system utility. For example, you run this UNIX command:

```
% cp /dsk2/backup/users1.dbf /oracle/dbs/users1.dbf
```

Assuming that you have all necessary archived redo logs, you can recover the datafile with the following SQL*Plus command:

```
SQL> RECOVER AUTOMATIC DATAFILE '/oracle/dbs/users1.dbf';
```

Finally, bring the tablespace online as follows:

```
SQL> ALTER TABLESPACE users ONLINE;
```

See Also: *Oracle9i User-Managed Backup and Recovery Guide* to learn how to restore and recover by means of operating system utilities and SQL*Plus

4 Backup and Recovery Strategies

This chapter offers guidelines and considerations for developing an effective backup and recovery strategy. It includes the following topics:

- Backup Strategies
- Restore and Recovery Strategies

Backup Strategies

Before you create an Oracle database, decide how to protect the database against potential media failures. If you do not develop a backup strategy before creating your database, then you may not be able to perform recovery if a disk failure damages the datafiles, online redo log files, or control files.

This section describes general guidelines that can help you decide when to perform database backups and which parts of a database you should back up. Of course, the specifics of your strategy depend on the constraints under which you are operating.

This section contains these topics:

- Obeying the Golden Rule of Backup and Recovery
- Choosing the Database Archiving Mode
- Multiplexing Control Files, Online Redo Logs, and Archived Redo Logs
- Performing Backups Frequently and Regularly
- Performing Backups Before and After You Make Structural Changes
- Backing Up Often-Used Tablespaces
- Performing Backups After Unrecoverable Operations
- Performing Whole Database Backups After Opening with the RESETLOGS Option
- Archiving Older Backups
- Knowing the Constraints for Distributed Database Backups
- Exporting Data for Added Protection and Flexibility
- Avoiding the Backup of Online Redo Logs
- Keeping Records of the Hardware and Software Configuration of the Server

Obeying the Golden Rule of Backup and Recovery

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

- The last backup of the control file and all the datafiles
- All archived redo logs generated after the last backup was taken

- A duplicate of the online redo log files generated by Oracle multiplexing, operating system mirroring, or both
- A duplicate of the current control file generated by Oracle multiplexing, operating system mirroring, or both
- Configuration files such as the server parameter file, `tnsnames.ora`, and `listener.ora`

The golden rule of backup and recovery is: *the set of disks or other media that contain the redundancy set should be separate from the disks that contain the datafiles, online redo logs, and control files.* This strategy 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.

Always keep the redundancy set separate from the primary files in every way possible: on separate volumes, separate file systems, and separate RAID devices. These systems are reliable, but they can and do fail. Keeping the redundancy set separate ensures that you can recover from a failure without losing committed transactions.

You can implement a system that follows the golden rule in several different ways. Oracle recommends following these guidelines:

- Multiplex the online redo log files and current control file at the Oracle level, not only at the operating system or hardware level. Multiplexing at the Oracle level has the advantage that an I/O failure or lost write should only corrupt one of the copies.
- Use operating system or hardware mirroring for at least the control file, because Oracle does not provide complete support for control file multiplexing: if one multiplexed copy of the control file fails, then the Oracle instance shuts down.
- Use operating system or hardware mirroring for the primary datafiles if possible to avoid having to apply media recovery for simple disk failures.
- Keep at least one copy of the entire redundancy set—including the most recent backup—on hard disk.

  If the redundancy copy is created by splitting a local mirror, then it is not as good as a backup created through operating system or RMAN commands because it relies on the mirroring subsystem for both the primary files *and* redundancy set copy. The last file backup, such as the last backup to tape, is the redundancy set copy. Hence, keep archived logs needed to recover this copy.

- If your database is stored on a RAID device, then place the redundancy set on a set of devices that is not in the same RAID device.
- If you keep the redundancy set on tapes, then maintain at least two copies of the data because tapes can fail. 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.

Choosing the Database Archiving Mode

Before you create an Oracle database, decide how you plan to protect it against potential failures. Answer the following questions:

- **Is it acceptable to lose any data if a disk failure damages some of the files that constitute a database?**

  If not, then run the database in ARCHIVELOG mode, ideally with a multiplexed online redo log, a multiplexed control file, and multiplexed archive redo logs. If you can afford to lose all data from your last backup to the point of failure, then you can operate in NOARCHIVELOG mode and avoid the extra maintenance chores. You may have alternative ways of re-creating the data.

- **Will you need to recover to a noncurrent time?**

  If you need to perform incomplete recovery to correct an erroneous change to the database, then run in ARCHIVELOG mode and perform control file backups whenever making structural changes. Incomplete recovery is easiest when you have a backup control file reflecting the database structure at the desired time.

- **Does the database need to be available at all times?**

  High-availability databases always operate in ARCHIVELOG mode to take advantage of open datafile backups.

After you have answered these questions and determined which mode to use, follow the guidelines for either:

- Backing Up a NOARCHIVELOG Database
- Backing Up an ARCHIVELOG Database

Backing Up a NOARCHIVELOG Database

If you run the database in `NOARCHIVELOG` mode, Oracle does not archive filled groups of online redo log files. Therefore, the only protection against a disk failure is the most recent whole backup of the database. Follow these guidelines:

- Make whole database backups regularly, according to the amount of work that you can afford to lose. For example, if you can afford to lose the amount of work accomplished in one week, then make a consistent whole database backup once every week. If you can afford to lose only a day's work, then make a consistent whole database backup every day. For large databases with a high amount of activity, you usually cannot afford to lose work. In this case, you should operate the database in `ARCHIVELOG` mode.
- Whenever you alter the physical structure of a database operating in `NOARCHIVELOG` mode, immediately take a consistent whole database backup. A whole database backup fully reflects the new structure of the database.

Backing Up an ARCHIVELOG Database

If you run your database in `ARCHIVELOG` mode, then the archiver archives groups of online redo log files. Therefore, the archived redo log coupled with the online redo log and datafile backups can protect the database from a disk failure, providing for complete recovery from a disk failure to the instant that the failure occurred (or, to the desired noncurrent time). Following are common backup strategies for a database operating in `ARCHIVELOG` mode:

- Back up the entire database after you create it. This initial whole database backup is the foundation of your backups because it provides backups of all datafiles and the control file of the associated database.

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

- Make backups of tablespaces when the database is open or closed to keep the database backups up-to-date. So long as you have the necessary archived logs to recover the backup, you never have to shut down the database to make a backup.

  In particular, back up the datafiles of extensively used tablespaces frequently to reduce database recovery time. If a more recent datafile backup restores a damaged datafile, then you need to apply less redo (or incremental backups) to the restored datafile to roll it forward to the time of the failure.

You can also use a datafile copy taken while the database is open and the tablespace is online to restore datafiles. You must apply the appropriate redo log files to these restored datafiles to make the data consistent and bring it forward to the specified point in time.

- Back up the control file every time you make a structural change to the database. If you run in `ARCHIVELOG` mode and the database is open, then use either RMAN or the SQL statement `ALTER DATABASE BACKUP CONTROLFILE`.
- Back up archived logs frequently. It is strongly recommended that you keep at least two copies of archived logs: one on disk and another on off-line storage (tape, optical disks, and so forth). Keep the logs on disk as long as possible but back them up as soon as possible.

Multiplexing Control Files, Online Redo Logs, and Archived Redo Logs

Control files, online redo logs, and archived redo logs are crucial files for backup and recovery operations. The loss of any of these files can cause you to lose data irrevocably. You should maintain:

- At least two copies of the control file on different disks mounted under different disk controllers. You should use Oracle to multiplex the copies and your operating system to mirror each copy.
- Two or more copies of your online redo log on different disks. The online redo data is crucial for instance, crash, and media recovery.
- Two or more copies of your archived redo log on different disks and, if possible, different media.

See Also: *Oracle9i Database Concepts* for a conceptual overview of all Oracle data structures.

Performing Backups Frequently and Regularly

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

If users generate a significant amount of DML, then database backup frequency should be proportionally high. Alternatively, if a database is mainly read-only, and if updates are issued only infrequently, then you can back up the database less frequently.

You can use either RMAN or user-managed methods to create backup scripts. If you set persistent configurations using RMAN's `CONFIGURE` command, however, then you should not typically need to write extensive scripts. You can regularly run `BACKUP DATABASE PLUS ARCHIVELOG`.

> **See Also:** *Oracle9i Recovery Manager User's Guide and Reference* to learn how to create, delete, replace, and print stored scripts

Performing Backups Before and After You Make Structural Changes

Administrators as well as users make changes to a database. 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 `ALTER DATABASE` statement with the `BACKUP CONTROLFILE` option) before and 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 before and after the modification.

Backing Up Often-Used Tablespaces

Many DBAs find that regular whole database backups are not in themselves sufficient for a robust backup strategy. If you run in `ARCHIVELOG` mode, then you can back up the datafiles of an individual tablespace or even a single datafile. This option is useful if a portion of a database is used more extensively than others, for example, the `SYSTEM` tablespace and automatic undo tablespaces.

By making more frequent backups of the extensively used datafiles of a database, you avoid a long recovery time. For example, you may make a whole database backup once every two weeks. If the database experiences heavy traffic during the week, then a media failure on Friday can force you to apply a tremendous amount of redo during recovery. If you had backed up your most frequently accessed tablespaces three times a week, then you could apply a smaller number of changes to roll the restored file forward to the time of the failure.

If you are running in automatic undo management mode, then be sure to regularly back up your undo tablespaces. If you run in manual undo management mode, then be sure to regularly back up all tablespaces containing rollback segments.

See Also: *Oracle9i Database Administrator's Guide* for information about managing undo tablespaces

Performing Backups After Unrecoverable Operations

If users are creating tables or indexes using the `UNRECOVERABLE` option, then make backups after the objects are created. When tables and indexes are created as `UNRECOVERABLE`, Oracle does not log redo data, which means that you cannot recover these objects from existing backups.

Note: If using RMAN, then you can make an incremental backup.

See Also: *Oracle9i SQL Reference* for information about the `UNRECOVERABLE` option of the `CREATE TABLE ... AS SELECT` and `CREATE INDEX` statements.

Performing Whole Database Backups After Opening with the RESETLOGS Option

After you open a database with the `RESETLOGS` option, Oracle Corporation recommends that you immediately perform a whole database backup. If you do not, and if a disaster occurs, then it is possible to lose all changes made after opening the database.

In certain cases, you can restore a backup made prior to a `RESETLOGS` and recover the database, but the procedure is complicated and requires you to have a control file backup from before and after the `RESETLOGS` operations. A whole database backup created after a `RESETLOGS` protects against this situation.

See Also: *Oracle9i Recovery Manager User's Guide and Reference* to learn how to recover using a backup created before a `RESETLOGS`

Archiving Older Backups

You may need to store older backups for two basic reasons:

- An older backup is necessary for performing incomplete recovery to a time before your most recent backup
- Your most recent backup is corrupted

If you want to recover to a noncurrent time, then you need a database backup that completed before the desired time. For example, if you make backups on the 1st and 14th of February, then decide at the end of the month to recover your database to February 7th, you must use the February 1st (or earlier) backup.

For a database operating in `NOARCHIVELOG` mode, the backup that you use must be a consistent whole database backup. Of course, you cannot perform media recovery using this backup. For a database operating in `ARCHIVELOG` mode, the whole database backup:

- Does not need to be consistent because redo is available to recover it
- Should have completed before the intended recovery time
- Should have all archived logs necessary to recover the datafiles to the required point-in-time
- Should be recovered with a control file that reflects the database's structure at the point-in-time that ends the recovery

For added protection, keep two or more database backups (with associated archived redo logs) previous to the current backup. Thus, if your most recent backups are not usable, then you will not lose all of your data.

Knowing the Constraints for Distributed Database Backups

If the database is a member of a distributed database system, then all databases in the system should operate in the same archiving mode. Note the consequences and constraints contained in the following table.

Mode	Constraint	Consequence
`ARCHIVELOG`	Closed cleanly	Backups at each node can be performed autonomously, that is, individually and without time coordination.
`NOARCHIVELOG`	Closed cleanly	Consistent whole database backups must be performed at the same global time to plan for global distributed database recovery. For example, if a database in New York is backed up at midnight EST, the database in San Francisco should be backed up at 9 PM PST.

See Also: *Oracle9i Database Administrator's Guide* to learn how to manage distributed database systems

Exporting Data for Added Protection and Flexibility

Because the Oracle Export utility can selectively export specific objects, consider exporting portions or all of a database for supplemental protection and flexibility in a database's backup strategy. This strategy is especially useful for logical backups of the RMAN recovery catalog, because you can quickly reimport this data into any database and rebuild the catalog if the recovery catalog database is lost.

Database exports are not a substitute for whole database backups and cannot provide the same complete recovery advantages that the built-in functionality of Oracle offers. For example, you cannot apply archived logs to logical backups in order to update lost changes. An export provides a snapshot of the logical data (tables, stored procedures, and so forth) in a database when the export was made.

See Also: *Oracle9i Database Utilities* for an account of the Export utility

Avoiding the Backup of Online Redo Logs

Although it may seem that you should back up online redo logs along with the datafiles and control file, this technique is dangerous. You should not back up online redo logs for the following reasons:

- The best method for protecting the online logs against media failure is by multiplexing them, that is, having multiple log members in each group, on different disks and disk controllers.

- If your database is in ARCHIVELOG mode, then the archiver is already archiving the filled redo logs.
- If your database is in NOARCHIVELOG mode, then the only type of backups that you should 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 needed.
- You may accidentally restore backups of online redo logs while not intending to, thereby corrupting the database.

A number of situations are possible in which restoring the online logs cause significant problems to the database. The following sections describe scenarios that illustrate how restoring backed up online logs severely compromises recovery.

Unintentionally Restoring Online Redo Logs: Scenario

When a crisis occurs, it is easy to make a simple mistake. When restoring the whole database, you can accidentally restore the online redo logs, thus overwriting the current online logs with the older, useless backups. This action forces you to perform incomplete recovery instead of the intended complete recovery, thereby losing the ability to recover valuable data contained in the overwritten redo logs.

Erroneously Creating Multiple Parallel Redo Log Timelines: Scenario

If you face a problem where the best course of action is to restore the database from a consistent backup and not perform any recovery, then you may think it is safe to restore the online logs and thereby avoid opening the database with the RESETLOGS option. The problem is that Oracle eventually generates a log sequence number that was already generated by the database during the previous timeline.

For example, say that the most recent archived log for database prod1 has a log sequence number of 100. Assume that you restore a consistent backup of the database along with backed up online redo logs and then do *not* open with the RESETLOGS option. Assume also that the restored online log is at log sequence 50. Eventually, the database archives a log with the log sequence number of 100—so you now have two copies of log 100 with completely different contents.

If you then face another disaster and need to restore from this backup and roll forward, then you may find it difficult to identify which log with sequence number 100 is the correct one. If you had reset the logs, then you would have created a new incarnation of the database. You could only apply archived logs created by this new incarnation to this incarnation.

Note: RMAN does not permit you to back up online redo logs.

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 of the node that hosts the database
- The make and model of the production machine
- The version and patch of the operating system
- The disk capacity of the host
- The number of disks and disk controllers
- The disk capacity and free space
- The media management vendor (if you use a third-party media manager)
- The type and number of media management devices

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)
- The datafile mount points

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 this data available.

Restore and Recovery Strategies

Oracle provides a variety of procedures and tools to assist you with recovery. To develop an effective recovery strategy, do the following:

- Testing Backup and Recovery Strategies
- Validating Backups and Restores Using RMAN
- Planning a Response to Media Failures
- Planning a Response to Datafile Block Corruption
- Planning the Response to Non-Media Failures

Testing Backup and Recovery Strategies

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.

When testing your backup and recovery strategy, ask yourself these questions:

- If a disk failed and destroyed some of the database files, could I perform a full recovery of the files on this disk? Test separately for loss of datafiles, control files, and online redo logs.
- If a user accidentally dropped a table, how could I recover from it? Test scenarios involving incomplete recovery of the whole database, tablespace point-in-time recovery, and using the Import utility.
- What if the `alert_`*`SID`*`.log` revealed that one or more tables contained corrupt blocks? Test block recovery using the RMAN `BLOCKRECOVER` command. Also, troubleshoot recovery with the SQL*Plus `RECOVER ... TEST` command.

- If the entire data center was destroyed by a fire, could you perform disaster recovery? Assume that all you have is an archived tape containing backups. How would you recover the database?

> **See Also:** *Oracle9i Recovery Manager User's Guide and Reference* for RMAN testing methods, and *Oracle9i User-Managed Backup and Recovery Guide* to learn how to troubleshoot SQL*Plus recovery

Validating Backups and Restores Using RMAN

If you use RMAN, then you can use the `VALIDATE` keyword on the `BACKUP` and `RESTORE` commands. `BACKUP VALIDATE` tests whether you are able to make a valid backup of database files. `RESTORE VALIDATE` tests whether you are able to restore an RMAN backup. Note that neither of these commands produces any actual output files.

Planning a Response to Media Failures

Media failure is the biggest threat to your data. A media failure is a physical problem that occurs when a computer unsuccessfully attempts to read from or write to a file necessary to operate the database. Common types of media problems include:

- A disk drive that holds one of the database files experiences a head crash.
- A datafile, online or archived redo log, or control file is accidentally deleted, overwritten, or corrupted.

The technique you use to recover from media failure of a database file depends heavily on the type of media failure that occurred. 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.

The basic steps for media recovery are:

- Determine which files to recover.
- Determine the type of media recovery required: complete or incomplete, open database or closed database.
- Restore backups or copies of necessary files: datafiles, control files, and the archived redo logs necessary to recover the datafiles.

Note: If you do not have a backup, then you can still perform recovery if you have the necessary redo logs and the control file contains the name of the damaged file. If you cannot restore a file to its original location, then you must relocate the restored file and rename the file in the control file.

- Apply redo records (and/or incremental backups when using Recovery Manager) to recover the datafiles.
- Reopen the database. If you perform incomplete recovery or restore a backup control file, then you must open the database with the `RESETLOGS` option.

See Also: *Oracle9i Recovery Manager User's Guide and Reference* to learn how to perform media recovery using RMAN

When you perform datafile media recovery, you choose either complete recovery or incomplete recovery. The type of recovery method you use depends on the situation. Table 4–1 displays typical scenarios and strategies.

Table 4–1 Typical Media Failures and Recovery Strategies

Lost/Inaccessible Files	Archiving Mode	Status	Strategy
One or more datafiles	`NOARCHIVELOG`	Closed	Restore whole database from a consistent database backup. All changes made after the backup are lost. Open the database with the `RESETLOGS` option. **Note:** The *only* time you can open a database without performing `RESETLOGS` after restoring a `NOARCHIVELOG` backup is when you have not already overwritten the online log files that were current at the time of the most recent backup.
One or more datafiles and an online redo log	`NOARCHIVELOG`	Closed	Restore whole database from consistent backup. You lose all changes made after the last backup. Open the database with the `RESETLOGS` option.
One or more datafiles and all control files	`NOARCHIVELOG`	Closed	Restore the whole database and control file from consistent backup. You lose all changes made after the last backup. Open the database with the `RESETLOGS` option.

Table 4–1 Typical Media Failures and Recovery Strategies (Cont.)

Lost/Inaccessible Files	Archiving Mode	Status	Strategy
One or more (but not all) datafiles	`ARCHIVELOG`	Open	Perform tablespace or datafile recovery while the database is open. The tablespaces or datafiles are taken offline, restored from backups, recovered, and placed online. No changes are lost and the database remains available during the recovery.
All datafiles	`ARCHIVELOG`	Closed	Restore the backup datafiles, then mount the control file and recover the database completely. Assuming all redo logs are available, you can open the database as normal (that is, do not perform a `RESETLOGS`).
One or more online redo log groups	`ARCHIVELOG`	Closed	Perform incomplete recovery of the database up to the point of the lost online redo log. Open the database with the `RESETLOGS` option.
One or more datafiles and an archived redo log required for recovery	`ARCHIVELOG`	Open	Perform TSPITR on the tablespaces containing the lost datafiles up to the point of the latest available archived redo log.
All control files and possibly one or more datafiles	`ARCHIVELOG`	Not open	Restore the lost control files and datafiles from backups and recover the datafiles. No changes are lost, but the database is unavailable during recovery. Open the database with the `RESETLOGS` option.
All control files and possibly one or more datafiles, as well as an archived or online redo log required for recovery	`ARCHIVELOG`	Not open	Restore the necessary files from backups, then perform incomplete recovery of the database up to the point of the most recent available log. You will lose all changes contained in the lost log and in all subsequent logs. Open the database with the `RESETLOGS` option.

Planning a Response to Datafile Block Corruption

If selected blocks within a datafile are corrupt, then you may not have to restore and recover the whole datafile. Instead, you can perform **block media recovery**. The Recovery Manager `BLOCKRECOVER` command can restore and recover specified data blocks while the database is open and the corrupted datafile is online.

See Also: *Oracle9i Recovery Manager User's Guide and Reference* to learn how to perform block media recovery

Planning the Response to Non-Media Failures

Although media recovery is your primary concern when developing your recovery strategy, you should understand the basic types of non-media failures as well as the causes and solutions for each.

Statement Failure

A statement failure is a logical failure in the handling of a statement in an Oracle program. The Oracle database server or the operating system usually returns an error code and a message when a statement failure occurs.

User Error

Users errors are any mistakes that users make in adding data to or deleting data from the database. If you have a logical backup of a table from which data has been lost, sometimes you can simply import it back into the table.

Depending on the scenario, you may have to perform some type of incomplete media recovery to correct user errors. You can perform either database point-in-time recovery (DBPITR) or tablespace point-in-time recovery (TSPITR). The following table explains the difference between these types of incomplete recovery.

Type	Description
DBPITR	**1.** Restore whole database backup. **2.** Recover the database to the time just before the error. **3.** Open `RESETLOGS`
TSPITR	**1.** Create auxiliary instance with RMAN or user-managed methods. **2.** Recover the tablespace on the auxiliary to the time just before the error. **3.** Import data back into the primary database.

Instance Failure

Instance failure occurs when an instance abnormally terminates. An instance failure can occur because:

- A power outage causes the server to crash.
- The server becomes unavailable because of hardware problems.
- The operating system crashes.
- One of the Oracle background processes fails.

- You issue a `SHUTDOWN ABORT` statement.

Fortunately, Oracle performs instance recovery automatically: all you need to do is restart the database. Oracle automatically detects that the database was not shut down cleanly, then applies committed and uncommitted redo records in the redo log to the datafiles and rolls back uncommitted data. Finally, Oracle synchronizes the datafiles and control file and opens the database.

Glossary

advancing the checkpoint

The action that occurs when the redo log entry marking the checkpoint changes. For example, the CKPT process may record one record as the checkpoint, then three seconds later record a later log entry as the checkpoint. This action moves the checkpoint forward by saving all changes to the datafiles before the SCN reflected by the new checkpoint. Advancing the checkpoint reduces the amount of data that potentially requires recovery.

See Also: thread checkpoint, redo record

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 archive log destinations. This copy is the archived redo log.

ARCHIVELOG mode

The mode of the database in which Oracle copies filled online redo logs to disk. Specify the mode at database creation or by using the ALTER DATABASE statement. You can enable automatic archiving either dynamically using the ALTER SYSTEM statement or by setting the initialization parameter LOG_ARCHIVE_START to TRUE.

Running the database in ARCHIVELOG mode has several advantages over NOARCHIVELOG mode. You can:

- Back up the database while it is open and being accessed by users.
- Recover the database to any desired point in time.

To protect the ARCHIVELOG mode database in case of failure, back up the archived logs.

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.

ATL (automated tape library)

A unit that contains one or more tape drives, a robotic arm, and a shelf of tapes. The ATL, also called a **tape silo**, is able to load and unload tapes into the tape drive from the shelf without operator intervention. More sophisticated tape libraries are able to identify each tape; for example, the robotic arm can use a bar-code reader to scan each tape's barcode and identify it.

See Also: media manager

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**. Unlike in **manual undo management mode**, 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 using 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: TSPITR, 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 clone database for the TSPITR set to be successful. These auxiliary files include:

- Backup control file
- `SYSTEM` tablespace
- Any datafiles containing rollback segments
- Temporary tablespace (optional)

See Also: auxiliary database, recovery set, TSPITR

backup

(1) A copy of data, that is, a database, tablespace, table, datafile, control file, or archived redo log. You can make a backup by:

- Making a copy of one or more tables with the Export utility
- Using Recovery Manager to back up one or more datafiles, control files, or archived redo logs
- Making a copy either to disk or to tape using operating system utilities (such as `cp`, `tar`, `dd`)

(2) An RMAN command that creates a backup set.

See Also: copy, backup set, multiplexing, RMAN

backup, closed

See **closed backup**

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 performing maintenance on backups as well as keeping records.

backup control file

A backup of the control file. Make the backup by:

- Using the Recovery Manager `BACKUP` or `COPY` command. Never create a backup control file by using operating system commands
- Using the SQL command `ALTER DATABASE BACKUP CONTROLFILE TO 'filename'`

Typically, you restore backup control files when all copies of the current control file are damaged; sometimes you restore them before performing certain types of point-in-time recovery.

See Also: control file

backup mode

The database mode (also called **hot backup mode**) initiated when you issue the `ALTER TABLESPACE ... 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 an operating system backup of one or more datafiles in an online tablespace. Recovery Manager 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, fractured 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 redundancy

The number of backups of a given file.

See Also: retention policy, obsolete backups and copies

backup retention policy

See **retention policy**

backup set

A backup of one or more datafiles, control files, or archived logs 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, compression, multiplexing, RMAN

block media recovery

The recovery of specified blocks within a datafile by using the Recovery Manager `BLOCKRECOVER` command. Block media recovery leaves the affected datafiles online and restores and recovers only the damaged or corrupted blocks.

breaking a mirror

The termination of a disk mirroring procedure so that a mirror image contains a static copy of the current data. The broken mirror is no longer kept up-do-date. You can create operating system database backups by placing the tablespaces in the database in **backup mode** and then breaking the mirror. After taking the tablespaces out of backup mode, back up the broken mirror side to tape. After the backup is complete, you can **resilver** the mirror.

See Also: resilvering a mirror

buffer cache

The portion of the SGA that holds copies of Oracle data blocks. All user processes concurrently connected to the instance share access to the buffer cache.

The buffers in the cache are organized in two lists: the dirty list and the least recently used (LRU) list. The dirty list holds dirty buffers, which contain data that has been modified but has not yet been written to disk. The least recently used (LRU) list holds free buffers (unmodified and available), pinned buffers (currently being accessed), and dirty buffers that have not yet been moved to the dirty list.

See Also: SGA (System Global Area)

cache recovery

There is a delay between when a data block is changed in the **buffer cache** and when it is saved to the datafiles on disk. Hence, an Oracle instance may crash before a data block has been saved. To prevent loss of these changes, Oracle writes the change to the data block (and the change to the rollback or undo blocks) to the online redo log *before* making the change to the block in the cache.

In cache recovery, Oracle rolls forward to recover data that has not been recorded in the datafiles. Unlike media recovery, crash recovery never needs to read the contents of any archived logs: all of the changes needed are in the online redo logs.

See Also: **crash recovery**, **instance recovery**, **media recovery**

cancel-based recovery

A type of incomplete media recovery in which you use the `RECOVER` command with the `UNTIL CANCEL` clause. Recovery proceeds until you issue the `CANCEL` command.

See Also: **incomplete recovery**, **media recovery**

change vector

The description of a change, usually to a single Oracle block. A set of change vectors form the content of a redo record. A change vector is the smallest unit of change recorded in the redo log.

See Also: **redo record**

change-based recovery

A type of incomplete media recovery that recovers up to a specified SCN. You can also perform cancel-based recovery, which recovers until you issue the `CANCEL` command, and time-based recovery, which recovers to a specified time.

See Also: **cancel-based recovery**, **incomplete recovery**, **media recovery**, **system change number (SCN)**, **time-based recovery**

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 determines whether the Oracle server process will attempt to read or write and whether it will work through a third-party media manager. If the channel is of type:

- `DISK`, the server process reads backups from or write backups to disk
- `sbt`, the server process reads backups from or write backups to a third-party media manager

Channels are always able to read and write datafiles to and from disk, no matter what their type.

See Also: **channel limits**, **media manager**, **target database**

channel limits

RMAN parameters that allow you to control I/O for backups and copies created by a **channel**.

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. The three types of checkpoints are: **thread checkpoint**, **database checkpoint**, and **datafile checkpoint**.

checksum

A numeric value that is mathematically derived from the contents of an Oracle data block. The checksum allows Oracle to validate the consistency of the block.

See Also: **data block**

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 long a given record must be kept before it can be overwritten.

See Also: **noncircular reuse records**

clean 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.

closed backup

A backup of one or more database files taken while the database is closed. Typically, closed backups are also whole database backups. If you closed the database cleanly, then all the files in the backup are consistent. If you shut down the database using a `SHUTDOWN ABORT` or the instance terminated abnormally, then the backups are inconsistent.

See Also: **clean shutdown**, **consistent backup**

closed database

A database that is not available to users for queries and updates. When the database is closed you can start the instance and optionally mount the database.

See Also: open database

cluster

Multiple nodes, each of which is capable of accessing data on a set of shared disks. In a **cold failover cluster**, the database instance is active on only one node. If the instance on the active node crashes, a script can automatically start an instance on the passive node and recover the database. In an Oracle Real Applications Cluster configuration, each node has an active instance against the same database and can perform I/O on a shared disk at the same time.

cold backup

See **closed backup**

command file

A file containing a sequence of RMAN commands that you can run from the command line. The contents of the command file should be identical to commands entered at the command line.

complete recovery

Recovery of one or more datafiles that applies all online and archived redo generated after the restored backup. Typically, you perform complete media 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. If you use RMAN, you can also apply incremental backups during complete recovery.

See Also: incomplete recovery, media recovery

compression

The process of copying 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; it follows that RMAN does not write never-used blocks into backup sets.

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 datafiles in this backup for it to be consistent. All datafiles in a consistent backup must:

- Have the same checkpoint **system change number (SCN)** in their headers, unless they are datafiles in tablespaces that are read-only or offline normal (in which case they will have a clean SCN that is earlier than the checkpoint SCN)
- Contain no changes past the checkpoint SCN, that is, are not fuzzy
- Match the datafile checkpoint information stored in the control file

You can only take consistent backups after you have made a **clean shutdown** of the database. The database must not be opened until the backup has completed.

See Also: **fuzzy file**, **inconsistent backup**

control file

A binary file associated with a database that maintains the physical structure and time stamps of all files in that database. Oracle updates the control file continuously during database use and must have it available for writing whenever the database is mounted or open.

See Also: **backup control file**, **current control file**

control file autobackup

The automatic backup of the current control file that RMAN makes in the situations:

- After every `BACKUP` or `COPY` command run at the RMAN prompt
- After every `BACKUP` or `COPY` command within a `RUN` block that is not followed by another `BACKUP` or `COPY` 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

(1) To replicate data. You make copies of Oracle datafiles, control files, and archived redo logs in two ways:

- Using operating system utilities (for example, the UNIX `cp` or `dd`)
- Using the Recovery Manager `COPY` command

(2) A Recovery Manager command that makes a replica of a database's datafiles, control file, or archived redo logs. This replica is made by an Oracle server process, allocated to a Recovery Manager channel, which reads the Oracle file and writes a

replica out to disk. Recovery Manager can copy the files of an open database without putting the tablespaces into **backup mode**.

See Also: **backup**

current datafile

In RMAN, the datafile in the target database pointed to by the control file. You can make a datafile copy current again by executing a `SWITCH` command.

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 one of two types:

- Logically corrupt. For example, the block was corrupted by an Oracle internal error but does not appear to be media corrupt.
- Media corrupt, that is, the block format is not correct. The block may have:
  - An incorrect checksum
  - A wrong data block address
  - An impossible block type

You can only repair a media corrupt block by:

- Replacing the block and initiating recovery. Replace the block by restoring the datafile, or recovering the individual block using the `BLOCKRECOVER` command.
- Renewing the block. Renew a block by dropping the table (or other 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**, **data block**, **fractured block**

corrupt datafile

A datafile that contains one or more corrupt blocks.

See Also: **corrupt block**

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.

In crash recovery, an instance automatically recovers the database before opening it. In general, the first instance to open the database after a crash or `SHUTDOWN ABORT` automatically performs crash recovery.

See Also: **recovery**, **redo record**

crosscheck

A check to determine whether files on disk or in the media management catalog correspond to the information in the **recovery catalog** (if used) 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 recovery catalog and control file can contain outdated information about backups and image copies.

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 *n*-1 or lower. For example, in a cumulative level 2 backup, RMAN determines which level 1 or level 0 backup is most recent and then backs up all blocks changed since that backup. When recovering with cumulative incremental backups, only one backup at each level needs to be applied.

See Also: **data block**, **differential incremental backup**, **multilevel incremental backups**

current control file

The **control file** on disk; it is the most recently modified control file for the current incarnation of the database. For a control file to be considered current during recovery, it must not have been restored from 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.

Every database must contain at least two online redo log files. If you are **multiplexing** the online redo logs, LGWR concurrently writes the same redo data to multiple files. The individual files are called **members** of an **online redo log group**.

See Also: redo log, **redo log buffer**, **redo log groups**

data block

The smallest unit of data in an Oracle database. Every database has a default block size, although data blocks in different tablespaces can have different sizes.

See Also: corrupt block, **data block address (DBA)**

data block address (DBA)

The location identifier of an Oracle **data block**. A data block address is constituted by a datafile number and a data block number. You can specify the `DBA` of a data block in the `BLOCKRECOVER` command.

data block number

The number that identifies a specific **data block** within a datafile. Blocks in a datafile are numbered sequentially. You can specify the data block number within the `BLOCKRECOVER` command.

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

A datafile is a physical operating system file on disk that was created by Oracle and contains data structures such as tables and indexes. A datafile can only belong to one database.

See Also: inaccessible datafile

datafile copy

A copy of a datafile on disk produced by either:

- The Recovery Manager COPY command
- An operating system utility

See Also: backup, copy

datafile checkpoint

The **checkpoint** structure stored in the header of each datafile. All redo in all threads prior to the datafile checkpoint SCN is guaranteed to have been saved to the datafile.

datafile header

See **file header**

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.

default file system directory

When using the Oracle Managed Files feature, the default directory specifies where Oracle creates files when no file specification has been given in the creation operation. You define the location through the initialization parameter DB_CREATE_FILE_DEST.

differential incremental backup

A type of **incremental backup** that backs up all blocks that have changed since the most recent backup at level *n* or lower. For example, in a differential level 2 backup RMAN determines which level 2, level 1, or level 0 backup is most recent and then backs up all blocks changed since that backup. Differential backups, also called **non-cumulative incremental backups**, are the default type of incremental backup. When recovering using differential incremental backups, one or more backups at each level must be applied.

See Also: cumulative incremental backup, multilevel incremental backups

dirty buffer

A buffer in the database **buffer cache** that contains a change that has not yet been written to the datafiles.

disk controller

A hardware component that is responsible for controlling one or more disk drives. The term applies to controllers integrated with the disk drive they control, as well as to high performance disk array controllers supporting various RAID configurations.

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 using the Export utility. You can then use the Import utility to import the data into a database.

See Also: full export, logical backups

file header

The first block of an Oracle datafile. The file header contains bookkeeping information related to the file, including the checkpoint SCN. Oracle requires media recovery when the checkpoint SCN in the datafile header does not match the file header information stored in the control file.

See Also: thread checkpoint

file manager

A software package that manipulates file systems.

See Also: file system

file system

A file system is a data structure built inside a contiguous disk address space. One computer can have multiple file systems, each independent of the others.

The file system allows a hard disk to contain files. Each file on the file system is distinguished by a unique filename. A file system is commonly built on top of a logical volume constructed by a **logical volume manager (LVM)**.

fractured block

A type of media corruption that can occur when database writer is writing a block at the same time an operating system utility is reading the block for backup. The block that the operating system reads can be **split**, that is, the top of the block is written at one point in time while the bottom of the block is written at another point in time. If you restore a file containing a fractured block and Oracle reads the block, then the block is considered a **corrupt block**.

The potential for fractured blocks necessitates putting tablespaces in **backup mode** before user-managed online backups. A database in backup mode writes whole Oracle data blocks to the redo log, so that if a block is split during the backup, you can repair it by using redo. Recovery Manager does not experience this problem because the server process performing the backup or copy reads each block to determine whether it is split and re-reads the block until it gets a consistent version.

See Also: **corrupt datafile**

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.

The only difference between a full backup and an incremental level 0 backup is that the full backup will not affect the number of blocks backed up by any subsequent **incremental backup**.

full export

An **export** of the whole database.

full resynchronization

A Recovery Manager operation that updates the **recovery catalog** with all changed information in the database's **control file**. You can initiate a full catalog **resynchronization** by issuing the RMAN command `RESYNC CATALOG`. Recovery Manager initiates resynchronization operations 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.

See Also: thread checkpoint

hot backup

See **online backup**

hot backup mode

See **backup mode**

image copy

A **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 is in an RMAN-specific format)
- Generated using the RMAN `COPY` command or an operating system command such as the UNIX `cp`

inaccessible datafile

A **datafile** that Oracle is attempting to read, but cannot find. Attempts to access an inaccessible file result in errors. Typically, a file is inaccessible because the media on which it is stored is faulty or the file has been moved or deleted.

See Also: media failure

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 buffer**, **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. Make a whole database backup of all files that are not offline-clean or read-only after opening with the `RESETLOGS` option. Note that if you run the RMAN command `ALTER DATABASE OPEN RESETLOGS`, RMAN automatically resets the database incarnation.

incomplete recovery

The recovery of a database in which you do not apply all of the changes generated since you created the restored backup.

Incomplete recovery is usually performed when:

- The online logs are lost due to hardware failure. In this case, you recover the database until the last archived log generated before the failure.
- A user error necessitates recovery up until just before the error occurred.

  The requirement is to recover up until some point in time before an incorrect action occurred in the database. For example, a user mistakenly deletes payroll transactions before the transactions are sent to the payroll agency. In this example, the DBA will need to restore the whole database and then perform incomplete recovery up until the point just before the user deleted the transactions.
- An archived redo log required for recovery is missing

  An archived redo log which is needed for complete recovery was not backed up, or the archived redo log contents are corrupt. In this case, you only option is to recover up to the missing log.

In each case, open the database with the `RESETLOGS` option after performing media recovery.

See Also: complete recovery, **media recovery**, **recovery**, **redo record**

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 open database backups; that is, the database is open while the files are being backed up. You can also make an inconsistent backup by backing up datafiles while a database is closed, either:

- Immediately after an Oracle instance crashed (or all instances in an Oracle Real Application Clusters configuration)
- After shutting down the database using `SHUTDOWN ABORT`

Note that 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 levels greater than 0 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**.

instance

An **SGA (System Global Area)**, Oracle code, and background processes. Create an instance by issuing any of the following commands:

STARTUP NOMOUNT	The instance starts, but does not mount the control file or open the database.
STARTUP MOUNT	The instance mounts the control file but does not open the database.
STARTUP	The instance starts, mounts the control file, and opens the database.

An instance is terminated by executing a SHUTDOWN statement.

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 cleanly (that is, with a SHUTDOWN, SHUTDOWN IMMEDIATE, or SHUTDOWN TRANSACTIONAL statement). Crash or instance recovery is always required after an instance failure.

instance recovery

In an Oracle Real Applications Cluster configuration, the application of redo data to an open database by an instance when this instance discovers that another instance has crashed. A surviving instance automatically uses the redo log to recover the data in the instance's **buffer cache**. Oracle undoes any uncommitted transactions that were in progress on the failed instance when it crashed and then clears any locks held by the crashed instance after recovery is complete.

See Also: recovery, **redo record**

job

The contents of an RMAN `RUN` command.

See Also: job commands

job commands

RMAN commands such as `BACKUP`, `COPY`, and `RECOVER` that you can execute at the RMAN prompt or within the brackets of a `RUN` command.

See Also: standalone commands

LogMiner

A utility that allows log files to be read, analyzed, and interpreted by means of SQL statements. LogMiner can view any valid online or archived redo log from Oracle8 and higher databases. You can use LogMiner to do the following:

- Track specific sets of changes based on transaction, user, table, time, and so forth. For example, you can determine who modified a database object and the before image and after image of the data.
- Pinpoint when an incorrect modification was introduced into the database. Hence, you can perform logical recovery at the application level rather than media recovery at the database level.
- Provide supplemental information for tuning and capacity planning. You can also perform various forms of historical analysis to determine trends and data access patterns.
- Retrieve critical information for debugging complex applications.

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. For example, if you create a database with two online log files, then the first file is assigned log sequence number 1. When the first file fills and Oracle switches to the second file, it assigns log sequence number 2; when it switches back to the first file, it assigns log sequence number 3, and so forth.

See Also: log switch, redo log

log sequence recovery

For RMAN, a type of incomplete recovery that recovers up to a specified log sequence number.

See Also: incomplete recovery

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.

If you run the database in `ARCHIVELOG` mode, Oracle archives the redo data in inactive log files into archived redo logs. When a log switch occurs and LGWR begins overwriting the old redo data, you are protected against data loss because the archived redo log contains the old data. If you run in `NOARCHIVELOG` mode, Oracle overwrites old redo data at a log switch without archiving it. Hence, you lose all old redo data.

See Also: redo log

logical backups

Backups in which the Export utility uses SQL to read database data and then export it into a binary file at the operating system level. You can then import the data back into a database using the Import utility.

Backups taken with the Export utility differ in the following ways from RMAN backups:

- Database logical objects are exported independently of the files that contain those objects.
- Logical backups can be imported into a different database, even on a different platform. RMAN backups are not portable between databases or platforms.

See Also: physical backups

logical volume manager (LVM)

A software program that allows sections of multiple physical disks to be combined into a single contiguous address space. This space appears as one disk to higher layers of software.

long-term backup

A backup that you want to exclude from an expiration 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. For example, you may want to survey employee salaries in past years.

managed recovery mode

A mode of a **standby database** in which the standby waits for archived log files from a target database and then automatically applies the redo logs once the files become available. This feature eliminates the need for you to interactively provide the recovery process with filenames of the archived redo logs.

manual undo management mode

A mode of the database in which undo blocks are stored in user-managed rollback segments. In **automatic undo management mode**, undo blocks are stored in a system-managed, dedicated undo tablespaces.

Mean Time To Recover (MTTR)

The desired time required to perform instance or media recovery on the database. For example, you may set 10 minutes as the goal for media recovery from a disk failure. A variety of factors influence MTTR for media recovery, including the speed of detection, the type of 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. A common example is a disk head crash that causes the loss of all data on a disk drive. 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 the data in the **buffer cache** of the SGA 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 management interface

An Oracle published API to which media management vendors have written compatible software libraries. This software integrates with Oracle so that an Oracle server process is able to issue commands to the **media manager** to write backup files to sequential storage, and read files from sequential storage. When Oracle issues a request to backup or restore a file, the media manager handles the actions required to load, label, and unload the correct tape.

The media management interface is also called the **media management layer**, the **media management library (MML)**, and the **SBT interface**.

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:

- The whole database
- A tablespace
- A datafile
- A set of blocks within a datafile

If you use all redo data, you perform complete recovery; if you use only part of the redo data, you perform incomplete recovery. Typically, you perform media recovery after a **media failure** damages some or all of the database files (datafiles, control files, or online redo logs).

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 data.

See Also: **block media recovery**, **recovery**, **redo record**

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. For example, you can mirror a datafile so that Oracle writes the same information to two different disk drives. The operation of **breaking a mirror** splits off the copy and makes a backup, while **resilvering a mirror** rejoins the split copy.

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.

mounted database

An **instance** that is started and has the control file associated with the database open. You can mount a database without opening it; typically, you put the database in this state for maintenance or for restore and recovery operations.

multilevel incremental backups

RMAN-generated incremental backups that allow you to conserve space by planning which blocks to back up and when. A level 0 **incremental backup**, which is the base for subsequent incremental backups, copies all blocks containing data. When you generate a level *n* incremental backup in which *n* is greater than `0`, you back up either:

- All blocks that have changed since the most recent backup at level *n* or lower. This is the default type of incremental backup, called a **differential incremental backup**.
- All blocks used since the most recent backup at level *n*`-1` or lower. This type of backup is called a **cumulative incremental backup**.

You can create a backup strategy in which you generate a backup at a different level each day, thereby controlling how much data you back up.

multiplexing

- **online redo logs**

  The automated maintenance of more than one identical copy of the online redo log. To multiplex the online logs, create multiple members in each redo log group. The degree of multiplexing is directly related to the number of members in each group.

- **control file**

  The automated maintenance of more than one identical copy of a database's control file. To multiplex the control file, create multiple entries in the `CONTROL_FILES` initialization parameter.

- **backup set**

  The RMAN technique of reading database files *simultaneously* from the disks and then writing the blocks to the *same* backup piece. The degree of multiplexing is smaller of these two parameter settings: `FILESPERSET` (on `BACKUP` command) and `MAXOPENFILES` (on `ALLOCATE CHANNEL` or `CONFIGURE CHANNEL`). The degree of multiplexing can be smaller than number of files included in the backup piece. For example, if RMAN writes ten datafiles into a backup set but reads from only one at a time (because `MAXOPENFILES = 1`), then blocks from ten datafiles are included in the backup piece but the degree of multiplexing is one.

- **archived redo logs**

  The Oracle archiver process is able to archive multiple copies of a redo log. You can multiplex archived redo logs by setting `LOG_ARCHIVE_DEST_`*n* (where *n* is an integer) in the initialization parameter file.

See Also: **mirroring**

multiple archiver processing

Using multiple archiver processes (ARC*n*) to archive online redo logs to one or more locations. Multiple archiver processing prevents the bottleneck that occurs when LGWR writes to the online redo log faster than a single archive process can write to the archive destinations. You can enable this feature at startup or at runtime by setting the initialization parameter `LOG_ARCHIVE_MAX_PROCESS` = *n*, where *n* is any integer from 1 to 10.

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 by using the `ALTER DATABASE` command. Oracle does not recommend running in `NOARCHIVELOG` mode because it severely limits the possibilities for recovery of lost data.

See Also: **archived redo log**, **ARCHIVELOG mode**

newed block

During block media recovery, redo records are applied serially to a block. When a block is newed all previous redo for the block becomes irrelevant because the redo applies to a defunct incarnation of the block. Because blocks can be newed, block media recovery can sometimes complete successfully even if redo records are missing.

See Also: **corrupt block**

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
- Online redo logs
- Redo threads

See Also: **circular reuse records**

normal archiving transmission

The transmittal of archived redo log files to a local disk.

See Also: **standby transmission**

obsolete backups and copies

An RMAN backup or image copy is obsolete when it is no longer needed for media recovery, for example, when multiple more recent backups and copies exist. A **retention policy** determines when a backup or copy is obsolete.

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 clean

When a tablespace is taken offline clean, 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 clean, then its datafiles must be recovered before being brought online.

offline-end checkpoint

The SCN that specifies when a datafile was brought online after being offline, or made read / write after being read-only. This SCN is stored in the control file and is the last SCN in the **offline range**. The offline-end checkpoint is important because it indicates that changes after this SCN are required to recover the datafile.

offline range

The span between the **offline-start SCN** and **offline-end checkpoint** fields of the record for a datafile in the control file. The offline range specifies a period during which there is guaranteed to be no redo for the datafile, because during this range the datafile was offline-normal or read-only. Thus, media recovery can skip this log range when recovering the datafile.

offline-start SCN

The SCN that specifies when a datafile was taken offline cleanly or made read-only. This SCN is stored in the control file and is the first SCN in the **offline range**. The offline-start SCN is important for recovery because it indicates that no changes made between this SCN and the **offline-end checkpoint** are required to recover the datafile.

offline tablespace

A tablespace that is not available to users when the database is open. You can only take a tablespace offline while the database is open. If a tablespace is taken offline, all online datafiles contained in the tablespace are taken offline.

You can take a tablespace offline using the `ALTER TABLESPACE OFFLINE` statement with three different options:

- `NORMAL`

  All the files in the tablespace are checkpointed, then taken offline. If any datafile belonging to the tablespace is not available, the tablespace cannot be taken offline normal. Datafiles in a tablespace taken offline cleanly do not need to be recovered before the tablespace is brought back online.

- `TEMPORARY`

  All datafiles in the tablespace that are accessible to Oracle are checkpointed, then taken offline. Files that were checkpointed by the `OFFLINE TEMPORARY` command do not need recovery. Datafiles that were not checkpointed because they were not accessible at the time of an `OFFLINE IMMEDIATE` command must be recovered before the tablespace is brought back online.

- `IMMEDIATE`

  All files in the tablespace are taken offline without any attempt to checkpoint the files first. All files in the tablespace must be recovered before the tablespace is brought online.

See Also: **offline datafile**

offline datafile

A datafile that is not available to users when the database is open. In exceptional circumstances, Oracle will automatically take a datafile offline if required. This file will need recovery before it can be brought online.

You can take a datafile offline either:

- As a consequence of an `ALTER TABLESPACE OFFLINE` operation.
- By issuing the statement `ALTER DATABASE DATAFILE ... OFFLINE` when the database is mounted or open.

If you take an individual datafile offline, then you must recover it before bringing it back online.

See Also: **offline tablespace**

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. When you make an RMAN backup while the database is open, however, you do not need to put the tablespaces in backup mode.

online datafile

A datafile that users can access. The database can be open or mounted when you issue the command `ALTER DATABASE DATAFILE ... ONLINE`. If the database is open, the datafile must be consistent with the rest of the database before you can bring it online. If the database is mounted, then you can bring the datafile online without being consistent with the other datafiles, but it will require recovery before the database is opened.

See Also: **online tablespace**

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.

online tablespace

A tablespace that is available to users while the database is open. You can make a tablespace available for access by users by issuing the command `ALTER TABLESPACE ... ONLINE`. The database must be open to alter a tablespace online, and all files in the tablespace must be consistent with the rest of the database before the tablespace can be made online.

See Also: **online datafile**

open database

A database that is available to users to query and update. The database is opened either automatically through a `STARTUP` statement or explicitly through an `ALTER DATABASE OPEN` statement.

operating system backup

See **user-managed backups**

operating system backup and recovery

See **user-managed backup and recovery**

Oracle managed file (OMF)

A file that is created automatically by the Oracle database server when it is needed and automatically deleted when it is no longer needed.

orphaned backups

Backups and copies 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 by specifying the `RECOVERY_PARALLELISM` initialization parameter or options to the SQL / SQL*Plus `RECOVER` command. Oracle uses one process to read the log files sequentially and dispatch redo information to several recovery processes, which apply the changes from the log files to the datafiles.

See Also: **serial recovery**

parallelization

Allocating multiple channels for Recovery Manager backup and recovery operations. You can parallelize:

- Backup set creation by allocating multiple channels before issuing a `BACKUP` command
- File copy creation by allocating multiple channels and including multiple files to be copied within a single `COPY` command
- Restore operations, with the degree of parallelism depending on the number of channels allocated as well as the distinct number of backup sets or file copies that must be read during the restore operation
- Recovery operations when applying incremental backups, with the degree of parallelism depending on the number of channels allocated and also the distinct number of backup sets that are available to read from

partial resynchronization

A type of **resynchronization** in which RMAN transfers information about archived redo logs, backup sets and datafile copies from the target database control file to the **recovery catalog**. Partial resynchronization does not transfer information such as:

- New datafiles
- New or removed tablespaces

- New or removed online log groups and members

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 *Oracle9i Database Administrator's Guide*.

physical backups

Physical database files that have been copied from one place to another. The files can be datafiles, archived redo logs, or control files. You can make physical backups using Recovery Manager or with operating system commands such as the UNIX `cp`.

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.

A full **resynchronization** of the recovery catalog updates all changed RMAN metadata in the repository, including physical schema information. If the database is open, RMAN also gathers information about rollback segments. A partial resynchronization of the recovery catalog does not update physical schema or rollback information.

pluggable tablespace

See **transportable tablespace**

point of recoverability

The earliest time within a recovery window. A **retention policy** that specifies a recovery window mandates that the database must be able to be recovered to the earliest time in the window. For example, in a recovery window of 7 days, the point of recoverability is always 7 days before the current time.

See Also: obsolete backups and copies

proxy copy

The functionality that enables a **media manager** to take over the transfer of data between the media storage device and disk during RMAN backup and restore operations.

read errors

Errors that occur when Oracle is unable to read a datafile, control file, or online redo log. Oracle returns an error to the operating system and to the application, along with an Oracle error indicating that the file cannot be found, cannot be opened, or cannot be read. Note that Oracle does not automatically take a datafile offline if it is unable to read it.

read-only database

A database opened with the `ALTER DATABASE OPEN READ ONLY` command. As their name suggests, read-only databases are for queries only and cannot be modified. Oracle allows a standby database to be run in read-only mode, which means that it can be queried while still serving as an up-to-date emergency replacement for the primary database.

read-only tablespace

A tablespace whose status has been changed to prevent it from being updated. You put in read-only mode by executing the SQL statement `ALTER TABLESPACE ... READ ONLY`. Typically, you put a tablespace in read-only mode to reduce the frequency with which it is backed up. For example, instead of backing up the tablespace nightly, you reduce the backup frequency to once a month.

Note: The longer the duration between backups of a tablespace, the longer you will need to retain the backup media and the larger the risk of failed backup media (as you will have backed it up fewer times).

recover

(1) A Recovery Manager command that updates a restored datafile by the application of incremental backups (if they exist) and then by the application of archived or online redo logs.

(2) A SQL*Plus command that updates a restored file by the application of archived or online redo logs.

See Also: **recovery**

recovery

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. Only Recovery Manager can recover datafiles by applying incremental backups.

See Also: **complete recovery**, **incomplete recovery**

recovery catalog

A set of Oracle tables and views used by Recovery Manager to store information about Oracle databases. Recovery Manager uses this data to manage the backup, restore, and recovery of Oracle databases. The recovery catalog is optional. If you choose not to use a recovery catalog, RMAN uses the target database 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 Manager environment

A environment containing the **Recovery Manager (RMAN)** executable and the various computers, databases, applications, and APIs with which it interacts.

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.

For example, if you implement a recovery window of one week, then this window of time must extend back exactly seven days from the present so that you can restore a backup and recover it to any point within the last week.

The recovery window always keeps pace with the current time. For example, if the current day is January 14 and the recovery window is 7 days, then the recovery window stretches between January 7 and January 14. When the current day is January 28, then the recovery window stretches between January 21 and January 28.

Any backups or logs not needed for recovery are considered obsolete. If you make periodic backups, then as the recovery window moves forward in time the older backups become obsolete.

See Also: obsolete backups and copies, point of recoverability, 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 copied to an offline destination. If the database is in `ARCHIVELOG` mode and automatic archiving is enabled, then the archiver process or processes copy each online redo log to one or more archive log destinations after it is filled.

See Also: archived redo log, online redo log, redo record

redo log buffer

The memory buffer in the system global area (SGA) in which Oracle logs redo records. The background process LGWR writes the buffers into the current online redo log.

See Also: redo record

redo log files

Redo log files are the operating system files that log writer writes its changes. A redo log member within a redo log group corresponds to one and only one redo log file. The `V$LOGFILE` view displays the filenames of redo log files.

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 record

A group of change vectors describing a single, atomic change to the database. Oracle constructs redo records for all data block changes and saves them on disk in the current online redo log. Redo records allow changes to database blocks to be reconstructed should data loss occur.

See Also: **redo log**

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. If you run in an Oracle Real Application Clusters configuration, then you have multiple redo threads, one for each instance.

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 collection of 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: **control file**, **recovery catalog**

RESETLOGS option

A method for opening a database that results in a new database incarnation, the resetting of the log sequence number to 1, and the re-formatting or re-creation of the online redo logs. A database must be opened with the `RESETLOGS` keyword after:

- Incomplete recovery
- Recovery using a backup control file

RESETLOGS SCN and time stamp

Together with the RESETLOGS time stamp, the RESETLOGS SCN uniquely identifies each execution of an ALTER DATABASE OPEN RESETLOGS statement. When the online logs are reset, Oracle creates a new and unique **incarnation** of the database.

The RESETLOGS SCN and time stamp are stored in the control file, in each datafile header, and in each redo log file header. An online or archived redo log cannot be applied by recovery if its RESETLOGS data does not match the database information in the control file.

Except for special circumstances (for example, offline normal or read-only tablespaces), a datafile cannot be recovered or accessed if its RESETLOGS SCN and time stamp do not match the database information in the control file. This precaution ensures that changes discarded by RESETLOGS cannot get reapplied to the database.

resilvering a mirror

Configuring the operating system or hardware managing the mirror so that you refresh a broken mirror from the half that is up-to-date and then maintain both sides of the mirror.

See Also: breaking a mirror, **mirroring**

restore

The replacement of a lost or damaged file with a backup. You can restore files either with operating system commands such as UNIX `cp` or the RMAN `RESTORE` command.

See Also: recover

resync

See **resynchronization**

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 copies 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.

Any backup or copy that is not needed according to the retention policy is considered obsolete. RMAN can automatically delete such obsolete backups.

See Also: obsolete backups and copies

RMAN

See **Recovery Manager (RMAN)**

rollback segments

Database segments that record the before-images of changes to the database. A rollback segment contains a transaction table with two or more extents of undo blocks. Undo blocks are arranged in a circular fashion so that older pieces of undo data are overwritten in chronological order.

rolling back

The use of rollback segments to undo uncommitted transactions applied to the database during the **rolling forward** stage of **recovery**.

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: recovery, **rolling back**

SBT

System Backup to Tape

See Also: media management interface

serial recovery

A form of recovery in which a single process applies the changes in the redo log files sequentially.

See Also: parallel recovery

SGA (System Global Area)

A group of shared memory structures that contain data and control information for one Oracle database instance. The SGA and Oracle processes constitute an Oracle instance. Oracle automatically allocates memory for an SGA whenever you start an instance and the operating system reclaims the memory when you shut down the instance. Each instance has one and only one SGA.

snapshot control file

A copy of a database's control file taken 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.

split block

See **fractured block**

staging

The process of restoring archived logs from tertiary storage to disk in order to allow recovery to proceed. RMAN stages the logs to disk when the `RECOVER` command is executed. To use this feature, you must configure a media manager.

See Also: media manager

standalone commands

RMAN commands that you do not have to execute within the brackets of a `RUN` command.

See Also: job commands

standby database

A copy of a production database that you can use for disaster protection. You can update the standby database with archived redo logs from the production database in order to keep it current. If a disaster destroys the production database, you can activate the standby database and make it the new production database.

standby transmission

The transmittal of archived redo log files over a network to either a local or remote **standby database**.

stop SCN

Each control file record for a datafile has a field called the **stop SCN**. If the datafile is offline or read-only, the stop SCN is the SCN beyond which no further redo exists for the datafile. If the datafile is online and any instance has the database open, then the stop SCN is set to infinity. Oracle uses the stop SCN during media recovery to determine when recovery for a datafile can end. This feature ensures that media recovery terminates when recovering an offline file while the database is open.

stored script

A sequence of RMAN commands stored in the **recovery catalog**.

switch

A Recovery Manager command which converts a datafile copy into a datafile used by an Oracle database. It performs the equivalent function of the SQL statement `ALTER DATABASE RENAME FILE 'original_name' TO 'new_name'`, and also marks the datafile copy as no longer available.

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

A database is divided into one or more logical storage units called tablespaces. Each tablespace has one or more physical datafiles exclusively associated with it.

See Also: datafile

tablespace point-in-time recovery (TSPITR)

The recovery of one or more non-`SYSTEM` tablespaces to a point in time that is different from the database. You can use either RMAN or user-managed methods to perform TSPITR.

tag

A user-specified character string that acts as a symbolic name for a backup set or image copy. You can specify a tag when executing the `RESTORE` or `CHANGE` command. The maximum length of a tag is 30 characters.

tail of the log

The most recent redo record in the redo log file. As users make changes to the database, the tail keeps moving forward. Since the latest checkpoint is always temporally behind the tail of the log, it is said to **lag** the tail of the log. If the checkpoint lags the tail of the log significantly, crash recovery time increases.

tape streaming

Writing output to a tape drive fast enough to keep the tape constantly busy. If the device is not kept busy, then its performance decreases because the drive mechanism must be started and stopped for each piece of data that is received.

tape drive

A piece of hardware that reads and writes magnetic tapes.

tape silo

See **ATL (automated tape library)**

tape volume

One physical piece of tape media.

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.

thread

See **redo thread**

thread checkpoint

A type of **checkpoint** stored in the **control file** indicating that all changes to online datafiles in a given thread prior to the checkpoint SCN have been saved to disk. Oracle updates the thread checkpoint every time an instance checkpoints its thread.

Checkpointing is crucial for recovery because it limits the amount of transaction redo that crash and instance recovery must potentially apply. Online switch management guarantees that the current checkpoint has moved out of an online log file before that log file eligible for reuse. Checkpointing works in conjunction with online log switch management to ensure that crash and instance recovery can be accomplished using only online redo logs.

See Also: database checkpoint, **datafile checkpoint**, **redo record**

time-based recovery

The incomplete recovery of database files to a noncurrent time. Time-based recovery is also known as **point-in-time recovery**. There are two types:

database point-in-time recovery (DBPITR)	signifies the incomplete recovery of all datafiles and control file to a time before the most recent time
tablespace point-in-time recovery (TSPITR)	signifies the incomplete recovery of all datafiles in one or more tablespaces on an auxiliary database to a specific time before the most current time. The tablespace is then re-integrated into the original database.

See Also: incomplete recovery, **media recovery**, **recovery**

transaction recovery

Transaction recovery involves rolling back all uncommitted transactions of a failed instance. These are "in-progress" transactions that did not commit and that Oracle needs to roll back. It is possible for uncommitted transactions to get saved to disk. In this case, Oracle uses undo data to reverse the effects of any changes that were written to the datafiles but not yet committed.

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. This feature is often an advantage because:

- It is faster than import/export or unload/load, since it involves only copying datafiles and integrating metadata
- You can use it to move index data, allowing you to avoid rebuilding indexes

TSPITR

See **tablespace point-in-time recovery (TSPITR)**

undo blocks

Oracle blocks that contain the before-image of a change to the database. For example, if you update a salary column in a table from 55 to 65, Oracle writes the before-image of 55 into an undo block. If you run in **manual undo management mode**, then undo blocks are stored in user-managed rollback segments. If you run in **automatic undo management mode**, then undo blocks are stored in system-managed undo tablespaces.

undo tablespace

A dedicated tablespaces that stores only undo information when the database is run in **automatic undo management mode**. An undo tablespace contains one or more **undo segments**. The creation of any other types of segment (for example, tables, indexes) in undo tablespaces is not allowed.

In the automatic mode, each Oracle instance is assigned one and only one undo tablespace. Each undo tablespace is composed of a set of undo files. Undo blocks are grouped in extents. At any point in time, an extent is either allocated to (and used by) a transaction table, or is free.

Blocks in undo tablespaces are grouped into the following categories:

- File control blocks, bitmap blocks, and so forth used for space management
- Undo segments containing transaction table blocks, undo blocks, and extent-map blocks used for transaction management
- Free blocks that are unallocated to file control or undo segments

undo segments

Segments containing all undo data when the database is run in **automatic undo management mode**. Undo segments are internally similar to the rollback segments that are used to store undo data in **manual undo management mode**.

Like a rollback segment, an automatic undo segment must have a transaction table and two or more undo extents. Oracle stores the status of transactions in the transaction tables. Automatic undo segments are internally managed and cannot be manipulated by users through an external interface.

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 backup**.

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` statement or the SQL `ALTER DATABASE RECOVER` statement.

validation

A test that checks whether a backup set or copy can be restored. RMAN scans all of the copies or backup pieces in the specified backup sets and looks at the checksum to verify that the contents can be successfully restored.

Use the `RESTORE ... VALIDATE` or `VALIDATE BACKUPSET` command when you suspect that one or more copies or backup pieces in a backup set are missing or have been damaged. Note that `RESTORE ... VALIDATE` and `VALIDATE BACKUPSET` actually test whether the files can be restored, whereas `CROSSCHECK` merely examine the file headers.

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**

write errors

Errors that occur when Oracle is unable to write data into a datafile, control file, or online redo log.

write-ahead logging

The Oracle protocol of recording changes to Oracle data blocks and undo segments in the online redo log *before* applying the changes to the actual data blocks. In other words, before database writer can write out a cache buffer containing a modified data block, log writer must write out the redo log buffer containing redo records describing both the changes to the data block as well as the undo required to roll back those changes if the transaction does not commit. This protocol ensures that changes can be recovered even if an instance failure occurs before the change can be written to the datafiles.

Index