CHAPTER 1

Backup and Recovery 101

Oracle *backup and recovery* refers to the theory and practice of protecting a real-life Oracle database against data loss and recovering data after a loss. You can lose data either because of a technical problem such as media failure (such as a disk drive breaking down) or because of errors made by the users (such as a wrong update or an overeager sysadmin or DBA deleting the wrong file). Oracle backup is the set of concepts, strategies, and steps to make copies of a database so you can use them to recover from a failure/error situation. Backups in this sense refer to physical backups of database files, control files, and archived redo log files. Oracle recovery is the set of concepts, strategies, and steps to actually recover from a system/user error or a potential data loss due to media-related problems such as the loss of a disk drive.

Ideally, we all like to never have any data loss or downtime because of a database failure. However, the constraints of both humans and machinery such as disk drive technology means that there's bound to be some type of failure during the course of your life as a practicing DBA, since you're the one in charge of maintaining and tuning databases that support the business. Here is your more realistic set of goals then:

- Protect the database from as many types of failure as possible.
- Increase the mean time between failures.
- Decrease the mean time to recover.
- Minimize the loss of data when there is a database failure.

Recovery Manager (RMAN) is Oracle's main backup and recovery tool and is a built-in component of the Oracle server. You don't have to pay additional licensing fees to use RMAN, as is the case when you use other Oracle products such as the Enterprise Manager Grid Control, for example. Since its introduction as part of the Oracle 8 release, RMAN has improved considerably to the point where it has become the most powerful tool to back up and recover Oracle databases, with its wide array of sophisticated and powerful capabilities. You can still use traditional user-managed backup and recovery techniques, but the powerful backup and recovery features offered by RMAN mean you won't be taking full advantage of your Oracle server software if you don't use RMAN. This book provides comprehensive coverage of RMAN's backup and recovery capabilities.

Before starting our discussion of how to perform backup and recovery tasks with RMAN, it's important to get an overview of key backup- and recovery-related concepts. We discuss the following topics in this chapter before turning to a detailed discussion of RMAN backup and recovery techniques starting in Chapter 2:

- Types of database failures
- Oracle backup and recovery concepts
- Backup types
- Recovery types
- An introduction to RMAN
- Backup and recovery best practices

We use the Oracle Database 11*g* release throughout this book, thus providing you with cutting-edge RMAN backup and recovery solutions. Most of what we say, however, applies equally to Oracle Database 10*g*. We specifically mention whenever we're discussing a feature not available in Oracle Database 10*g*.

Types of Database Failures

Since database backups are made to protect against a database failure, let's quickly review the types of database failures that can occur. A database can fail, either entirely or partially, because of various reasons. You can recover from some types of database failure with scarcely any effort on your part, because the Oracle database can recover automatically from some types of failures. The more critical types of failures require you to go in and "recover" the database by using your backups. You can divide database failures into the categories covered in the following sections.

Statement Failure

A typical example of a statement failure is when a program attempts to enter invalid data into an Oracle table. The statement will fail because of the checks built into the data insertion process. The solution here is to clean up the data by validating or correcting it. Sometimes, a program may fail to complete successfully because of programmatic logical errors. You must then refer the problem to the development group for corrections.

It is fairly common for a long data insertion job or a data import job to fail midway because there is no more room to put the data in. If you haven't already invoked the *resumable space allocation* feature, you must add space to the tablespace that contains the table that ran out of space. Another common cause of a statement failure is not having the proper privileges to perform a task. Your task as a DBA is to simply grant the appropriate privileges for the user who invoked the failed SQL statement.

User Process Failure

Sometimes, a user process may be terminated abruptly because of, say, the user performing an abnormal disconnect or performing a terminal program error and losing the session connection. As a DBA, there is not much you need to do here: the Oracle background processes

will roll back any uncommitted changes to the data and release the locks that were held by the abnormally disconnected user session. The user will have to reconnect after the abrupt termination.

Network Failure

A network failure can also cause a database failure. Network failures can occur because the Oracle Net listener, the network interface card (NIC), or the network connection has failed. The DBA must configure multiple network cards and a backup network connection and backup listener to protect against these errors. In addition, you can use the connect-time failover feature to protect against a network failure.

Instance Failure

You experience an Oracle instance failure when your database instance comes down because of an event such as a hardware failure, a power failure, or an emergency shutdown procedure. You may also experience an instance shutdown when the key Oracle background process such as `pmon` shuts down because of an error condition.

Following an instance failure, first you check the alert log and trace files for any potential hints about the cause of the instance failure. Following this, you can just restart the database instance by using the Oracle command `startup` from the SQL*Plus command line.

Since the database wasn't cleanly shut down and the database files aren't synchronized, Oracle will perform an automatic instance or crash recovery at this point. Oracle will automatically perform a rollback of the uncommitted transactions by using data from the undo segments and will roll forward the committed changes it finds in the online redo log files. You don't need to use any sort of backup when restarting the database instance following an instance failure. Once the uncommitted changes are backed out and the committed changes are rolled forward, the datafiles are in sync again and will contain only committed data.

User Error

Inadvertently dropping a table is every DBA's nightmare. In addition to accidentally dropping a table, users can also wrongly modify or delete data from a table. You can use techniques such as the `flashback table` feature to restore a table to a previous point in time. You can use the `flashback drop` feature to recover an accidentally dropped table. Of course, if the transaction isn't committed yet, you can simply roll back the unwanted changes. Oracle's LogMiner tool also comes in handy in situations like this.

Media Failure

Media failure occurs when you lose a disk or a disk controller fails, hindering access to your database. A head crash, file corruption, and the overwriting or deletion of a datafile are all examples of a media failure. In general, any failure to read from or write to a disk constitutes a media failure. Although the first four types of failures don't require you to resort to a backup, media failure in most cases would require performing a media recovery with the help of backups of the datafiles and archived redo logs.

Each type of media failure may have a different solution as far as recovery is concerned. For example, if a control file copy is accidentally deleted, you won't have to go to your backups. On the other hand, deleting a datafile most likely requires you to restore the datafile from

a backup as well as use the archived redo logs to bring the database up-to-date. If only a few blocks in a datafile are corrupt, you may use RMAN's block media recovery feature instead of restoring datafiles and performing media recovery.

In this book, we are mostly concerned with problems caused by media failures and how to recover from them. For this reason, let's analyze how database failures can occur because of media problems. Once your Oracle database instance is running in open mode, it could crash because of the loss of several types of files. For example, the database will crash if any of the following are true:

- Any of the multiplexed control files are deleted or lost because of a disk failure. You must restore the missing control file by copying from an existing control file and restarting the instance.
- Any datafile belonging to the system or the undo tablespace is deleted or lost because of a disk failure. If you lose one of these files, the instance may or may not shut down immediately. If the instance is still running, shut it down with the `shutdown abort` statement. You then start up the database in mount state, restore the lost datafile, and recover it before opening the database for public access.
- An entire redo log group is lost. If you have at least one member of the redo log group, your database instance can continue to operate normally. Restore the missing log file by copying one of the other members of the same group.

The database won't crash if any of the following are true:

- Any nonsystem or undo tablespace datafile is lost. If you lose a nonsystem or undo tablespace file, also known as a noncritical datafile from the point of view of the Oracle server, you must first restore and then recover that datafile. The database instance can continue operating in the meanwhile.
- At least a single member of each redo log group is available, although you might have lost other members of one or more groups.

Oracle Backup and Recovery Concepts

Before you jump into Oracle backup and recovery concepts, it's a good idea to review the basic Oracle backup and recovery architecture. Oracle uses several background processes that are part of the Oracle instance, and some of these background processes play a vital role in backup and recovery tasks. For a quick understanding of the Oracle background processes involved in backup and recovery, please see Figure 11-1 (in Chapter 11). Oracle also has several physical structures that are crucial components of backup and recovery, which we discuss in the following sections.

Backup and Recovery Instance Architecture

The Oracle instance consists of the system global area (SGA), which is the memory allocated to the Oracle instance, and a set of Oracle processes called the *background processes*. The Oracle processes start when you start the instance and keep running as long as the instance is alive. Each of the Oracle background processes is in charge of a specific activity, such as writing changed data to the datafiles, cleaning up after disconnected user sessions, and so on.

We'll briefly review the key Oracle background processes that perform critical backup and recovery–related tasks, which are the checkpoint process, the log writer process, and the archiver process.

The Checkpoint Process

The checkpoint process does three things:

- It signals the database write process (DBWn) at each checkpoint.
- It updates the datafile headers with the checkpoint information.
- It updates the control files with the checkpoint information.

The Log Writer Process

Oracle's online redo log files record all changes made to the database. Oracle uses a "write-ahead" protocol, meaning the logs are written to before the datafiles are. Therefore, it is critical to always protect the online logs against loss by ensuring they are multiplexed. Any changes made to the database are first recorded in the redo log buffer, which is part of the SGA.

Redo log files come into play when a database instance fails or crashes. Upon restart, the instance will read the redo log files looking for any committed changes that need to be applied to the datafiles. Remember, when you commit, Oracle ensures that what you are committing has first been written to the redo log files before these changes are recorded in the actual datafiles. The redo log is the ultimate source of truth for all changes to the data in an Oracle database, since an instance failure before the changes are written to the datafiles means that the changes are only in the redo log files but not in the datafiles.

The log writer (LGWR) process is responsible for transferring the contents of the redo log buffer to the online redo log files. The log writer writes to the online redo files under the following circumstances:

- At each commit
- Every three seconds
- When the redo log buffer is one-third full

The important thing to remember here is that the log writer process writes before the database writer does, because of the *write-ahead* protocol. Data changes aren't necessarily written to datafiles when you commit a transaction, but they are always written to the redo log.

■ **Note** In fact, some esoteric features in the Oracle database allow you to make changes without generating redo log entries. Such features are helpful, for example, when loading large amounts of data. However, their benefits do not come without additional risk. The important point to take away from this section is that unless you are specifically using a feature that disables logging, any changes you commit are first written to the redo log files, and it is the log writer process that does the writing.

The Archiver Process

The archiver (ARCn) is an optional background process and is in charge of archiving the filled online redo log files, before they can be overwritten by new data. The archiver background process is used only if you're running your database in archivelog mode.

Physical Database Structures Used in Recovering Data

You need to deal with four major physical database structures during a database recovery:

- Datafiles
- Redo logs (archived and online)
- Control files
- Undo records

In a basic database recovery situation, you'd need to first restore datafiles by using backups (from a past period, of course). Once the restoration of the datafiles is completed, you issue the `recover` command, which results in the database rolling forward all committed data and thus bringing the database up-to-date. The database also rolls back any uncommitted data that's recorded in the undo segments that are part of the undo tablespace. The database server automatically performs the rollback of uncommitted data by using undo records in the undo tablespace to undo all uncommitted changes that were applied to the datafiles from the redo logs during the recovery process. This rolling back of uncommitted data takes place by using the information about all the changes made since the last database start-up. Oracle records all changes made to the database in files called the *online redo log files*. Since Oracle uses a round-robin method of writing the online redo log members, it is critical that you save the filled online redo logs before they are written. The process of saving the filled redo log files is called *archiving*, and the saved redo log files are termed *archived redo log files*. A media recovery process uses both the archived redo log files and the online redo log files.

The control file is essential for the Oracle instance to function, because it contains critical information concerning tablespace and datafile records, checkpoints, redo log threads in the current online redo log, log sequence numbers, and so on.

RMAN lets you back up all the files you need for a database recovery, including datafiles, control files, and archived redo logs. RMAN also lets you make image copies of both datafiles and control files, in addition to the standard RMAN-formatted backup pieces. You should never back up online redo log files; instead, always duplex these files to protect against the loss of an online redo log.

Archivelog and Noarchivelog Mode of Operation

You can operate your Oracle database in either *archivelog* mode or *noarchivelog* mode. In noarchivelog mode, Oracle will overwrite the filled online redo logs, instead of archiving (saving) the online redo logs. In this mode, you're protected only from instance failures, such as those caused by a power failure, for example, but not from a media failure. Thus, if there is a media failure, such as a damaged disk drive, the changes that were overwritten are gone forever, and the database won't be able to access those data modifications to recover the database up to the current point in time. The transactions made since the last backup are lost forever, and you can restore the database only to the point of the last backup you made.

If you are running your database in noarchivelog mode and you happen to lose a datafile, for example, you follow these steps to get back to work again:

1. If the instance isn't already shut down, first shut it down.
2. Restore the entire database (datafiles and control files) from the backups.
3. Restart the database by using the `startup` (open mode) command.
4. Users lose any data that was changed or newly entered in the database since you took the backup that was just restored. You can enter the data if you have a source, or you're going to have a data loss situation.

If you are running a production database—or if you want to make sure that all the data changes made to any database, for that matter, are always protected—you must operate your database in archivelog mode. Only a database running in archivelog mode can recover from both instance and media failures. You can't perform a media recovery on a database running in noarchivelog mode.

If you're running the database in noarchivelog mode, remember that you can make a whole-database backup only after first shutting the database down. You can't make any online tablespace backups in such a database. A database in noarchivelog mode also can't use the tablespace point-in-time recovery technique. Make sure you take frequent whole-database backups if an important database is running in noarchivelog mode for some reason.

Flashback Technology

Traditionally, restoring backed-up datafiles and recovering the database with the help of archived redo logs was the only way you could rewind the database to a previous point in time or view older data. Oracle's *flashback* technology offers new techniques that let you recover from several types of errors without ever having to restore backup files. The key idea behind the flashback technology is to improve database availability while you're fixing logical data errors. While you're correcting the logical data errors in one or more errors, all the other database objects continue to be available to the users unhindered. Flashback technology actually consists of a half dozen specific features, most but not all of which rely on the use of undo data to undo the effect of logical errors:

Oracle flashback query (uses undo data): This feature lets you view results from a past period in time. You can choose to use this query to retrieve lost or wrongly deleted data.

Oracle flashback version query (uses undo data): This feature lets you view all versions of a table's rows during a specific interval. You can use this feature for retrieving old data as well as for auditing purposes.

Oracle flashback transaction query (uses undo data): This feature enables you to view all the changes made by one or more transactions during a specified period of time.

Oracle flashback transaction backout (uses undo data): This new Oracle Database 11*g* feature lets you back out unwanted transactions by using compensating transactions.

The Oracle flashback table (uses undo data): This feature lets you recover a table (online) to a previous point in time. You can recover a table or a set of tables to a past point in time by using the contents of the undo tablespace. The database can remain online during this

time, thus enhancing its availability. All of a table's constraints, triggers, and indexes are restored during the recovery, while the database remains online. You don't have to restore from a backup when you perform a flashback table operation. Since you're using undo data to restore the table instead of media recovery, you'll get done faster, and with less effort to boot.

Oracle flashback drop feature (uses the recycle bin): This relies on the concept of a recycle bin and lets you restore a dropped table. When you accidentally drop a table with the `drop table` statement, information about the purged table is saved in the recycle bin (which is actually a data dictionary table) under a system-assigned name. Actually, the table's contents remain intact and in place, but the data dictionary marks the table as having been dropped. You can then "undrop" the table at a later time by using the `flashback table ... to before drop` statement, which recovers the dropped object from the recycle bin. The flashback table feature relies entirely on the recycle bin concept.

A new feature of the Oracle Database 11*g* release, the *flashback data archive* lets you use the previously described flashback features to access data from a period of time that's as old as you want. By using a flashback data archive, you overcome the limitation of a short undo retention time in the undo tablespace.

The *Oracle flashback database* feature serves as an alternative to traditional database point-in-time recovery. You use this feature to undo changes made by logical data corruption or by user errors. The essential point to understand here is that the opposite of flashback is to *recover*. In normal database recovery, you update the backups by applying logs forward. In flashback, you rewind the database by applying flashback logs backward. Thus, in most cases, a flashback database operation will take much less time than the time it takes to restore and recover during the traditional alternative, which is a database point-in-time recovery. The flashback database feature takes the database back in time, essentially rewinding it to a past point in time by undoing all changes made to the database since that time. Unlike traditional point-in-time recovery, you don't have to perform a media recovery by restoring backups. You simply use the new flashback logs (stored in the flash recovery area) to access older versions of the changed data blocks. In addition, the database makes use of the archived redo logs as well.

■ **Note** The flashback database feature is useless in dealing with cases of lost datafiles or damaged media. You can use this feature to undo the changes made to an Oracle database's datafiles only by reverting the contents of the datafiles to a previous point in time.

When you enable flashback logging so that you can use the flashback database feature, you may not always be able to return to a specific point in time, if the flashback logs for that period aren't available. Oracle's *guaranteed restore points* feature lets you specify an system change number (SCN) to which you can always restore the database. That is, the database will ensure that the flashback logs from the specific SCN on are saved, no matter what. Thus, guaranteed restore points, which are an adjunct to the flashback database feature, let you ensure that you'll be at least able to recover until the specified SCN, even if you aren't necessarily able to recover up to the current SCN.

Backup Types

When we talk about a database backup, your first thought might be that it is simply a copy of all the database physical files. However, an Oracle database offers several types of backups. We summarize the main types of backups in the following sections.

Physical and Logical Backups

When you make a copy of a database file using an operating system utility such as `cp`, for example, you are making an actual physical copy of the database file. You can use this file to restore the database contents if you happen to lose the disk containing that file. Physical backups are simply physical copies of the files used by the database, such as datafiles, redo logs, and control files. However, making exact physical copies of the database file isn't the only way to copy the contents of an Oracle database. You can also make a logical backup by using Oracle's Data Pump Export tool wherein you copy the definitions and contents of all of the database's logical components such as tables and so on. You can use Oracle's Data Pump Import utility to later import the logical data into the same or another Oracle database. Logical backups are, however, not a complete backup and recovery solution; they serve as a secondary means of backing up key tablespaces or tables in some situations.

Whole and Partial Backups

A *whole-backup* of a database is the backup of the *entire* database; this is the most commonly made type of Oracle database backup. A whole-database backup includes all the datafiles plus the control files. A *partial backup* refers to backups of a tablespace or datafile in a database. A datafile backup will include only a single operating system file. A tablespace backup includes all the datafiles that are part of that tablespace. You can also back up the control file just by itself by making either a text or a binary copy of it. The control file is a crucial part of the recovery process, since it contains key information about various recovery-related structures.

Online and Offline Backups

RMAN supports both *offline* and *online* backups. An offline backup, also called a *cold backup*, is one made after shutting down the database using the `shutdown` command or the `shutdown` command with the `immediate` or `transactional` clause. An offline backup, provided you make one after the database is shut down gracefully, is always consistent, whether you're operating in archivelog or noarchivelog mode. When making an offline backup with RMAN, you must, however, start the database you want to back up in the mount mode.

An online backup, also called a *hot* or *warm* backup, is one made while the database instance is still open. By definition, an online backup is always inconsistent. During a recovery, the application of the necessary archived redo logs will make the backup consistent. Thus, you can make online backups of any database you're operating, and the resulting inconsistent backups can be made consistent with the application of archived redo logs. However, for databases running in noarchivelog mode, open inconsistent backups aren't recommended.

Full and Incremental Backups

A full backup of a database will contain complete backups of all the datafiles. Incremental backups contain only the changed data blocks in the datafiles. Obviously, then, incremental

backups can potentially take a much shorter time than full backups. You can make incremental backups only with the help of RMAN—you can't make incremental backups using user-managed backup techniques.

Consistent and Inconsistent Backups

To understand the crucial difference between consistent and inconsistent backups, you must first understand the concept of the system change number (SCN). The SCN is an Oracle server–assigned number that indicates a committed version of the database. It's quite possible that different datafiles in the database might have a different SCN at any given point in time. If the SCNs across all the datafiles are synchronized, it means that the data across the datafiles comes from a single point of time and, thus, is consistent.

During each checkpoint, the server makes all database file SCNs consistent with respect to an identical SCN. In addition, it updates the control file with that SCN information. This synchronization of the SCNs gives you a consistent backup of your database. Not only does each of the datafiles in the database have the same SCN, it must also not contain any database changes beyond that common SCN.

If you back up your database while it's running, you may end up with backups of the various datafiles at various time points and different SCNs. This means your backups are inconsistent, since the SCNs aren't identical across all the datafiles.

If you're operating the database in noarchivelog mode, you can use only consistent backups to restore your database. If you're operating in archivelog mode, however, you can use consistent or inconsistent backups to restore the database. If you're using a consistent backup, you can open a whole-database backup without recovery and without using the `open resetlogs` command. If you're using inconsistent backups, however, you must use archived redo logs to make the data current and synchronize the SCNs across the datafiles.

The key fact here is that the recovery process will make your inconsistent backups consistent again by using the data from the archived redo logs and the online redo log files to apply all the necessary changes across the datafiles to make them all consistent with reference to a single SCN.

If you're running the database in noarchivelog mode, the recommended approach to backing up the database is to shut down the database cleanly first and then to back up all the datafiles. If you're using RMAN to perform an offline backup, the database must be mounted before you can actually perform the RMAN backup. This is because RMAN needs to update the target database control file.

When you follow the approach suggested in the previous paragraph, you'll be backing up a consistent database. It's not recommended that you back up an inconsistent database resulting from an abrupt shutdown using the `shutdown abort` command, for example.

If you're running the database in archivelog mode, you can back up a whole database in any of the following ways:

- Closed and consistent
- Closed and inconsistent
- Open and inconsistent

The ability to back up a database while it is open and in use is a key benefit of running a database in archivelog mode.

Recovery Types

There are several methods of recovering data, and the particular recovery strategy you adopt will depend on your backup strategy to a large extent. For example, if you are operating in noarchivelog mode, then in most cases you can't go perform a complete recovery. You can restore only the latest backup and will lose all the data that was entered since the time of the backup. In the following sections, we'll briefly describe the major recovery techniques you can use. Similarly, the flashback database technique offers a much faster means of restoring a database to a previous point in time than traditional media recovery, but of course, you can't avail yourself of this wonderful feature if you haven't configured and used a flashback recovery area (to store the flashback logs).

Database Recovery and Consistent vs. Inconsistent Backups

If you shut down your database using either `shutdown normal` (same as the `shutdown` command), `shutdown immediate`, or `shutdown transactional`, you'll have a *consistent* database. A shutdown following each of the previously mentioned variations of the `shutdown` command will result in the following actions:

- All uncommitted changes are rolled back first.
- The contents of the database buffer cache are written to the datafiles on disk.
- All resources such as locks and latches are released.

Since the database was cleanly shut down, when you restart the database, there is no need for an instance recovery, which is the main implication of performing and using a consistent backup.

If you shut down your database using either the `shutdown abort` or `shutdown force` command or if there is an instance failure, you'll end up with an *inconsistent* database, wherein the database is said to be in a "dirty" state. Once the `shutdown` command is issued or the instance is terminated abruptly because of some reason, the following things will be true:

- Any committed changes are *not* rolled back automatically.
- Changes made to the database buffers aren't written to the datafiles on disk.
- All resources such as locks and latches are still held and aren't released.

In other words, there is simply no time to perform a graceful and tidy closure of the database. Your database instance is simple terminated, even though it may be in the middle of processing user transactions and hasn't properly recorded all the modified data to the datafiles. Upon restarting your database, the Oracle database instance will do the following things first:

- Use the information in the online redo logs to reapply changes.
- Use the undo tablespace contents to roll back the uncommitted changes to data.
- Release the resources held.

The work that the Oracle database performs upon a restart following an inconsistent shutdown is known as *instance recovery*. Instance recovery is thus mandatory and entirely automatic, with the database itself performing all the work without any intervention by the DBA.

Crash Recovery and Media Recovery

As noted in the previous section about instance or crash recovery, if your Oracle instance crashes, because of a power failure, for example, you don't have to perform a media recovery of the database, which requires that you restore backups of the database and bring them up-to-date with the help of the archived redo logs. The Oracle server will perform an automatic crash recovery when you restart the instance. However, if you lose a disk drive, for example, or you can't access the disk's contents because of some kind of media failure, you may have to restore your backups and bring them up-to-date using the archived redo logs.

Crash Recovery

Crash recovery or instance recovery is the automatic recovery of the database by the Oracle server, without any intervention by the DBA. For example, if a power outage brings down your database instance, when the power supply resumes, you only need to restart the database instance. You don't have to perform any restore or recovery tasks, because the server will use the information in the undo tablespace to perform automatic instance recovery by rolling back uncommitted transactions in the database. The server uses the online redo logs to record in the datafiles the changes that were committed before the outage but couldn't be written to the database files before the occurrence of the failure.

The Oracle server automatically performs crash recovery whenever you open a database whose files were not cleanly synchronized before shutting down. Since an abrupt shutdown doesn't provide a chance to synchronize the datafiles, it is a given that, in most cases, an instance recovery will be performed by the Oracle server when you restart the Oracle instance. The Oracle server will use the information saved in the online redo log files to synchronize the datafiles. Instance recovery involves the following two key operations:

Rolling forward: During this operation, the Oracle server will update all datafiles with the information from the redo log files. The online redo log files are always written to before the data is recorded in the datafiles. Thus, an instance recovery may usually leave the online log files "ahead" of the datafiles.

Rolling back: During this operation, uncommitted changes that were added to the datafiles during the rollforward operation are rolled back. Oracle does this by using the undo tablespace contents to return uncommitted changes to their original states. At the end of the rollback stage, *only committed data* at the time of the instance failure is retained in the datafiles.

During instance recovery, in the first rollforward operation, the database server must apply all transactions between the last checkpoint and the end of the redo log to the datafiles. Thus, in order to tune instance recovery, you control the gap between the checkpoint position and the end of the redo log. You use the Oracle initialization parameter `fast_start_mttr_target` to specify the number of seconds you want the crash recovery to take. Oracle will try to recover the instance as close as possible to the time that you specify for the `fast_start_mttr_target` parameter. The maximum value of this parameter is 3,600 seconds (1 hour).

Media Recovery

When a disk drive fails and you can't access the contents of an Oracle datafile, you're looking at a potentially much more serious situation than a crash recovery, since the server won't be able to automatically recover from such a catastrophe. You must provide the lost datafiles from backup. Since it's likely that data has changed in the meanwhile, you must provide the changes stored both in the archived redo log files and in the online redo log files. When the Oracle database issues an error indicating media problems, you must first find which files you must recover by querying the V$RECOVER_FILE view, which lists all files that need media recovery.

RMAN completely automates the process of media recovery. You use two basic commands—`restore` and `recover`—to perform media recovery. The `restore` command restores the necessary datafiles from RMAN's backup sets or image copies to the same or an alternative location on disk. The `recover` command performs the recovery process by applying necessary archived redo logs or incremental backups to the restored datafiles. You must do the following as part of a media recovery operation:

- Restore the necessary datafiles from backup, either to the old or to an alternative location.
- Rename the datafiles, if necessary, so the database will know about their new location.
- Recover the datafiles (bring them up to date), if necessary, by applying redo information to them.

To open the database after a successful restore and recovery, the following must be true:

- You must have synchronized copies of all the control files.
- You must have synchronized online datafiles.
- You must have at least one member of each redo log group.

If all these are true, you can open the recovered database.

Complete and Point-in-Time Recovery

You perform a complete recovery when you bring a database, a tablespace, or a datafile up-to-date with the most current point in time possible. It's important to emphasize that complete recovery isn't synonymous with recovering the complete database. Rather, completeness here alludes to the completeness of the entire database or part of it (tablespace or datafile) with reference to the time element. If you update the database tablespace or datafile completely by applying all changes from the archived redo logs to the backup files, you're performing a complete backup. In other words, complete recovery will ensure that you haven't lost any transactions. Note that when using RMAN, you may also use incremental backups as well, in addition to archived redo logs, during the recovery process.

When you perform media recovery, it isn't always the case that you can or should bring the database up-to-date to the latest possible point in time. Sometimes you may not want to recover the database to the current point in time. Following a loss of a disk or some other problem, the complete recovery of a database will make the database current by bringing all of its contents up to the present. A *point-in-time recovery*, also known as *incomplete recovery*, brings the database to a specified time in the past. A point-in-time recovery implies that changes made to the database after the specified point may be missing. On the face of it, a

point-in-time recovery may seem strange. After all, why would you recover your database only to a past period in time and not bring it up-to-date? Well, there may be situations where a point-in-time recovery is your best bet, as in the following examples:

- You lose some of the archived redo logs or incremental backups necessary for a complete recovery following a media failure.
- The DBA or the users delete data by mistake or make wrong updates to a table.
- A batch job that's making updates fails to complete.

In all of these situations, you can use either point-in-time recovery or Oracle's flashback technology to get the database back to a previous point in time. Prior to the introduction of the flashback technology, a database point-in-time recovery (DBPITR) and a tablespace point-in-time recovery (TSPITR) were the automatic solutions when confronted by situations such as an erroneous data entry or wrong updates. Flashback technology offers you the capability to perform point-in-time recovery much quicker than the traditional point-in-time recovery techniques that rely on media recovery. The flashback database feature is the alternative to traditional database point-in-time recovery, while the flashback table feature lets you avoid having to perform a media recovery in most cases.

Deciding on the Appropriate Recovery Technique

Fortunately for the Oracle database administrators, several recovery techniques are available, such as media recovery, Oracle flashback, and so on, each geared toward recovering from a certain type of problem. Here's a summary of when to use the various types of recovery techniques:

- Use media recovery if you're confronted with damaged, missing, or inaccessible datafiles.
- If a user drops a table or commits a major data entry error, you can perform a point-in-time media recovery, but the best option is to use the `flashback drop` feature. You can also import the affected table using the Data Pump Import utility or have users reenter data in some situations.
- If you run into logical errors, perform a TSPITR or consider using an appropriate flashback technique to make a point-in-time recovery.
- If you have data corruption in a few blocks in a datafile or a set of datafiles, use block media recovery. Again, there's no need to perform a media recovery and make the rest of the database inaccessible.
- If a user error affects a large set of tables or the entire database, use the flashback database feature to revert the database to a previous "good" time by undoing all the changes since that point in time.
- Use the flashback table feature to revert to a previous state of a table in order to undo unwanted changes.

RMAN Architecture

You can start performing backups with RMAN without installing or configuring a thing. Simply invoke the RMAN client by using the RMAN executable (named `rman`) from the $ORACLE_HOME/bin directory, and you're ready to go. Just specify the target database you want to work with at the command line, and that's it. You can perform backup and recovery actions with RMAN through the RMAN client or through the Enterprise Manager GUI.

In addition to the RMAN client, you may use additional optional components to make your backup and recovery strategy robust and easy:

The recovery catalog: The target database control file will always store the RMAN repository, which is the set of RMAN-related backup and recovery information. This data is also referred to as RMAN's *metadata*. However, it's smarter to use a dedicated database to store the RMAN repository. You can then create a special schema called the *recovery catalog* in this dedicated database and have RMAN store its repository in it, thus avoiding the risk of the critical metadata being overwritten when the control file runs out of space. As you'll see in Chapter 6, using a recovery catalog, which is optional, has several other advantages.

The flash recovery area: This is a location on disk where the database will store the backup and recovery–related files. This is also optional but highly recommended. See Chapter 3 for a detailed discussion of the flash recovery area.

Media management layer: As mentioned earlier, RMAN can directly interact only with disk drives. If you want to use tape drives to store your backups, you'll need a media management layer in addition to RMAN, since RMAN can't directly interact with the tape drives. You can use any of several Oracle-certified third-party media management layers. Oracle also provides Oracle Secure Backup, which it claims is the "most well-integrated media management layer for RMAN backups." Together, RMAN and Oracle Secure Backup provide a complete end-to-end backup solution for all Oracle environments. Chapter 18 deals with the media management layer.

An RMAN session in Unix/Linux systems consists of the following processes:

- The RMAN client process.
- A default channel, which is the connection to the target database.
- Additional channels you allocate and the corresponding target connection to each of the target databases.
- If you're using the recovery catalog, there will be a catalog connection to the recovery catalog database.
- During database duplication or TSPITR operations, there will be an auxiliary connection to the auxiliary instance.
- By default, RMAN makes one polling connection to each of the target databases to help monitor the execution of RMAN commands on the different allocated channels.

Benefits of Using RMAN

You can perform basic backup and recovery tasks using operating system utilities and standard SQL commands. However, there are several drawbacks to using these so-called user-managed backup and recovery techniques. For example, you can't perform incremental backups using user-managed techniques. In general, user-managed backup and recovery techniques require you to manually keep track of your backup files, their status, and their availability. You must write your own SQL and operating system scripts to manage the backup and recovery operations. In addition, you must provide the necessary datafiles and archived log files during a database recovery operation. If the database is operating during your backups (online or hot backups), you must place the database files in the backup mode before performing the actual file backups.

Oracle explicitly states that you can use user-managed techniques to perform backup/recovery activities. Oracle actually states that both user-managed techniques and RMAN are alternative ways of performing backup and recovery tasks. However, Oracle strongly recommends using RMAN to make your backups and perform database recovery, because of the tool's strengths and powerful features. Although you can perform a basic backup and recovery task with user-managed techniques without ever having to even start the RMAN interface, you should make RMAN your main backup and recovery tool for several reasons. Several important backup and recovery features are available to you only through RMAN.

Here's a brief description of the important benefits of using RMAN instead of user-managed backup and recovery techniques:

- You can take advantage of the powerful Data Recovery Advisor feature, which enables you to easily diagnose and repair data failures and corruption (Chapter 20 discusses the Data Recovery Advisor).
- There are simpler backup and recovery commands.
- It automatically manages the backup files without DBA intervention.
- It automatically deletes unnecessary backup datafiles and archived redo log files both from disk and tape.
- It provides you with detailed reporting of backup actions.
- It provides considerable help in duplicating a database or creating a standby database.
- It lets you test whether you can recover your database, without actually restoring data.
- It lets you verify that available backups are usable for recovery.
- It lets you make incremental backups, which isn't possible by any other means of backup.
- It lets you perform database duplication without backups by using the network-enabled database duplication feature, also known as *active duplication*.
- It automatically detects corrupt data blocks during backups, with the corruption relevant information recorded in the V$DATABASE_BLOCK_CORRUPTION view.
- When only a few data blocks are corrupted, you can recover at the data block level, instead of recovering an entire datafile.

- You can take advantage of the unused block compression feature, wherein RMAN skips unused data blocks during a backup.
- Only RMAN provides the ability to perform encrypted backups.
- You can use RMAN with a variety of third-party storage systems.
- You can use a powerful yet easy-to-use scripting language, which lets you write custom backup and recovery scripts quickly.

Backup and Recovery Best Practices

To successfully recover from unforeseen database mishaps, you must of course be fully conversant with the Oracle recovery techniques and concepts. In addition, you must ensure you are following certain basic steps to make sure you can successfully carry out the database recovery when you're pressured for time.

In addition, you must always document your backup and recovery procedures. You must have a detailed recovery plan for each type of failure you anticipate. If possible, you must write scripts to automate the execution of the recovery plan during a crisis. You must also update the written backup and recovery procedures on a regular basis and communicate these changes to all the personnel involved in the backup and recovery process in your organization. The following is a summary of basic Oracle backup and recovery best practices that will ensure that your database recovery efforts are successful.

Configure a Flash Recovery Area

It's common for backed-up datafiles and archived redo logs to be archived to tape storage. However, the problem is that when you're recovering a database, tape drives are rather slow media to copy to disk. Oracle strongly supports a*utomatic disk-based backup and recovery*, wherein all the necessary backup files are stored on disk itself. You make the initial copy of the necessary datafiles and archived redo log files to the flash recovery area and, from here, copy them to tape so you can store them off-site in a secure location.

Oracle recommends using the *flash recovery area* to store the entire set of backup and recovery–related files. The flash recovery area is simply a location on a server where you decide to store backup and recovery–related files such as RMAN's backup pieces, copies of control files and the online redo log files, and so on. At the minimum, Oracle recommends that you size the flash recovery area large enough to hold all archived redo logs that have not yet been copied to tape. It's easy to maintain the flash recovery area—all you have to do is specify the size of the area and the retention policy, which dictates when RMAN will discard unnecessary files from the flash recovery area. It's RMAN's job to keep the maximum number of backups possible in the flash recovery area, while discarding both obsolete backups and the backup files already copied to tape.

Oracle recommends that you size the flash recovery area large enough so it equals the sum of the size of the database plus the size of the archived redo logs not yet copied to tape and the size of any incremental backups.

Although the flash recovery area is by no means mandatory, Oracle recommends that you use one. You must have activated a flash recovery area in order to avail of the flashback database or the guaranteed restore point feature. In addition, using a flash recovery area means you're reducing your recovery time, since necessary backups and archived redo logs can be

kept on disk instead of having to recover from tape backups. Since obsolete backups are automatically deleted when space is needed for fresh files, you won't be running the risk of accidentally deleting necessary files.

Make and Protect a Database Redundancy Set

You may have to perform a database recovery when you lose or can't access (because of a media problem) any of these three types of Oracle database files: datafiles, online redo log files, and control files. Oracle recommends that you maintain a *database redundancy set*, which is a set of files that'll help you recover any of the three key types of Oracle files when they become unavailable to the database. This essential set of recovery-related files, called the *redundancy* set, will enable you to recover your database from any contingency. Here are the components of the redundancy set:

- Most recent backups of all datafiles plus the control file
- All archived redo logs made after the last backup
- Current control files and online redo file copies
- Oracle database-related configuration file copies (spfile, password file, tnsnames.ora and listener.ora files, for example)

To maintain the database redundancy set described here, you must duplex the control file as well as the online redo log files *at the database level*. That is, although a mirrored disk setup means that a copy of the redo log files and the control file will be automatically made at the operating system level, that doesn't provide you with complete safety.

Although you can mirror the online redo files at the operating system level, Oracle advises against this. Follow these Oracle best practices for protecting your database files:

- Multiplex the online redo log file at the database level. If you're using the flash recovery area, make this the destination for the duplexed copies of the online redo log file.
- Ensure that you use hardware or software (OS) mirroring to duplex the control file. This way, the database will always continue to operate following the loss of one control file.
- Mirror the datafiles in the database so you don't have to perform media recovery for simple disk failures.
- Keep more than one set of backups so you can withstand a database corruption issue.
- Consider making more than one copy of the redundancy set on tape if you aren't going to be using a disk-based recovery plan.

Oracle recommends that you use at least two disk drives on all production systems (one for the redundancy set and the other for the datafiles) and completely separate them by using different volumes, file systems, disk controllers, and RAID devices to hold the two sets of files: database files and the files in the redundancy set. One way to do this is to simply use the Oracle recommended flash recovery area. In fact, Oracle recommends the flash recovery area as a logical candidate to keep a copy of all the files belonging to the redundancy set (which includes the most recent database backup) on disk.

Create Powerful Backup Strategies

The strength of your backup strategy determines the strength of your recovery strategy. No backups, no recovery! Your backup strategies are derived entirely from your recovery strategies. Ideally, you must plan your recovery strategy based on the potential types of database failures you might encounter. The more types of database failures you want to guard against, the more complex your backup strategy will be.

Schedule Regular Backups

Schedule your backups on a regular basis, thus reducing your exposure to media failures. You, of course, can recover any database from a backup made at any remote time in the past, provided you have all the archived redo logs from that point forward. But can you imagine applying all those archived redo logs to the backups and suffering a horrendous downtime?

Create Regular Backups of the Control File

Back up a database's control file after any structural change to your database, such as creating a new tablespace or adding or renaming a datafile or an online redo log member. The best way to do this is to issue the RMAN command `configure controlfile autobackup on`. By default, the automatic backup of the control file is turned off. By turning control file autobackups on, you make sure that at the end of every RMAN `backup` command, RMAN automatically backs up the control file. When you make some changes via SQL*Plus, even though you're outside the purview of RMAN, the control file is automatically backed up, if you set the control file autobackup feature on. Using the control file autobackup, you can restore RMAN's backup and recovery information (called RMAN's *repository*), when you lose all your control files and aren't using the optional recovery catalog.

Run the Database in Archivelog Mode

To be able to restore a database completely (that is, bring them up-to-date with all the changes ever made to that database), you must run the database in archivelog mode. Only development and test databases where data loss isn't an issue should be run in noarchivelog mode.

Multiplex the Control File

Since the control file is absolutely necessary during a recovery, use the following guidelines to safeguard the control file:

- Keep the Oracle-recommended three copies of the control file.
- Put each copy of the control file on a separate disk.
- Place at least one of the three copies on a separate disk controller.

Multiplex the Redo Log Groups

If you lose your online redo logs, you may not be able to recover all committed changes to your database following a media failure and subsequent recovery. You must always duplex the online redo logs, using the following guidelines:

- Have a minimum of two members in each redo log group.
- Place each member on a separate disk drive.
- Place each redo log member on a separate disk controller.

Adopt the Right Backup Storage Strategy

Where you store your backups is quite critical to your recovery strategy, since different storage strategies have different implications for recovery time. If you use a flash recovery area, of course, the backups are all on disk, and consequently, you can recover with the least amount of elapsed time. If you store your backups only on tape or you store them off-site, it means you have to endure a longer interval to restore and recover your database.

Plan Your Backup Retention Duration

One of the key questions every backup strategy must address is how long you want to keep a backup. Although you can specify that a backup be kept forever without becoming obsolete, it's not common to follow such a strategy, unless you're doing it for a special reason. Instead, backups become obsolete according to the retention policy you adopt. You can select the retention duration of backups when using RMAN in two ways. In the first method, you can specify backup retention based on a *recovery window*. That is, all backups necessary to perform a point-in-time recovery to a specified past point of time will be retained by RMAN. If a backup is older than the point of time you chose, that backup will become obsolete according to the backup retention rules. The second way to specify the retention duration is to use a *redundancy-based* retention policy, under which you specify the number of backups of a file that must be kept on disk. Any backups of a datafile greater than that number will be considered obsolete.

You can set a default retention policy for all files that RMAN backs up. Once you do this, you can choose to delete any files that are obsolete under that retention policy using simple RMAN commands. The files you delete may be on disk or on tape storage. When you delete the obsolete files using RMAN commands, RMAN will remove the relevant information from its metadata. If, however, you're using the flash recovery area to store your backups, RMAN will automatically delete all obsolete files as and when it needs space for accommodating newer datafile backups or archived redo logs in the flash recovery area.

Plan Your Backup Schedules

Determining a backup schedule means how often you use RMAN to back up your database files, as well as what files you back up. Do you perform nightly or weekly backups, or do you back up different files at different intervals? How frequently you create a backup will, of course, depend on how fast the data in your database is changing. If your database performs a very large number of DML operations on a daily basis, you must back it up on a daily basis rather than a weekly basis, for example. If, on the other hand, a database is being mostly used for lookup purposes, with minimal DML changes, you can back up at a more infrequent interval, say on a weekly basis. An incremental backup strategy may be especially apt in a case such as this, because of the small amount of changes.

Validate Your Recovery Strategy

A key part of a backup and recovery strategy is the validation of your backups. Merely backing up the database regularly doesn't guarantee that you can recover your database successfully with those backups. You must choose a method to regularly validate the backups you take with RMAN. Since the only goal in creating database backups is to use them in a recovery situation, you must make sure you regularly validate your backups and test your data recovery strategy. RMAN provides commands that let you validate the database files you're planning to back up by reading those files without actually backing them up.

Conduct Regular Trial Recoveries

Another key part of a solid backup and recovery strategy is to schedule regular trial recoveries using your current recovery plan and the latest backups for various simulated scenarios. In addition to verifying that your backups are being made correctly, you'll also get plenty of practice with the recovery techniques and commands. Aside from that, it is only during the test restore/recovery that you'll know the duration of a restore/recovery and, therefore, how fast you can perform the actual restore/recovery.

It's much better to get acquainted with the recovery techniques this way rather than to try them for the first time after a production database runs into problems and you're under the gun to recover it fast.

■ **Note** You can configure the `nls_date_format` environment variable to include the date and time format, such as DD-MON-RRRR HH24:MI:SS (in the Korn shell, use the command `export nls_date_format=YYYY-MM-DD:HH24:MI:SS`) because by default only the data is displayed in the RMAN log. This is helpful when troubleshooting, because most often you want to know the exact date and time a specific problem or error occurred. Furthermore, this will also display the date/time of the RMAN backup completion and datafile checkpoints.

Record Accurate Software and Hardware Configuration

Always keep handy vital information that you might have to send to the Oracle Support personnel, such as the following:

- Server model and make
- Operating system version and patch number
- Oracle database version number and patch release
- Database identifier (DBID)
- Names and location of all your datafiles
- Version of the recovery catalog database and the recovery catalog schema, if you're using one
- Version of the media management software you are using

Of course, it's always a good idea to keep the complete RMAN log file generated during the RMAN backup (even though this is already captured in the V$RMAN_OUTPUT), which is useful when you lose the control file or recovery catalog that has information about the RMAN backups you want to restore from.

In this introductory chapter, we have provided a quick review of the essentials of Oracle backup and recovery concepts and have defined key terms. We also introduced the Recovery Manager tool and explained its basic architecture and an overview of its important features. Later chapters, of course, delve into the intricacies of using RMAN to perform backup and recovery.