Chapter No. 1
"Understanding the Basics of Backup and Recovery"
In this package, you will find:

A Biography of the authors of the book

A preview chapter from the book, Chapter NO.1 "Understanding the Basics of Backup and Recovery"

A synopsis of the book’s content

Information on where to buy this book

About the Authors

Francisco Munoz Alvarez has over two decades of experience in consulting, analysis, support, implementation, and migration of Oracle products. He is also an expert in most phases of a database life cycle, for example, development, stabilization, security, backup and recovery, tuning, installations, and data warehouse (ETL) with excellent implementation and support methodologies.

He is a popular speaker at many Oracle conferences around the world. He is also the President of CLOUG (Chilean Oracle Users Group), LAOUC (Latin American Oracle Users Group Community, which is the umbrella organization for all of Latin America), and NZOUG (New Zealand Oracle Users Group). He worked as an Associate Technologist for Oracle in Brazil and Chile. He was an Oracle instructor for the New Horizons Centre (Chile) and for Oracle Education (Brazil and Chile). He also worked in the first team to introduce Oracle to South America (Oracle 6 and the beta version of Oracle 7). He was also the first Master Oracle 7 Database Administrator in South America, as well as the first Latin American Oracle professional to be awarded a double ACE (ACE in 2008 and ACE Director in 2009) by Oracle HQ. In 2010, he had the privilege to receive a prestigious Oracle Magazine Editor's Choice Award as the Oracle Evangelist of the Year—a huge recognition for his

For More Information:  
outstanding achievements in the Oracle world that includes the creation and organization of the already famous OTN Tours that are the biggest Oracle evangelist events in the world.

Currently, Francisco works for Revera Limited, which is a leading provider of utility computing infrastructure and enterprise data management in New Zealand, as the Oracle Professional Services Manager since June 2011. He also maintains an Oracle blog (http://www.oraclenz.org) and you can always contact him through this or Twitter (@fcomunoz) regarding any questions about this book.

Aman Sharma is an Oracle Database consultant and instructor. He holds a Master's degree in Computer Applications and has been working with Oracle Database for over a decade. His main focus is to understand how Oracle Database works internally. Besides the core database, he has a very strong knowledge of Linux, Solaris, Oracle RAC, Data Guard, RMAN, Oracle Exadata and Oracle Enterprise Manager.

He is an Oracle Certified Professional and also an Oracle Certified Expert in Oracle RAC, SQL and Linux. He is also an Oracle Certified Implementation Specialist and a certified Solaris System Administrator. He loves to share his learning with the Oracle community, so when he is not delivering an Oracle related session, he spends a lot of his time participating in OTN (Oracle Technology Network) discussion forums. He also maintains an Oracle blog (http://blog.aristadba.com), is also on Twitter (@amansharma81), you can reach to him using either ways. He is a strong supporter of user groups and is a member of India Oracle Users Group (http://www.aioug.org) and has been a speaker at various Oracle conferences organized by AIOUG. He is also an associate member of IOUG (http://www.ioug.org). In 2010, he was awarded the prestigious Oracle ACE award from Oracle Corporation.

For More Information:  
Oracle Database 12c Backup and Recovery Survival Guide

Knowledge is only valuable when shared.

The three main responsibilities for a successful DBA are to ensure the availability, recoverability, and performance of any database that the DBA is accountable for.

This book will focus on the recoverability set of skills, and will also include some tips and ideas regarding availability. All examples showed in the book are executed over Oracle Enterprise Linux 6.4 and Oracle Database 12.1.0.1 (also known as Oracle 12c), please be aware of these specific versions to ensure that you will be able to reproduce the same results, you will see reflected in this book.

To ensure the recoverability of any database, a DBA needs to have a strong backup and recovery skills set; this knowledge is essential for any good DBA. Without this knowledge, you will be in violation of my most important rule that I have used in my entire career, "The most important rule with respect to data is to never put yourself into an unrecoverable situation". If you follow this simple tip every time you work with data, I can guarantee that you will be always protected against any possible situation that could surprise you in your daily journey, including of course the unexpected ones.

My key intention by writing this book is that if you are a new DBA, introduce you to this fantastic world that is vital to your success. If you are an experienced DBA, this book will become a reference guide and will also help you to learn some possible new skills, or give some new ideas that you never knew about. It will also help you to easily find the solution to some of the most well-known problems you could find during your career, and this book will be rich with screenshots, full of scripts, examples, and tutorials that you will find more than useful and handy.

Most of the books currently available in the market only concentrate on the RMAN utility for backup and recovery; this book will be an exception to the rule and will become a must-have reference to allow you to achieve a real complete backup and recovery strategy. This is not in any case a replacement to the official Oracle documentation available at http://www.oracle.com/pls/db121/homepage; I will always recommend to any serious DBA to read the complete documentation set as a complement to this book.

For More Information:
This book contains my knowledge of more than two decades working with Oracle technologies and also shows several topics and situations that came to my attention when speaking at several conferences around the world or helping others on Oracle forums or virtual communities.

I hope you will enjoy reading this book as the same way I enjoyed writing it.

**What This Book Covers**

*Chapter 1, Understanding the Basics of Backup and Recovery*, covers topics such as understanding the need for creating backups, getting familiar with the different backup types, an overview of backup strategy, understanding what is redo and how it affects your database recoverability, and understanding database operational modes and redo generation.

*Chapter 2, NOLOGGING Operations*, covers topics such as LOGGING versus NOLOGGING, disabling redo generation, NOLOGGING operations, how to reduce redo generation, redo log wait events, practice with some interesting scripts, and much more interesting topics.

*Chapter 3, What is New in 12c*, covers topics such as pluggable database, RMAN's new features and enhancements, and Data Pump's new features and enhancements.

*Chapter 4, User-managed Backup and Recovery*, covers backup and recovery using user-managed methods. Understanding the basics involving a manual backup and recovery will help you to easily understand what is going on in the background of your database when using RMAN and it will also help you to compare and easily understand all benefits of using RMAN against any other backup method when working with Oracle.

*Chapter 5, Understanding RMAN and Simple Backups*, describes that being the custodians of databases, DBA's should always try to minimize the loss of data. This can be accomplished through an effective strategy that enables us to secure a backup of these databases that can be accessed in case of systemic failures. However, it is rightly said that any strategy is as good as the tool that implements it. In this chapter, we shall introduce you to a tool that's just like one of those super heroes that can fix almost any issue. So in the list of such amazing heroes such as Superman and Batman, please welcome RMAN, the Recovery Manager—Oracle's one stop solution for both backups and recoveries.

*Chapter 6, Configuring and Recovering with RMAN*, looks into the two abilities of RMAN, that is, how to configure it and how to use it for doing database recoveries.

*Chapter 7, RMAN Reporting and Catalog Management*, discusses about the topics such as benefits of recovery catalog, creation and use of recovery catalog, using the CATALOG command, and RMAN reporting using the LIST and REPORT commands.

---

For More Information:  
Chapter 8, *RMAN Troubleshooting and Tuning*, looks at various ways to get the best performance from RMAN and also techniques to troubleshoot it when it won’t behave itself.

Chapter 9, *Understanding Data Pump*, describes about Data Pump and its architecture, new concepts with Data Pump, methods to move data, and play with many Data Pump scenarios.

Chapter 10, *Advanced Data Pump*, covers topics such as data masking, build a metadata repository, create a version control, clone users (create a new user using an existent user as a template), create smaller copies of production, create your database in a different file structure, move all objects from one tablespace to another, move a object to a different schema (a simple example, change a table owner), migrate data for a database upgrade, downgrade an Oracle database, transport a tablespace, and use Data Pump with flashback.

Chapter 11, *OEM12c and SQL Developer*, discusses topics such as configuring our backup and recovery settings (including catalog settings) in OEM12c, scheduling backups in OEM12c, creating restore points in OEM12c, understanding database export/import operations in OEM12c, and getting familiar with the SQL developer.

Appendix: *Scenarios and Examples – A Hands-on Lab*, allows you to practice some of the scenarios you saw in this book (step-by-step) and learn by practice. If you have any doubt about a command of what it will be doing, please refer to the corresponding chapter in this book for more information.

For More Information:  
Understanding the Basics of Backup and Recovery

The constant evolution of IT has, among other things, affected the role of a database administrator (DBA). Today the DBA is not merely a Database Administrator anymore, but is morphing more into the Database Architect role. If you want to become a successful DBA and be more competitive in the market, you should have a different skill set than what was normally required in the past. You need to have a wide range of understanding in architectural design, network, storage, licensing, and much more. The more knowledge you have, the better opportunities you will find.

The main idea of this chapter is to introduce you to some basic concepts regarding backup and recovery, giving you a general overview of the most important methods and tools available for you to achieve your backup goals. Therefore, in this chapter, we will cover the following topics:

- Understanding the need for creating backups
- Getting familiar with the different backup types
- An overview of backup strategy
- Understanding what is redo and how it affects your database recoverability
- Understanding database operational modes and redo generation

Purpose of backup and recovery

As a DBA, you are the person responsible for recovering the data and guarding the business continuity of your organization. Consequently, you have the key responsibility for developing, deploying, and managing an efficient backup and recovery strategy for your institution or clients that will allow them to easily recover from any possible disastrous situation. Remember, data is one of the most important assets a company can have. Most organizations would not survive after the loss of this important asset.

For More Information:

Testing backups

It's incredible how many corporations around the world do not have a proper disaster recovery plan (DRP) in place, and what is worse, many DBAs never even test their backups. Most of the time when auditing Oracle environments for clients, I ask the following question to the DBA team:

- Are you 100 percent sure that you can trust your backups? For this question I generally receive answers like:
  - I'm not 100 percent sure since we do not recover from backups too often
  - We do not test our backups, and so I cannot guarantee the recoverability of them

Another good question is the following:

- Do you know how long a full recovery of your database will take? Common responses to this question are:
  - Probably anything between 6 and 12 hours
  - I don't know, because I've never done a full recovery of my database

As you can see, a simple implementation of a procedure to proactively test the backups randomly will allow you to:

- **Test your backups and ensure that they are valid and recoverable:**
  I have been called several times to help clients because their current backups are not viable. Once I was called to help a client and discovered that their backup-to-disk starts every night at 10 P.M. and ends at 2 A.M. Afterwards, the backup files are copied to a tape by a system administrator every morning at 4 A.M. The problem here was that when this process was implemented, the database size was only 500 GB, but after few months, the size of the database had grown to over 1 TB. Consequently, the backup that was initially finishing before 2 A.M. was now finishing at 5 A.M., but the copy to a tape was still being triggered at 4 A.M. by the system administrator. As a result, all backups to a tape were unusable.

- **Know your recovery process in detail:** If you test your backups, you will have the knowledge to answer questions regarding how long a full recovery will take. Answering that your full recovery will take around three and a half hours, but you prefer to say five hours just in case of any unexpected problem that you will come across, you will look more professional. This will let me know that you really know what you are talking about.

For More Information:

Document and improve your recovery process: The complete process needs to be documented. If the process is documented and you also allow your team to practice on a rotation basis, this will ensure that they are familiar with the course of action and will have all the knowledge necessary to know what to do in case of a disaster. You will now be able to rest in your home at night without being disturbed, because now you are not the only person in the team with the experience required to perform this important task.

Good for you if you have a solid backup and recovery plan in place. But have you tested that plan? Have you verified your ability to recover?

Protecting data

As being the main person responsible for the recovery and availability of the data, you need to have a full understanding of how to protect your data against all possible situations you could come across in your daily job. The most common situations you could see are:

- Media failure
- Hardware failure
- Human error
- Application error

Let's take a closer look at each of these situations.

Media failure

Media failure occurs when a system is unable to write or read from a physical storage device such a disk or a tape due to a defect on the recording surface. This kind of failure can be easily overcome by ensuring that your data is saved on more than one disk (mirrored) using a solution such as RAID (Redundant Array of Independent Disks) or ASM (Automatic Storage Management). In the case of tapes, ensure that your backups are saved in more than one tape and as mentioned earlier, testing the recoverability from them.
Hardware failure

Hardware failure is when a failure occurs on a physical component of your hardware such as when your server motherboard, CPU, or any other component stops working. To overcome this kind of situation, you will need to have a high availability solution in place as part of your disaster and recovery strategy. This could include solutions such as Oracle RAC, a standby database, or even replacement hardware on the premises. If you are using Oracle Standard Edition or Standard Edition One and need to implement a proper standby database solution, I will recommend you to take a closer look at the Dbvisit Standby solution for Oracle databases that is currently available in the market to allow you to fulfill this need (http://dbvisit.com).

Human error

Human error, also known as user error, is when a user interacting directly or through an application causes damage to the data stored in the database or to the database itself. The most frequent examples of human error involve changing or deleting data and even files by mistake. It is likely that this kind of error is the greatest single cause of database downtime in a company.

No one is immune to user error. Even an experienced DBA or system administrator can delete a redo log file that has the extension .log as a mistake when taking it as a simple log file to be deleted to release space. Fortunately, user error can most of the time be solved by using physical backups, logical backups, and even Oracle Flashback technology.

Application error

An application error happens when a software malfunction causes data corruption in the logical or physical levels. A bug in the code can easily damage data or even corrupt a data block. This kind of problem can be solved using Oracle block media recovery, and is why it is so important to have a proper test done prior to promoting an application change to any production environment.

Always do a backup before and after a change is implemented in a production environment. A before backup will allow you to roll back to the previous state in case something goes wrong. An after backup will protect you to avoid to redo the change in case of a failure, due that it was not included in the previous backup available.

For More Information:

Types of backup

Now that you understand all types of possible failures that could affect your database, let's take a closer look at the definition of backup and the types of backups that are available to ensure the recoverability of our data.

A backup is a real and consistent copy of data from a database that could be used to reconstruct the data after an incident. Consequently, there are two different types of backups available, which are:

- Physical backups
- Logical backups

A physical backup

A physical backup is a copy of all the physical database files that are required to perform the recovery of a database. These include datafiles, control files, parameter files, and archived redo log files. As an Oracle DBA, we have different options to make a physical backup of our database. Backups can be taken using user-managed backup techniques or using Recovery Manager (RMAN). Both techniques will be discussed in more detail later in this book. Physical backups are the foundation of any serious backup and recovery strategy.

A logical backup

Oracle uses Oracle Data Pump to allow us to generate a logical backup that can be used to migrate data or even do a partial or full recovery of our database. The utilities available are the Data Pump Export program (expdp) and the Data Pump Import program (impdp).

Many people have a misconception of these tools in thinking that they can only be used to move data. Data Pump is a very flexible and powerful tool that if well utilized can easily become a DBA's best friend. It is not just for moving data. It can also play a crucial role in your backup and recovery strategy.

Chapter 9, Understanding Data Pump and Chapter 10, Advanced Data Pump will go into more detail about the use of Data Pump for logical backup and recovery.

For More Information:
The old Import and Export utilities

In the previous versions of Oracle we used to work with similar utilities called exp and imp. The exp utility is deprecated since Oracle 11g, but the imp utility is still currently supported by Oracle. The imp utility allows us to recover any backup generated by the old exp program. Just keep in mind that the use of exp is not supported anymore by Oracle and using it can bring future trouble to your environment.

Backup strategies

A backup and recovery strategy has the main purpose of protecting a database against data loss, and this document will contain all steps required to reconstruct the database after a disaster strikes. As the person responsible for the data of your company, it is very important to have a correct backup strategy in place to allow you to recover from any possible disaster.

Before you create a strategy, you will need to understand clearly all the Service Level Agreements (SLAs) in place with in your organization regarding this topic. To that end, you will need to ask some simple questions to the owners of the data:

• How much data can the company lose in case of a disaster? (RPO)
• How much time could the business wait to have the data restored and available again? (RTO)
• How much will it cost the company for the loss of one hour of data?
• What retention periods are required by law for the company data?

After receiving the answers to all these questions, you will be able to implement a proper backup and recovery strategy according to your real company needs and SLAs in place.

For example, if your company can only afford to lose three hours of data (RPO) but it can have the database down for up to 24 hours for a recovery process (RTO), all you will need to do to fulfill your SLA is to have a full backup of your database made daily. You will also need to make backups of all your archive logs every three hours to a tape or another network location to allow you to have all your data protected.

For More Information:

As part of creating a strategy, it is important to properly understand the concepts known as **Recovery Point Objective (RPO)** and **Recovery Time Objective (RTO)**. As you can see in the following figure, the RPO reflects how much data might be lost without incurring a significant risk or loss to the business, and the RTO is basically the maximum amount of time allowed to reestablish the service after an incident without affecting the company seriously.

![Diagram showing RPO and RTO]

**Restore versus recovery**

On several occasions, people have asked me about the differences between *restore* and *recovery*. Due to these questions, I will take this opportunity to explain the difference in some simple words to make it clear:

- **Restore**: It is the act that involves the restoration of all files that will be required to recover your database to a consistent state, for example, copying all backup files from a secondary location such as tape or storage to your stage area.

- **Recovery**: It is the process to apply all transactions recorded in your archive logs, rolling your database forward to a point-in-time or until the last transaction recorded is applied, thus recovering your database to the point-in-time you need.

You will see some examples of how restore and recovery work later in this book. Now let's take a closer look at what is redo log and the two possible modes your database could be operating in. This will help you understand in a bit more in depth what type of backup and recovery you could use on your environment.

For More Information:
What is redo?

Let’s look briefly at the redo process. When Oracle blocks (the smallest unit of storage in a database) are changed, including UNDO blocks, Oracle records the changes in the form of vector changes, which are referred to as redo entries or redo records. The changes are written by the server process to the redo log buffer in the System Global Area (SGA). The redo log buffer will then be flushed into the online redo logs in near real-time fashion by the log writer (LGWR) process (if the redo log buffer is too small, then you will start seeing log buffer space waits during bursts of redo generation).

The redo entries are written by the LGWR to a disk when:

- A user issues a commit
- The log buffer is one third full
- The amount of unwritten redo entries is 1 MB
- When a database checkpoint takes place
- Otherwise every three seconds

Redo entries are written to disk when one of the situations mentioned take place first. In the event of a checkpoint, the redo entries are written before the checkpoint to ensure recoverability.

Redo log files record changes to the database as a result of transactions and internal Oracle server actions. Redo log files protect the database from loss of integrity due to system failures caused by power outages, disk failures, and so on. Redo log files must be multiplexed using different disks (use of fast disk is preferred) to ensure that the information stored in them is not lost in the event of a disk failure.

The redo log consists of groups of redo log files. A group consists of a redo log file and its multiplexed copies. Each identical copy is said to be a member of that group, and each group is identified by a number. The LGWR process writes redo records from the redo log buffer to all members of a redo log group until the file is filled or a log switch operation is requested. Then, it switches and writes to the files in the next group. Redo log groups are used in a circular fashion as shown in the following figure:
Redo log groups need to have at least two files per group, with the files distributed on separate disks or controllers so that no single disk failure destroys an entire log group. Also, never rely exclusively on your ASM disk group or the file system if they have mirrored disks underneath. Remember that mirroring will not protect your database in the event of your online redo log file being deleted or corrupted.

The loss of an entire log group is one of the most serious possible media failures you can come across because it can result in loss of data. The loss of a single member within a multiple-member log group is trivial and does not affect database operation, other than causing an alert to be published in the alert log.

Remember that redo logs heavily influence database performance because a commit cannot be completed until the transaction information has been written to the logs. You must place your redo log files on your fastest disks served by your fastest controllers. If possible, do not place any other database files on the same disks as your redo log files.

It's not advisable to place members of different groups on the same disk. That's because the archiving process reads the online redo log files and will end up competing with the LGWR process.

For More Information:
As a resume about redo log files and redo log groups, it is important always to:

- Have a minimum of three redo log groups. If your database switches too often and you do not have an appropriate number of redo log groups, the LGWR process will need to wait until the next group is available before being able to overwrite it.
- All online redo logs and standby redo logs are equal in size.
- Tune your redo log files size to allow redo log switches to happen at no less than 20 minutes from each other at peak times.
- Remember to place the redo log files on high performance disks.
- Remember to have a minimum of two redo log members per group to reduce risk, and place them in different disks away from the data.
- Do not multiplex standby redo logs to prevent additional writes in the redo transport.

Remember as mentioned earlier that it is important to note that not all Oracle databases will have the archive process enabled.

**Redo generation and recoverability**

The purpose of redo generation is to ensure recoverability. This is the reason why Oracle does not give the DBA a lot of control over redo generation. If the instance crashes, then all the changes within the SGA will be lost. Oracle will then use the redo entries in the online redo log files to bring the database to a consistent state. The cost of maintaining the redo log records is an expensive operation involving latch management operations (CPU) and frequent write access to the redo log files (I/O).

You can avoid redo logging for certain operations using the `NOLOGGING` feature. We will talk more about the `NOLOGGING` feature in *Chapter 2, NOLOGGING Operations*.

**The NOARCHIVELOG mode**

When your database is created by default, it will be created using the `NOARCHIVELOG` mode. This mode permits any normal database operations but will not provide your database with the capability to perform any point-in-time recovery operations or online backups of your database.
When the database is using this mode, no hot backup is possible (hot backup is any backup done with the database open, causing no interruption for the users). You will only be able to perform backups with your database down (shutdown, also known as the offline backup or the cold backup), and you will only be able to perform a full recovery up to the point that your backup was made. You can see in the following example what will happen if you try to make a hot backup of your database when in the NOARCHIVELOG mode:

```sql
SQL> SELECT log_mode FROM v$database;

LOG_MODE
--------
NOARCHIVELOG
```

```sql
SQL> ALTER DATABASE BEGIN BACKUP;
ALTER DATABASE BEGIN BACKUP
*
ERROR at line 1:
ORA-01123: cannot start online backup; media recovery not enabled
```

The error shown in the preceding code is the result you will receive after trying to place your database in backup mode to make a hot backup of your database files. The example to follow shows the result you will receive when trying to make a backup of your open database when in the NOARCHIVELOG mode using RMAN. As you can see, neither approach is possible:

```sql
RMAN> BACKUP DATABASE;
```

Starting backup at 04-DEC-12
using target database control file instead of recovery catalog
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=36 device type=DISK
channel ORA_DISK_1: starting full datafile backup set
channel ORA_DISK_1: specifying datafile(s) in backup set
RMAN-03009: failure of backup command on ORA_DISK_1 channel at
12/04/2012 15:32:42
ORA-19602: cannot backup or copy active file in NOARCHIVELOG mode
continuing other job steps, job failed will not be re-run
channel ORA_DISK_1: starting full datafile backup set
channel ORA_DISK_1: specifying datafile(s) in backup set
including current control file in backup set
including current SPFILE in backup set

For More Information:
channel ORA_DISK_1: starting piece 1 at 04-DEC-12
channel ORA_DISK_1: finished piece 1 at 04-DEC-12
piece handle=/home/oracle/app/oracle/flash_recovery_area/ORCL/backupset/201
2_12_04/o1_mf_ncsnf_TAG20121204T153241_8cx20wfz_.bkp
tag=TAG20121204T153241 comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time: 00:00:01
RMAN-00571: ======================================================  
RMAN-00569: ========== ERROR MESSAGE STACK FOLLOWS ===============  
RMAN-00571: ======================================================  
RMAN-03009: failure of backup command on ORA_DISK_1 channel at 
12/04/2012 15:32:42
ORA-19602: cannot backup or copy active file in NOARCHIVELOG mode

To make a full backup of your database when in the NOARCHIVELOG mode, you will need to:

1. First shut down your database completely in a consistent mode.
2. Backup all your datafiles, parameter files, a control file, and your redo logs manually to a tape or a different location.
3. Re-start your database.

If a recovery is required, all you will need to do is to restore all files from your last backup and start the database, but you need to understand that all transactions made in the database after your backup will be lost.

The ARCHIVELOG mode
Oracle lets you save filled redo log files to one or more offline destinations to improve the recoverability of your data by having all transactions saved in case of a crash, reducing any possibility of data loss. The copy of the redo log file containing transactions against your database made to a different location is known as an ARCHIVELOG file, and the process of turning redo log files into archived redo log files is called archiving.

For More Information:  
Understanding the ARCHIVELOG mode

An archived redo log file is a physical copy of one of the filled members of a redo log group. Remember that redo log files are cyclical files that are overwritten by the Oracle database and are only archived (backup copy of the file before being overwritten) when the database is in the ARCHIVELOG mode. Each redo log file includes all redo entries and the unique log sequence number of the identical member of the redo log group. To make this point more clear, if you are multiplexing your redo log file (recommended to a minimum of two members per group), and if your redo log group 1 contains two identical member files such as redolog_1a.rdo and redolog_1b.rdo, then the archive process (ARCn) will only archive one of these member files, not both. If redo log file redolog_1a.rdo becomes corrupted, then the ARCn process will still be able to archive the identical surviving redo log file redolog_1b.rdo. The archived redo log generated by the ARCn process will contain a copy of every group created since you enabled archiving in your database.

When the database is running in the ARCHIVELOG mode, the LGWR process cannot reuse and hence overwrite a given redo log group until it has been archived. This is to ensure the recoverability of your data. The background process ARCn will automate the archiving operation and the database will start multiple archive processes as necessary (the default number of processes is four) to ensure that the archiving of filled redo log files does not fall behind.

You can use archived redo logs to:

- Recover a database
- Update and keep a standby database in sync with a primary database
- Get information about the history of a database using the LogMiner utility

For More Information:
In the ARCHIVELOG mode, the Oracle Database engine will make copies of all online redo log files via an internal process called ARCn. This process will generate archive copies of your redo log files to one or more archive log destination directories. The number and location of destination directories will depend on your database initialization parameters.

To use the ARCHIVELOG mode, you will need to first set up some configuration parameters. Once your database is in the ARCHIVELOG mode, all database activity regarding your transactions will be archived to allow your data recoverability and you will need to ensure that your archival destination area always has enough space available. If space runs out, your database will suspend all activities until it becomes able once again to back up your redo log files in the archival destination. This behavior happens to ensure the recoverability of your database.

Never use the extension .log for redo log files. As mentioned earlier, use a different extension such as, for example, .rdo. This is because anyone, including you, can delete .log files by mistake when running out of space.

Preparing for the ARCHIVELOG mode

When setting your database to work in the ARCHIVELOG mode, please never forget to:

1. Configure your database in a proper way. Some examples of what to do when configuring a database are:
   - **Read the Oracle documentation**: It's always important to follow Oracle recommendations in the documentation.
   - **Have a minimum of three control files**: This will reduce the risk of losing a control file.
   - **Set the CONTROL_FILE_RECORD_KEEP_TIME initialization parameter to an acceptable value**: Doing so will set the number of days before a reusable record in the control file can be reused. It will also control the period of time that your backup information will be stored in the control file.
   - **Configure the size of redo log files and groups appropriately**: If not configured properly, the Oracle Database engine will generate constant checkpoints that will create a high load on the buffer cache and I/O system affecting the performance of your database. Also, having few redo log files in a system will force the LGWR process to wait for the ARCh process to finish before overwriting a redo log file.

For More Information:

Chapter 1

- **Multiplex online redo log files**: Do this to reduce the risk of losing an online redo log file.

- **Enable block checksums**: This will allow the Oracle Database engine to detect corrupted situations.

- **Enable database block checking**: This allows Oracle to perform block checking for corruption, but be aware that it can cause overhead in most applications depending on workload and the parameter value.

- **Log checkpoints to the alert log**: Doing so helps you determine whether checkpoints are occurring at a desired frequency.

- **Use fast-start fault recovery feature**: This is used to reduce the time required for cache recovery. The parameter `FAST_START_MTTR_TARGET` is the one to look over here.

- **Use Oracle restart**: This is used to enhance the availability of a single instance (non-RAC) and its components.

- **Never use the extension .log for redo log files**: As mentioned earlier, anyone including you, can delete .log files by mistake when running out of space.

- **Use block change tracking**: This is used to allow incremental backups to run to completion more quickly than otherwise.

2. Always be sure to have enough available space in the archival destination.

3. Always make sure that everything is working as it is supposed to be working. Never forget to implement a proactive monitoring strategy using scripts or Oracle Enterprise Manager (OEM). Some important areas to check are:

- Database structure integrity
- Data block integrity
- Redo integrity
- Undo segment integrity
- Transaction integrity
- Dictionary integrity

For More Information:
Checking the status of the ARCHIVELOG mode

You can determine which mode or if archiving, is being used in your instance by issuing an SQL query to the log_mode field in the v$database (ARCHIVELOG indicates archiving is enabled and NOARCHIVELOG indicates that archiving is not enabled) or by issuing the SQL archive log list command:

```
SQL> SELECT log_mode FROM v$database;

LOG_MODE
-------------
ARCHIVELOG
```

```
SQL> archive log list

Database log mode                 Archive Mode
Automatic archival                 Enabled
Archive destination                USE_DB_RECOVERY_FILE_DEST
Oldest online log sequence         8
Next log sequence to archive       10
Current log sequence               10
```

Specifying parameters

When in the ARCHIVELOG mode, you can choose between generating archive redo logs to a single location or multiplexing them. The most important parameters you need to be familiar with when setting your database to work in this mode are:

- `LOG_ARCHIVE_DEST_n`: Use this parameter to specify from one to ten different archival locations (n can be a number between 1 and 10).
- `LOG_ARCHIVE_FORMAT`: This parameter will specify the default filename format when archiving the redo log files. The following variables can be used to format the file:
  - `%s` — log sequence number
  - `%S` — log sequence number, zero filled

For More Information:
One example of how to make use of these parameters could be something like this:

```
alter system set log_archive_format="orcl_%s_%t_%r.arc" scope=spfile.
```

This command will create archive log files with a name that will contain the word "orcl" that is the database ID, the log sequence number, the thread number, and the resetlogs ID.

- **LOG_ARCHIVE_MIN_SUCCEED_DEST**: This defines the minimum number of archival destinations that must succeed in order to allow a redo log file to be overwritten

### Viewing the status of archival destinations

You can also check the status of your archival destinations by querying the `V$ARCHIVE_DEST` view, in which the following variable characteristics will determine the status:

- **Valid/Invalid**: This indicates whether the disk location or service name specified is valid or not
- **Enabled/Disabled**: This indicates the availability state of the location and if the database can use it
- **Active/Inactive**: This indicates whether there was a problem accessing the destination

The FRA (called Flashback Recovery Area before Oracle 11g R2, and now called Fast Recovery Area) is a disk location in which the database can store and manage all files related to backup and recovery operations. Flashback database provides a very efficient mechanism to rollback any unwanted database change. We will talk in more depth about FRA and Flashback database in *Chapter 4, User Managed Backup and Recovery*. 

For More Information:  
Placing a database into the ARCHIVELOG mode

Now let’s take a look at a very popular example that you can use to place your database in the ARCHIVELOG mode, and use the FRA as a secondary location for the archive log files. To achieve all this you will need to:

1. Set up the size of your FRA to be used by your database. You can do this by using the command:
   ```sql
   SQL> ALTER SYSTEM SET DB_RECOVERY_FILE_DEST_SIZE=<M/G> SCOPE=both;
   ```

2. Specify the location of the FRA using the command:
   ```sql
   SQL> ALTER SYSTEM SET DB_RECOVERY_FILE_DEST='/u01/app/oracle/fast_recovery_area' scope=both;
   ```

3. Define your archive log destination area using the command:
   ```sql
   SQL> ALTER SYSTEM SET log_archive_dest_1= 'LOCATION=/DB/u02/backups/archivelog' scope=both;
   ```

4. Define your secondary archive log area to use the FRA with the command:
   ```sql
   SQL> ALTER SYSTEM SET log_archive_dest_10= 'LOCATION=USE_DB_RECOVERY_FILE_DEST';
   ```

5. Shutdown your database using the command:
   ```sql
   SQL> SHUTDOWN IMMEDIATE
   ```

6. Start your database in mount mode using the command:
   ```sql
   SQL> STARTUP MOUNT
   ```

7. Switch your database to use the ARCHIVELOG mode using the command:
   ```sql
   SQL> ALTER DATABASE ARCHIVELOG;
   ```

8. Then finally open your database using the command:
   ```sql
   SQL> ALTER DATABASE OPEN;
   ```

When in the ARCHIVELOG mode, you are able to make hot backups using RMAN. You are able to perform some user-managed backups using the `alter database begin backup` command (used to allow you to make a consistent backup of your entire database files). You may also use the `alter tablespace <Tablespace_Name> begin backup` command to make a backup of all datafiles associated to a tablespace.

For More Information:
Now that you know everything you are supposed to know about the ARCHIVELOG mode, let's take a deeper look into what is redo and why it is so important to the recoverability of our database.

**Differences between redo and undo**

Another common question relates to the difference between redo log entries and undo information saved as part of transaction management. While redo and undo data sound almost like they could be used for the same purpose, such is not the case. The following table spells out the differences:

<table>
<thead>
<tr>
<th></th>
<th>undo</th>
<th>redo</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Record of</strong></td>
<td>how to undo a change</td>
<td>how to reproduce a change</td>
</tr>
<tr>
<td><strong>Used for</strong></td>
<td>rollback, read-consistency</td>
<td>rolling forward database changes</td>
</tr>
<tr>
<td><strong>Stored in</strong></td>
<td>undo segments</td>
<td>redo log files</td>
</tr>
<tr>
<td><strong>Protect Against</strong></td>
<td>inconsistent reads in multiuser systems</td>
<td>data loss</td>
</tr>
</tbody>
</table>

In the end, an undo segment is just a segment like any other (such as a table, an index, a hash cluster, or a materialized view). The important point here is in the name, and the main rule you need to understand is that if you modify part of a segment (any segment, regardless of its type), you must generate redo so that the change can be recovered in the event of a media or instance failure. Therefore, if you modify the table EMPLOYEE, the changes made to the EMPLOYEE blocks are recorded in the redo log buffer, and consequently to the redo log files (and archive log files if running in the ARCHIVELOG mode). The changes made to EMPLOYEE also have to be recorded in UNDO because you might change your mind and want to rollback the transaction before issuing a commit to confirm the changes made. Therefore, the modification to the table EMPLOYEE causes entries to be made in an undo segment, but this is a modification to a segment as well. Therefore, the changes made to the undo segment also have to be recorded in the redo log buffer to protect your data integrity in case of a disaster.

If your database crashes and you need to restore a set of datafiles from five days ago, including those for the UNDO tablespace, Oracle will start reading from your archived redo, rolling the five day old files forward in time until they were four, then three, then two, then one. This will happen until the recovery process gets to the time where the only record of the changes to segments (any segment) was contained in the current online redo log file, and now that you have used the redo log entries to roll the data forward until all changes to all segments that had ever been recorded in the redo, have been applied. At this point, your undo segments have been repopulated and the database will start rolling back those transactions which were recorded in the redo log, but which weren't committed at the time of the database failure.

For More Information:

I can't emphasize enough, really, that undo segments are just slightly special tables. They're fundamentally not very different than any other tables in the database such as `EMPLOYEE` or `DEPARTMENT`, except that any new inserts into these tables can overwrite a previous record, which never happens to a table like `EMPLOYEE`, of course. If you generate undo when making an update to `EMPLOYEE`, you will consequently generate redo. This means that every time undo is generated, redo will also be generated (this is the key point to understand here).

Oracle Database stores the before and after image in redo because redo is written and generated sequentially and isn't cached for a long period of time in memory (as mentioned in the What is redo section in this chapter). Hence, using redo to rollback a mere mistake, or even a change of mind, while theoretically possible, would involve wading through huge amounts of redo sequentially, looking for the before image in a sea of changes made by different transactions, and all of these will be done by reading data off disk to memory as a normal recovery process. UNDO, on the other hand, is stored in the buffer cache (just as the table `EMPLOYEE` is stored in the buffer cache), so there's a good chance that reading the information needed will require only logical I/O and not physical. Your transaction will also be dynamically pointed to where it's written in UNDO, so you and your transaction can jump straight to where your UNDO is, without having to navigate through a sea of undo generated by all other transactions.

In summary, you need redo for recovery operations and undo for consistency in multiuser environments and to rollback any changes of mind. This in my personal opinion, is one of the key points that makes Oracle superior to any other database in the market. Other databases merely have transaction logs which serve both purposes, and suffer in performance and flexibility terms accordingly.

Facing excessive redo generation during an online backup?

One of the most common questions I see on the Oracle Technology Network (OTN) forums is why so much redo is generated during an online backup operation. When a tablespace is put in the backup mode, the redo generation behavior changes but there is not excessive redo generated. There is additional information logged into the online redo log file during a hot backup the first time a block is modified in a tablespace that is in the hot backup mode. In other words, as long as the tablespace is in the backup mode, Oracle will write the entire block to disk, but later it generates the same redo. This is done due as Oracle cannot guarantee that a block was not copied while it was being updated as part of the backup.
In the hot backup mode, only two things are different:

- The first time a block is changed in a datafile that is in the hot backup mode, the entire block is written to the redo log file, and not just the changed bytes. This is because you can get into a situation in which the process copying the datafile and the database writer (DBWR) are working on the same block simultaneously. Hence, the entire block image is logged so that during recovery, the block is totally rewritten from redo and is consistent with itself.

- The datafile headers which contain the **System Change Number (SCN)** of the last completed checkpoint are not updated while a file is in the hot backup mode. The DBWR process constantly writes to the datafiles during the hot backup. The SCN recorded in the header tells us how far back in the redo stream one needs to go to recover the file.

To limit the effect of this additional logging, you should ensure to place only one tablespace at a time in the backup mode and bring the tablespace out of the backup mode as soon as you have finished backing it up. This will reduce the number of blocks that may have to be logged to the least possible.

**Summary**

In this chapter, we refreshed some very important topics regarding backup and recovery basics. We started with understanding the purpose of backup and recovery, all the way to types of backups and the differences between redo and undo. If you are still a little confused about some of these topics, don't worry as most of them will be explained in more detail in the following chapters of this book.

In the next chapter, we will see in more depth why redo is so important to the recoverability of our data, why the NOLOGGING operations can be dangerous, and how to use them.
Where to buy this book

Free shipping to the US, UK, Europe and selected Asian countries. For more information, please read our shipping policy.

Alternatively, you can buy the book from Amazon, BN.com, Computer Manuals and most internet book retailers.

For More Information: