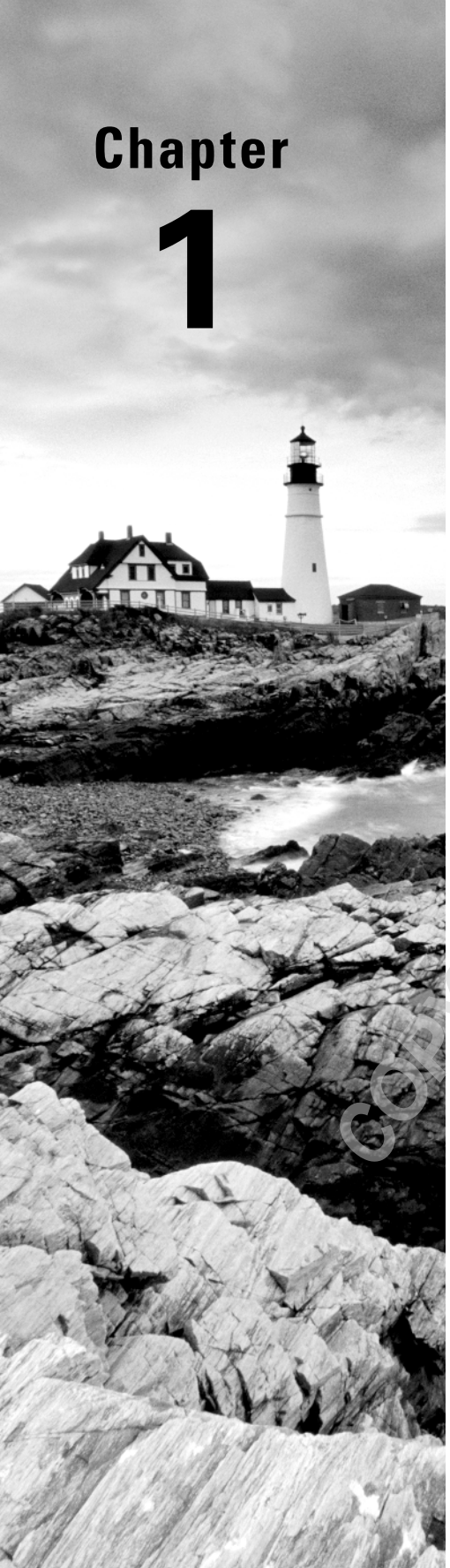


Chapter 1

Performing Oracle User-Managed Backups

ORACLE DATABASE 12*c*: ADVANCED ADMINISTRATION EXAM OBJECTIVES COVERED IN THIS CHAPTER:

- ✓ Explain Oracle backup and recovery solutions
- ✓ Describe types of database failures
- ✓ Describe the tools available for backup and recovery tasks
- ✓ Describe RMAN and maximum availability architecture
- ✓ Back up and recover a NOARCHIVELOG database
- ✓ Perform backup and recovery in NOARCHIVELOG mode
- ✓ Configure control files and redo log files for recoverability
- ✓ Multiplex control files
- ✓ Multiplex redo log files
- ✓ Describe and tune instance recovery
- ✓ Enable ARCHIVELOG mode
- ✓ Backup a control file to trace

This chapter begins your introduction to the information you need to pass your Oracle Database 12*c*: Certified Professional exam. Perhaps the main job of an Oracle DBA involves using your knowledge to ensure that your database is always backed up and recoverable.

This might sound simple. However, the expertise needed to ensure that your backup and recovery plan is effective is not easy to come by. This is one reason why the job of the DBA is in high demand. With your passing of the Oracle Database 12*c* Administration II exam, you will be able to present credentials to employers that indicate that you have a particular set of knowledge that will permit you to protect their data.

There is a lot to cover in this chapter. From the basics of how Oracle works and how that relates to backup and recovery, to configuring online redo log files and control file for higher availability. Also in this chapter, we will discuss manually backing up your database when it's in NOARCHIVELOG mode, and many more topics that you will want to know as you prepare for your Oracle OCP exam.

Exam objectives are subject to change at any time without prior notice and at Oracle's sole discretion. Please visit Oracle's Training and Certification website (`http://www.oracle.com/education/certification/`) for the most current exam-objectives listing.

Oracle Database Data Protection Options

There are a number of things to consider with respect to backup and recovery. There are different kinds of failures, different types of backups, and different tools that can be used to back up and restore an Oracle Database. In this section, we will look at the various Oracle Data protection options that are available to the DBA.

What Kind of Failures Can Happen to a Database

There are literally numberless combinations of things that can go wrong with an Oracle database. Like any technology, there are always opportunities for failure. Throughout this book you will see a number of examples of these types of failures. The most common failures include:

Physical failures – This includes the loss of disk drives, controller cards, memory, or even the server itself.

Logical failures – this includes failures that corrupt data, failures of software (such as bugs) and application failures corrupt data.

Networking failures – These are errors where one component cannot talk to another component. This can lead to a number of problems.

User errors – These are errors where the user accidently or purposely corrupts data.

Disasters – Things like earthquakes, tornadoes, and flooding happen from time to time and you have to be prepared for failures on a more massive scale.

When considering an overall backup and recovery architecture, you need to consider each of these kinds of failures and how to recover from them. Each type of failure requires a different approach when trying to protect from and mitigate the results of that failure. We will discuss these things in a number of places in this book.

Physical and Logical Backups

There are two different kinds of Oracle backup and recovery. The first is a physical backup and the second is a logical backup. A physical backup involves backing up the physical files of the database to some backup medium, such as another disk or a tape drive.

The logical backup extracts the data from the database and stores it in a separate backup file. Thus, the physical component of the database is not backed up, only the data and meta data needed to reconstruct the database if the logical backup should need to be used to restore the database.

In general the logical backup is of less use than a physical backup. This is because the recovery variations available with physical backups are generally greater than with logical backups. Another benefit of physical backups over logical backups is that large logical backups will usually take longer to restore than a physical backup from the same database.

Other benefits of physical backups are that they can be done in a consistent manner and they provide the ability to restore the database to any given point in time that is required. Logical backups do not provide this point-in-time recovery functionality.

Tools for Backup and Recovery

There are different ways of performing backup and recovery of an Oracle database. The most common tools that you will find in use are:

RMAN – Perhaps the most commonly used tool, RMAN is a free tool provided by Oracle that performs physical backups of the Oracle database. RMAN is easy to implement and use.

Oracle Data Pump – Oracle Data Pump is a method of performing logical backups of the Oracle Database.

Oracle Flashback features – Oracle Database provides a number of features that revolve around its Flashback Database functionality. These flashback features include the following:

- Flashback Database
- Flashback Query
- Flashback Version Query
- Flashback Transaction Query

- Flashback Transaction
- Flashback table
- Flashback table drop

As you can see there are a number of different variations on the Flashback features, specific to different kinds of needs.

Manual backups – You can write scripts that perform manual backups of your database. You can then take those manual backups and manually restore them.

Oracle MAA Recommendations

The Oracle database provides a number of ways to protect the enterprise from data loss. The different kinds of protection range from basic offline backup and recovery procedures to more complex functionality to provide disaster recovery and multisite solutions.

Oracle provides guidance with respect to data protection in a series of recommended best practices collectively called the Maximum Availability Architecture (MAA) best practices. These best practices describe the architectures, processes, standards, and procedures that should be followed to ensure that your database is protected to the maximum extent possible. The MAA practices actually extend to the various layers in the Enterprise computing landscape, including application and infrastructure.

Various Oracle database features and products are part of the MAA best practices including the following and many other topics:

- Backup and recovery
- High availability
- Disaster recovery
- Data replication and distribution
- Database rolling upgrades

As you will see in the next several chapters, the Oracle Database has a rich set of features to protect your database from data loss. This book concentrates a great deal on Oracle Database backup and recovery. Likewise, the OCP exam contains a large number of questions related to database backup and recovery, specifically those associated with RMAN. Backup and recovery is one of the most fundamental architectural features of the Oracle Database with respect to data protection.

However, there are other things that need to be considered when planning your enterprise data-protection strategy. These include topics such as high availability, disaster recovery, how to correct user or application data corruptions, and data replication. Solutions to these problems involve the following Oracle Database features and products:

Oracle Real-Application Clusters High availability is provided through the use of Oracle Real-Application Clusters (RAC). With RAC you can have more than one computer connected to your database. Thus, if one of the computers fails, the remaining computers can continue to run.

Oracle Data Guard Disaster recovery is provided through the Oracle Data Guard product. Oracle Data Guard makes it possible to duplicate your database to one or more sites and keep those duplicate databases synchronized with your primary database. When the primary database site fails, you can switch over operations to one of the Oracle Data Guard databases with minimum or zero data loss. The Oracle OCP exam does not contain any questions specific to Oracle Data Guard.

Oracle Flashback Database Accidental data corruption and data loss can be rectified through the use of Oracle Flashback Database. Oracle Flashback Database is a subject of the OCP exam and is covered in Chapter 8, "Understanding Flashback Technology," of this book.

Oracle GoldenGate Data replication is provided through the Oracle GoldenGate product. This product provides the ability to replicate data between both Oracle databases and non-Oracle databases. The OCP exam does not have any questions on Oracle GoldenGate.

Understanding the Oracle Database as It Relates to Backup and Recovery

As a DBA, recovering your database should be important to you. Correspondingly, recovery is also important to Oracle, so the database product has been built to be robust with respect to backup and recovery. We'll start this chapter with a quick primer on how Oracle supports backup and recovery. In this section, we'll give you the background you need to understand backup and recovery and to be successful with your OCP exam. In the following sections, we will discuss these topics:

- Oracle processes related to backup and recovery
- Oracle memory structures related to backup and recovery
- The data dictionary
- Oracle data files and tablespaces
- Online redo logs
- Control files
- Parameter files
- NOARCHIVELOG and ARCHIVELOG modes
- The Oracle instance and the Oracle Database

Note this is not the "kitchen sink" when it comes to an Oracle architecture discussion. We assume you are already somewhat familiar with the Oracle Database architecture (since to take the OCP exam you must first have completed the OCA certification track, which consists of two exams), so this is just a review of the pieces of it that are involved in backup and recovery in some way.

There are two different kinds of Oracle recoveries: instance/crash recovery and media recovery. *Instance recovery* is automatically managed by Oracle when you restart the database. *Media recovery* is a manual process done by the DBA and involves the use of Oracle Database backups.

Oracle Processes Related to Backup and Recovery

The front-line support for Oracle backup and recovery is the Oracle architecture. One part of this architecture is the processes related to the Oracle Database. Although the Oracle Database has a number of processes, only a few really matter with respect to backup and recovery and will be mentioned in this text. These processes are as follows:

- LGWR
- DBW
- ARCH
- Multithreaded Oracle
- User processes

Let's discuss each of these processes next so you can better understand how they impact database recovery.

LGWR Process

The *log writer process (LGWR)* is responsible for keeping the online redo logs up to date. The job of the LGWR process is to move redo from the volatile (nonpersistent) redo log buffer in the System (sometimes called Shared) Global Area (SGA) to the persistence of the online redo logs. A number of different things cause LGWR to wake up and write the redo data, among them are database session commit operations and when the redo log buffer fills to a certain point.

DBW Processes

The *database writer processes (DBWn)* (the *n* indicates that there might be more than one of these processes) are responsible for writing to the database data files. This writing occurs during events called *checkpoints*. A database checkpoint may, in reality, happen at just about any time while the database is running. DBW has very little to do with recovery of the database (other than to support it by writing to the data files) because database data file writes are often delayed and the blocks within the data files themselves are not consistent with the current state of the data inside of the SGA. By default, Oracle starts only one DBW process, but if needed, additional DBW processes can be utilized, though this is rare.

ARCH Processes

The *archiver processes (ARCn)* are responsible for the creation of archived redo logs. In a later section in this chapter, we will discuss how redo logs are filled with redo. Once the redo log file

fills, a log switch occurs and Oracle will begin to write to the next online redo log. If the database is in ARCHIVELOG mode (see the section "NOARCHIVELOG and ARCHIVELOG Modes"), the ARCH process will be responsible for taking that filled archived redo log and copying it to one or more backup locations.

If the database is in ARCHIVELOG mode, then the ARCH process will start automatically. Oracle can also start more than one ARCH process if multiple redo logs need to be archived. For ARCH to work properly, you will need to configure the appropriate archiving locations (see "Configuring the Database for Backup and Recovery" later in this chapter for more information). The ARCH process is so vital to backup and recovery of the database that if it cannot copy the archived redo logs to the mandatory archived log destinations, the database will eventually stall until the problem is corrected.

Database Architecture

You might wonder why we spend time talking about the architecture of the database. We don't want you to just understand *how* to do something; we want you to understand *why* you do it. Understanding backup and recovery requires knowing the architecture of the Oracle Database. This knowledge may well make the difference when you face a question and you are not sure of the answer. If you understand how something works, then figuring out an answer to a question becomes easier because you also know how it does not work.

If you are unsure of an answer to an OCP exam question, try to work out which answers are not correct and eliminate them. By knowing how the architecture works, you will be able to eliminate questions more easily and find the correct answer more often.

Multithreaded Oracle

Oracle Database 12*c* introduced an optional multithreaded process/threaded model for operating systems such as Unix and Linux. In this model, most of the database background processes are spawned as threads of a few primary processes. The Oracle multithreaded model should not be confused with Oracle Shared Servers. Shared Servers is a user connection pooling mechanism, whereas multithreaded Oracle is a different way of spawning the processes that make Oracle run.

You can enable multithreaded Oracle through the use of the `THREADED_EXECUTION` parameter. When this parameter is set to `TRUE`, there will no longer be a one-to-one relationship between an Oracle background process and an operating system process.

User Processes

User processes are the processes that are started by client programs to connect to the database. In the case of local user processes (also called foreground processes to distinguish them from background database instance processes) on the database server, they

will connect directly to the database without the need of a listener. User processes coming from the network will use the Oracle listener to establish a connection to the database. The listener is a program that "listens" for clients to make database connections on a specific port. The listener then spawns a foreground process and connects the network connection to that spawned process.

At first glance, it might seem that the user processes are not all that important to backup and recovery. As you will see, user processes are actually an integral part of backup and recovery since you have to be able to connect to the database instance with these processes. You will also find as you proceed through this book that you will need the listener up and running in order to perform certain functions.

Oracle Memory Structures Related to Backup and Recovery

The principle SGA memory structure to be aware of when it comes to backup and recovery is the *redo log buffer*. This is typically a small area of memory that is configured for Oracle to store redo records in. This is a very transient area of memory and its size can impact the performance of your database. Although the redo log buffer will not have a direct impact on backup and recovery, it's still important to be aware of it in the light of any discussion on backup and recovery.

The Oracle Data Dictionary

The *Oracle data dictionary* is a critical piece of the backup and recovery landscape. In the following sections, we will introduce you to the data dictionary. We will then give you some information on the basic format of the data dictionary so it will be more familiar to you when you actually use it. Finally, we will provide a list of views that you will find useful during your backup and recovery efforts.

Overview of the Data Dictionary

The data dictionary is a set of views and tables that expose metadata about your Oracle Database. For example, if you want to know the name of your database, you can look at the `NAME` column in a view called `V$DATABASE`.

The data dictionary is important because it will give you information on the following critical components of the database:

- Tablespaces
- Data files
- Redo logs
- RMAN backup–related information
- Database configuration

This information will be essential when configuring for backups and also when you need to recover your database from failure. Throughout this book you will be using the data dictionary, and it behooves you to become comfortable with it.

Forms of the Data Dictionary

The data dictionary views are named using a common naming convention. This convention can be used to identify the source of the data and when a view can be queried. The main types of data dictionary views are as follows:

We are introducing the CDB_* views here. We will not be covering the Multitenant Database option of the Oracle Database in detail in this chapter. Please see Chapter 12, "Managing Oracle Multitenant Databases," for more details on this option.

Static Data Dictionary Views The *static data dictionary views* are sourced from tables and views created when the database was first created. These tables and views are owned by the SYS schema and are located in the SYSTEM tablespace. The views typically contain structural metadata about the database, including such things as tables, indexes, and other database objects.

If you are running Oracle Multitenant, you will find versions of these views in each pluggable database (PDB) contained within the container database (CDB). Also you will find versions of these views in the root container. The scope of what is contained in these views varies depending on the PDB you are in, the user you are connected as, and the state of the database. Please see the chapters on Oracle multitenant databases for more information on what is contained in these views when using Oracle Multitenant.

The names of these views are all prefixed to indicate the scope of the data contained within that view. There are four main prefixes:

DBA_* The DBA_* views allow those with DBA privileges to see all data contained in the view. For example, if you were a DBA and you wanted to see all tables in the database named MY_DATA, you could query the DBA_TABLES view, as shown here:

```
Select owner, table_name from dba_tables
Where owner='MY_DATA';
```

ALL_* The scope of the ALL_* views is more reduced than that of the DBA_* views. When you query the ALL_* views, you see only those objects for which you have been granted some form of access. For example, if you wanted to see all instances of a table called MY_DATA that you had access to, you could query the ALL_TABLES view, as shown here:

```
Select owner, table_name from all_tables;
```

USER_* The USER_* views are the most restrictive of the data dictionary views. When you query the USER_* views, you see only those objects that are in the schema you are

currently logged into. For example, if you wanted to see if there was a table called MY_DATA in your schema, you could query the USER_TABLES view, as shown here:

```
Select table_name from user_tables;
```

CDB_* With the advent of Oracle Database 12*c*, Oracle has added a completely new set of data dictionary views. These views help you to deal with Oracle multitenant databases. The views are prefixed with CDB, and they allow you to look at the various pluggable databases within the Oracle container database. We will cover these views in a lot more detail in chapter 12.

Notice in the example query against USER_TABLES that we removed the owner column. It is quite common that the DBA_* and ALL_* views will have an owner column but that the USER_* views will not. This is because the user in the USER_* views is assumed to be the user you are logged in as.

Dynamic Performance Data Dictionary Views The *dynamic performance data dictionary views* typically start with a V$ prefix, such as V$DATABASE or V$SESSION. The views are often used for database monitoring and tuning, but there are times when they will be the only database views available to you for recovery purposes.

The data in these views source from either the database control file or C structures that are part of the Oracle Database kernel. Typically these views are available when the database instance is mounted (see "Oracle Database Startup and Shutdown" later in this chapter), but some views are available only after the instance has started.

Real World Scenario

Using Data Dictionary Views

In the real world, DBAs use the data dictionary a great deal. Although Oracle offers a nice graphical database administration tool called Oracle Enterprise Manager Cloud Control 12*c* that helps reduce the DBAs' need to use the data dictionary, the typical DBA will often need to access the data dictionary.

For example, recently we needed to know which users were on the system and what their OS process IDs were so we could kill a process. We were already in SQL*Plus, and it was easier to just query the data dictionary than to open a browser, log into the OEM, and surf to the page that would give us the information we wanted.

Our boss thought we were crazy for not using OEM, so we had a race to see who could get the information faster. Want to guess who won? Often DBAs will create their own set of scripts to access the data dictionary views. This makes it even faster to get the information you want without having to switch back and forth to OEM. OEM is a great tool, but sometimes it just pays to know the data dictionary. Plus, there are some places that don't use it at all.

Common Data Dictionary and Dynamic Performance Views You Will Use

During your backup and recovery experiences as an Oracle DBA, you will have occasion to use the data dictionary. Table 1.1 provides a list of some data dictionary views that you will want to be aware of and that will be helpful to you on your OCP exam.

TABLE 1.1 Examples of Data Dictionary/Dynamic Performance Views Useful for Recovery

View Name	Description
V$DATABASE	Provides basic database-related information, including the logging mode
V$INSTANCE	Provides basic instance information
V$DATAFILE	Provides database data file information stored in the control file
V$LOGFILE	Provides information on the individual redo log file members from the control file
V$LOG	Provides information on the redo log groups from the control file
V$ARCHIVED_LOG	Provides archive log information from the control file
V$LOG_HISTORY	Provides information on redo log switches in the database
DBA_DATA_FILES	Provides data file information
DBA_TABLESPACES	Provides information on tablespaces in the database

You can find all the data dictionary views available in Oracle by looking at the Oracle reference guide in the Oracle documentation available at http://tahiti.oracle.com.

Oracle Data Files and Tablespaces

Oracle data is stored in tablespaces, which comprise one or more data files. It is very important to understand data files and tablespaces and their relationships when it comes to backup and recovery. In the following sections, we will briefly reintroduce you to the important components of the Oracle Database.

Oracle Data Files

The Oracle Database *data files* are critical database physical files that are used to store all your Oracle data (well, almost all!). The physical data files are the structures most likely to be lost and subsequently recovered during a database-recovery operation. You are probably aware that database data files are preallocated in size and that they can be configured to grow automatically.

When you perform a physical backup of your database, the database data files will be the principle structures you back up. When you restore your database due to a media failure, you will be restoring one or more data files. You may also have to restore other database files such as the control file, the online redo logs, and the archived redo logs, which we will discuss later in this section.

Another type of database backup is called a logical backup. We will discuss logical backups in future chapters.

Data files can be in two different states. They can be online (the default) or offline. Some Oracle database restore operations will require that you take one or more data files offline.

Oracle Tablespaces

A *tablespace* is a logical, named entity in the database that is used to store database objects. For example, your database might have a table called `STORES` that contains data about stores in your organization. The table `STORES` will be assigned to a tablespace, perhaps called `STORE_DATA`.

A tablespace is assigned to one or more database data files, and the size of the tablespace is related to the size of the underlying database data files. Several recovery options exist with respect to tablespaces, including tablespace point-in-time recovery, which we will discuss in later chapters of this book.

Tablespaces can be in four different states. They can be online (the default), offline, read-write (the default), or read-only. If you put your tablespace in read-only mode, then you will need to back up the data files associated with that tablespace only once as long as the tablespace is in read-only mode. Some recovery operations require the database be offline to perform restore and recovery operations.

Redo Logs

The redo logs of the database are the principle vehicle for backup and recovery. In the following sections, we will cover two different types of redo logs. First, we will discuss the online redo log, and then we will discuss the archived redo log.

Online Redo Logs

Online redo logs are used by Oracle to store all the changes that occur in the database. Think of them as something akin to a videotape recording of everything that is going on.

Later on, you can rewind that videotape and replay it to see what happened. During recovery, Oracle does just that, using the online redo logs.

In this section we will discuss online redo log file basics, redo log file groups, redo log file members, and redo log file sequence numbers.

Redo Log File Basics

Online redo logs are created when the database is created. You must have a minimum of two redo logs in any Oracle database. When the database starts running, it will write to the first redo log group. Once that log fills up, it will switch over and begin to write to the next log. Once the second log fills up, the database will clear the first log and begin to write redo into that log. Thus, redo log files are used in a round-robin style.

Redo Log File Groups

Each online redo log file is a member of a specific *redo log file group*. In the case where there are just two online redo logs (the minimum allowed), there will be two groups, likely named group 1 and group 2. You can create new online redo log groups and drop existing groups (until you are down to just two groups) anytime online.

Redo Log File Members

As you might gather, the online redo log files are an important component of the database. If you lose the online log files, you could permanently lose data. To protect against this, Oracle provides for multiplexing of online redo logs within each group. Each copy of the online redo log is considered a *redo log file member*, often called just a member.

Multiplexing allows you to indicate to Oracle that it should create and maintain duplicate copies of the online redo log files. When multiplexing online redo logs, it's a good idea to put each member on a different disk for many reasons, even on a SAN.

Redo Log Sequence Numbers

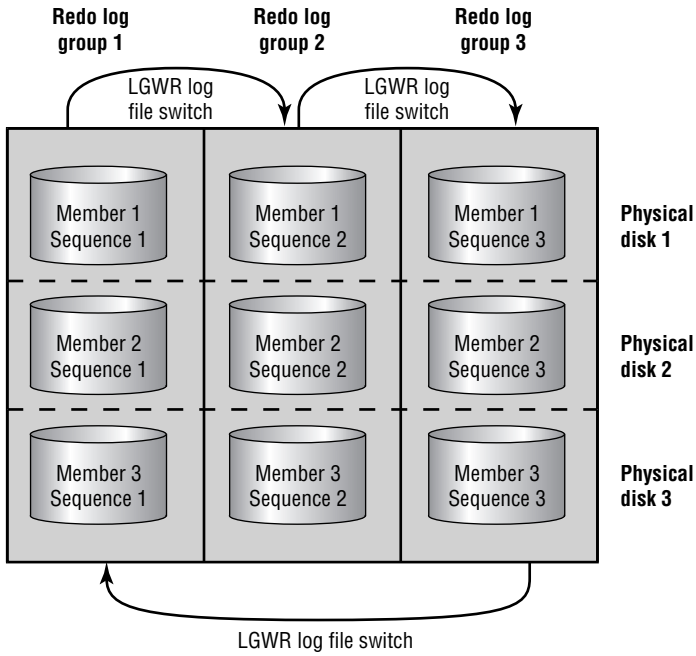
Each time an online redo log group is used, that group is assigned a unique *redo log sequence number* (typically 1 for a new database). As you can see in Figure 1.1, you have two online redo log groups. You start writing to the first online redo log group, which is log sequence 1. Once that log group fills up, you start writing to online group 2, which was assigned sequence number 2. Once that group fills up, you switch back to redo log group 1, reusing that redo log group. The sequence number is incremented, though, and the redo associated with that online redo log group will be part of sequence 3. Once that group fills up, Oracle will switch to online log group 3. Once that group fills up, Oracle switches back to redo log group 1, reusing that redo log group.

Sequence numbers can be very important when it comes to recovering your database, as you will see in later chapters.

Archived Redo Logs

You may have noticed from the previous discussion on online redo logs that the redo log files get reused over and over. As a result, the records in those log files will be lost forever when the log file is reused. This overwriting of the online redo logs limits the recovery options available for you to use with Oracle.

FIGURE 1.1 Redo log file round-robin writing



When the database is put in ARCHIVELOG mode (see the section "NOARCHIVELOG and ARCHIVELOG Modes"), Oracle will make copies of the online redo logs after they have been filled and after Oracle starts to write to the next online redo log group. The copies of the online redo logs are called *archived redo logs*. Archived redo logs are critical to advanced recoveries such as point-in-time recoveries and point-of-failure recoveries.

To protect the database transactions, online redo logs will not be reused until the archived redo logs have been successfully written. This can cause database operations to be suspended if the archived redo logs cannot be written successfully.

Control File

The *control file* of the database is kind of a central repository of important database related metadata. It's a binary file that contains information about physical database structures, redo logs, and archived redo logs; RMAN information is stored here too. The control file is critical to a good backup and recovery strategy, as you will see in this chapter and other chapters of this book.

Parameter Files

In this section, we will address what parameter files and parameters are. We will then discuss the two types of parameter files available in Oracle: the parameter file (pfile) and server parameter file (spfile).

Parameter Files and Parameters

Every Oracle Database has a *parameter file*. Inside the parameter file you'll find a variety of parameters that define global settings for that database. Parameter files are stored, by default, in `ORACLE_HOME/dbs` or `ORACLE_HOME/database`, depending on the operating system in use.

Each parameter file will contain many different parameters. Parameters are used to configure memory settings, auditing settings, destination directories for log files, and archived redo log files. When configuring a database for backup and recovery, you will configure several parameters, as you will see in the section "Configuring the Database for Backup and Recovery."

Parameters come in two flavors, static and dynamic. You must change static parameters and restart the database in order for the parameters to take effect. Dynamic parameters can be changed on the fly, without the need to restart the database.

You can find all the database parameters available in Oracle by looking at the Oracle reference guide in the Oracle documentation available at `http://tahiti.oracle.com`.

Parameter Files

The pfile is a text-based parameter file. To modify settings in this file, you simply open the file with your editor and change it. Once your changes are complete, save the file. You will then need to bounce the database to have those settings take effect.

Pfiles are kept, by default, in the `$ORACLE_HOME/dbs` (Unix) or `%ORACLE_HOME%\database` (Windows) directory. The default name for a pfile is `init<oracle_sid>.ora`. So if your Unix database is called `ORCL`, your pfile will be called `initORCL.ora` by default. Note that case sensitivity applies here based on the operating system.

Server Parameter Files

The spfile is an Oracle database managed parameter file managed by the Oracle server. We will first look closer at the spfile itself, and then we will discuss how to set parameters when using an spfile.

What Is an Spfile?

An Oracle spfile differs from a text-based parameter file in that the Oracle server manages it, and you as the DBA should never edit the file directly. The spfile is partly nontext (it has a header and a footer), but the actual parameter settings are in plain text (so you can view the file if you like and see what parameters are set to in the spfile).

Spfiles are kept, by default, in the `$ORACLE_HOME/dbs` (Unix) or `%ORACLE_HOME%\database` (Windows) directory. The spfile naming convention is `sp<oracle_sid>.ora` by default. When the DBA starts the Oracle database/instance, Oracle will first look for an spfile using the default filename. If one is not found, it will look for a file called `spfile<oracle_sid>.ora` in the same default directory. Finally, if no spfile is found, Oracle will look for a regular pfile as described previously. If no parameter file is found, then the database will signal an error and the startup will abort. Oracle generally recommends using an SPFILE over a text parameter file.

Note that if you try to start the database from RMAN without an spfile or pfile available, Oracle will use the default parameter settings and actually start the database. This does not happen if you try to start the database from SQL*Plus or OEM. We will discuss RMAN more in Chapter 3, "Configuring and Backing Up Using RMAN."

How Do You Set Parameter Values When Using an Spfile?

To modify a parameter when using an spfile, you use the `alter system` command. For example, if you want to change the parameter `DB_RECOVERY_FILE_DEST_SIZE`, which is a dynamic parameter (so it can be changed on the fly), you would issue the following `alter system` command:

```
Alter system set db_recovery_file_dest_size=100m;
```

The previous command only changes the parameter as long as the database instance is running. The parameter would be reset in this case unless you used the scope=both parameter. In this case, the parameter would be changed for the running instance, dynamically, and also the parameter would be changed in the database spfile so that the change would persist through an instance restart. Here is an example:

```
Alter system set db_recovery_file_dest_size=100m scope=both;
```

As we said earlier, some parameters are not dynamic. In this case, you have to indicate that you want to change only the parameter file. To do this, use the `alter system` command and include the `scope=spfile` keyword, as shown here:

```
Alter system set memory_max_target=200m scope=spfile;
```

In some cases, you may want to change the parameter in just the current instance of the database, but you will not want that change to persist after the next shutdown. In this case, use the `scope=memory` keyword when issuing the `alter system` command, as shown here:

```
Alter system set db_recovery_file_dest_size=100m scope=memory;
```

NOARCHIVELOG and ARCHIVELOG Modes

Oracle Database runs in two principal modes, NOARCHIVELOG (the default) and ARCHIVELOG. The logging bit has to do with archived redo logs and if they are saved or not, which makes a difference in the kinds of recoveries that you can do. Let's look at each mode in a bit more detail.

For manual backup and recovery questions in the OCP exam you will only have to be concerned with databases in NOARCHIVELOG mode. When you are asked RMAN questions you may be asked about either ARCHIVELOG mode or NOARCHIVELOG mode. So we cover both modes here.

NOARCHIVELOG MODE

NOARCHIVELOG mode is the default logging mode. In NOARCHIVELOG mode, the online redo logs are overwritten over time and no backups are created. Because of this, you are limited in the way you can back up your database and how you can recover it.

Backups are limited to cold or offline backups. This means that you must shut down your database before you can back it up. As you will see in later sections of this chapter, you will back up all the data files of the database plus the online redo logs and the control file(s).

Recovery in NOARCHIVELOG is equally limited. In NOARCHIVELOG mode, you can restore the database only to the point in time that the backup was taken. Thus you will lose any changes to the database that took place after the backup was complete and the database was opened for business. This is typically not an acceptable solution for production databases.

We will discuss how to back up your database in NOARCHIVELOG mode later in this chapter. In Chapter 2, "Performing Oracle User-Managed Database Recoveries," we will discuss recovering your database with backups taken in NOARCHIVELOG mode. In Chapter 3 we will discuss using RMAN to perform offline backups, and in Chapter 5, "Recovering Databases with RMAN," we will discuss using RMAN to restore these backups.

ARCHIVELOG MODE

ARCHIVELOG mode is a much more flexible method of operating your database and is strongly recommended for all production databases. In this mode, you can back up your database while it's up and running, allowing users to work at the same time that the backups are running.

When the database is in ARCHIVELOG mode, changes are recorded in the online redo logs as usual. What is different is that the archived redo logs are copied to a backup directory once they have filled up. These copies of the redo log files are called *archived redo logs* and the Oracle Database process that copies them is called the *ARCH process*. In Oracle Database 12*c*, the ARC*n* process starts automatically when the database is in ARCHIVELOG mode.

The archived redo logs may be copied to one or a number of different destinations. We will discuss configuring where Oracle should copy these archived redo logs to in the section "Configuring the Database for Backup and Recovery."

We will discuss how to back up your database in ARCHIVELOG mode later in this chapter. In Chapter 2 we will discuss recovering your database with backups taken in ARCHIVELOG mode. In Chapter 3 we will discuss using RMAN to perform online backups, and in Chapter 5 we will discuss using RMAN to restore these backups. Finally, in Chapter 7, "Performing Oracle Advanced Recovery," we will discuss more-advanced recoveries with backups taken in ARCHIVELOG mode.

The Oracle Instance and the Oracle Database

In the following sections, we will first review knowledge you should already have on the basics of the Oracle instances and the Oracle Database. Then we will discuss the startup and shutdown of the Oracle instance and the Oracle Database.

Oracle Instances and Oracle Databases: a Review

With respect to backup and recovery of Oracle databases, it is important to understand that there is a difference between an Oracle instance and an Oracle database. This is because certain backup and recovery operations must occur while the Oracle instance is running and the database is not. Other operations will require that the database be open (which presumes the instance is already running).

No doubt you are already somewhat familiar with the notion of the instance from your OCA experience as well as your actual database experience. To review, an Oracle *instance* is the collection of shared memory (SGA) and processes (LGWR, DBWR, and so on). When these are all up and running, the instance is said to be started (see the next section for more details on starting and stopping a database).

The Oracle *database* is essentially the collection of database data files, the control file, and online redo log files. When the instance is running, Oracle will attach to the database and open it for business. Once the database is open, users can begin to do their work, assuming there are no problems during the database open process that prevents it from opening.

Sybex's *OCA: Oracle Database 12c Administrator Certified Associate Study Guide* provides complete coverage of the Oracle Database memory and processes. In this text, we assume you have already taken and passed the OCA exam and that you understand the body of knowledge consanguineous to that exam.

Oracle Database Startup and Shutdown

It is important to understand the startup and shutdown process of an Oracle database. In this section, we will first discuss the different stages of the Oracle database startup and

shutdown processes. We will then discuss the mechanics of actually starting and stopping the database in its different phases. We will quickly discuss restricted-mode database operations, and finally we will discuss the different stages the database must be in for specific types of backup and recovery operations to occur.

Exploring the Stages of Database Startup and Shutdown

When an Oracle database is started, it goes through four different and distinct stages:

Shutdown When the database and instance are shut down, they are at rest. There are no processes present, no memory is allocated, and nothing is going on. It is important to note that even though the database/instance may be shut down and closed, other Oracle processes (like the listener or OEM agents) may be still running.

Nomount When the database is in nomount mode, the instance has been started. Thus, processes have been started and memory allocated.

Mount When the database is in mount mode (or mounted), the instance is started and the database has opened the control file. The control file is read, but its contents are not validated. Note that the database is not open at this time.

Open When the database is opened, the control file contents have been validated against the physical database. The data files are all confirmed to be present, and they are opened. Oracle will then analyze the data files to determine if the database is in a consistent state. If the database is not in a consistent state, some form of recovery will be required.

Typically, the form of recovery required, *crash or instance recovery*, does not require any DBA involvement. If instance recovery is not possible, then *media recovery* is required. Media recovery requires the application of backups and recovery operations to bring the database current to the point of failure (if this is possible). The principal determining factors for media recovery are the presence of the needed data files and the availability in the online redo logs of the redo needed to bring those files current. If either of these conditions does not exist, then media recovery is required.

Database shutdowns occur in much the same way as startups, except in reverse. There are two different kinds of shutdowns, however: consistent and inconsistent.

Consistent Shutdowns If your database shutdown is a *consistent shutdown*, then the database data files and the database control file will be synchronized upon shutdown. The dirty buffers in the database buffer cache will be flushed out to the database data files, making them consistent. A consistent shutdown is a nice, tidy shutdown.

Inconsistent Shutdowns An *inconsistent shutdown* is another term for a mess. When your database is shut down in an inconsistent manner, it is in an indeterminate state and will require some form of recovery (typically instance recovery, which requires no DBA intervention) when it is restarted. Inconsistent shutdowns, however bad they might sound, often are the only way to shut down a database in a timely manner.

About Instance Recovery

We mentioned instance recovery earlier, and there may be a question or two about instance recovery on your OCP exam, so let's talk about that for a second. In this section, we will discuss:

- Checkpoints
- The system change number
- Instance recovery
- Fast-start fault recovery
- Tuning database instance recovery times
- Monitoring cache recovery
- The mean time to recover advisor

Checkpoints The reason there is a variance between the current state of the database and the state of the database as represented in the database data files is that Oracle reduces the writes to disk in order to improve database performance. The DBW process typically writes to the database data files in a lazy fashion. This is so that the IO associated with the writes by the DBW processes will be minimized. Each time a write occurs, this is called a checkpoint.

Checkpoints occur irregularly most of the time, triggered by various events. It is quite possible that a changed block in memory will not be written to disk for a period of seconds or even minutes after a commit occurs. Thus, a commit does not cause a checkpoint. A commit only causes the redo change records (called change vectors) that have been generated in the redo log buffer to be flushed to the online redo log of the database. Since the online redo logs reside on disk, they provide the persistent image that, when added to the database data files, represent the most current image of the database to the most recent commit.

The System Change Number Oracle keeps track of all of the activity in the database through the use of the system change number (SCN). The SCN is a counter, and its job is to keep track of everything going on inside the database and assign it a temporal identity. This serves to keep transactions that occurred in a particular order in the same order later down the road (such as during recovery). You need to preserve the order of transactions because of dependencies that occur between transactions. For example, if you have a parent table and a child table, you want to make sure that during recovery all `inserts` into the parent table occur before `inserts` into the child table. This is because of the foreign key constraint that exists between the two tables to ensure the integrity of that parent/child relationship. The SCN helps Oracle to track the temporal flow of those changes, and thus the parent table `insert` will have a lower SCN than the child table `insert`. As a result, in the end, all is right with the world.

SCNs are loosely coupled with time. Thus, 12:30 p.m. local time would be associated with a specific SCN in a given individual database. The thing to remember is that 12:30 p.m. local time will most likely be associated with a different SCN in each database, so the coupling is very loose. The concept of the SCN is very important because there may be times when you

will want to restore your database to a specific SCN. This is supported during recovery operations. Also, Oracle's Flashback features support the use of the SCN when flashing back the database. See Chapter 8 for more information on the vast number of features available with Oracle Flashback Database.

Instance Recovery There are two different flavors of instance recovery: instance recovery and crash recovery. *Instance recovery* is associated with the crash of the entire database, including all of its instances, and the recovery of the database after that crash. *Crash recovery* occurs when a RAC database node fails, and the other nodes recover the transactions of the failed node. In the context of the Oracle OCP exam, you will need to understand instance recovery and how to tune the database to reduce the time it takes to do instance recovery. You will not need to be concerned with crash recovery.

Understanding and controlling the time it takes to perform instance recovery is important because your customers may require that you adhere to specific requirements with respect to how long it takes your database to open after it has crashed (say, for example, because of power loss). The maximum time allowed for your database to come back up is typically called the recovery time objective (RTO).

When starting the database, Oracle will check the SCN in each individual data file against what is called the checkpoint SCN in the database control file. The checkpoint SCN is a number that represents the SCN recorded after the completion of the last successful checkpoint.

If the SCNs are all the same when the database is started, then no instance recovery is required and the database is opened. This is the quickest way to get an Oracle database open.

If the SCNs in the data files differ from the checkpoint SCN in the control file, then instance recovery will be required. Instance recovery occurs in two stages. The first is cache recovery, where Oracle will apply the changes from the online redo logs, including any that might be recorded in the online redo logs that are not committed. Cache recovery is the portion of instance recovery that you need to tune because this is the process that is keeping the database from opening.

Once cache recovery is complete, Oracle will open the database and start the second stage of instance recovery, called transaction recovery. During transaction recovery, Oracle will roll back any uncommitted transactions that were applied during cache recovery. However, the database is open and fully functional.

As you might expect, as the version of the data on disk diverges from the version of the data at the current point in time, the time to perform instance recovery increases. While this improves performance at run time, it can also negatively impact performance during instance recovery. This balancing act between instance recovery and database performance is managed by Oracle's Fast-Start Fault Recovery feature, which we will discuss next.

Fast-Start Fault Recovery Fast-Start Fault Recovery provides ways to adjust DBW so that it will maintain write IO at a rate sufficient to maintain a specific target RTO while also enabling maximum database performance. Thus, you balance the frequency of DBW writing against the time it takes to do instance recovery.

Fast-Start Fault Recovery manages the recovery time by managing the data latency between the persistent storage (disk) and the database cache in memory. You control Fast-Start Fault Recovery through the use of the FAST_START_MTTR_RECOVERY parameter.

The FAST_START_MTTR_RECOVERY parameter indicates the maximum number of seconds that any instance recovery should take. The DBW process will then manage database writes in such a way as to attempt to meet the number of seconds that FAST_START_MTTR_RECOVERY is set to.

Some parameters can disable Fast-Start Fault Recovery. These include the parameters FAST_START_IO_TARGET, LOG_CHECKPOINT_INTERVAL, and LOG_CHECKPOINT_TIMEOUT. If your database is using these parameters, then you are not taking advantage of Fast-Start Fault Recovery.

Monitoring Instance Recovery Because instance recovery is the instance-recovery process that is the most time sensitive, you will want to monitor your critical databases to make sure that they can meet your instance recovery time SLAs. The view V$INSTANCE_RECOVERY provides information on both the target mean time to recover the database and the currently estimated mean time to recover the database. The ESTIMATED_MTTR column represents the mean time to recover (MTTR) for the current system based on its current activity.

The Mean Time To Recover Advisor Trying to correctly set the FAST_START_MTTR_RECOVERY parameter can be difficult. If you are sensitive to performance but also are sensitive to instance recovery time, the Mean Time to Recover Advisor (MTTR Advisor) can be of assistance. If you look at the V$INSTANCE_RECOVERY view, you will see the column TARGET_MTTR. This column represents the maximum attainable MTTR for that database and is calculated by the MTTR Advisor. Both the ESTIMATED_MTTR and TARGET_MTTR columns will vary based on database activity.

You can view the results of the MTTR Advisor through the V$MTTR_TARGET_ADVICE view. This view provides an analysis of the current setting of the FAST_START_MTTR_RECOVERY parameter and how adjustments of that parameter will impact instance recovery time.

Sybex's OCA preparation guide for Oracle Database 12*c*, *OCA: Oracle Database 12*c *Administrator Certified Associate Study Guide: Exams 1Z0-061 and 1Z0-062* (Sybex, 2014) by Biju Thomas, provides complete coverage on starting the Oracle Database and the different stages of opening the Oracle Database, so this is just a quick review.

Starting and Stopping the Database

During backup and recovery operations, you will need to know how to start up and shut down your database correctly. To start up the database in any of the modes described in the previous section, you will use the STARTUP command or the ALTER DATABASE command, as required. To stop the database, you will use the SHUTDOWN command. Typically, database startup operations are performed from SQL*Plus or Oracle Enterprise Manager.

The `STARTUP` Command This command is used to start the instance and/or database only when the database is in a shutdown state. The `STARTUP` command can be used to completely open the database, as shown in this code snippet:

```
SQL> startup
```

The `STARTUP` command also has options that you can use to indicate that you want Oracle to start the startup operation at a certain point. For example, you can indicate that you want the instance to be started only by using the `STARTUP NOMOUNT` command:

```
SQL> startup nomount
```

Or perhaps you want to start the instance and mount the database. In this case, the command would be as follows:

```
SQL> startup mount
```

Sometimes you want to shut down the database and start it up in one command. You can use the `STARTUP FORCE` command to perform this action. Note that the `STARTUP FORCE` command will shut down your database in an inconsistent manner (which we discussed earlier in this chapter), and some operations (such as putting the database in ARCHIVELOG mode) will not complete successfully if the database was shut down in an inconsistent manner. Here is an example of the `STARTUP FORCE` command:

```
SQL> startup force
```

The SHUTDOWN Command The `SHUTDOWN` command does what it says; it shuts down the database. As with the `SHUTDOWN` command, it comes with a few options. First there is the plain-Jane `SHUTDOWN` command, which will shut down the database if absolutely nothing is going on and if absolutely no one is logged in. You can guess how often those conditions happen in reality! Until its conditions are met, the `SHUTDOWN` command will just sit there, waiting for its opportunity to shut down the database.

Here is an example of the `SHUTDOWN` command:

```
SQL> shutdown
```

If waiting is not your forte, then you may want to try the `shutdown immediate` command. The `shutdown immediate` command will prevent new logons, roll back any uncommitted transactions, and then bring the database down. It's a consistent-shutdown, no-waiting approach to stopping the database, and lots of DBAs like it. Here is an example:

```
SQL> shutdown immediate
```

The cousin of `SHUTDOWN IMMEDIATE` is `SHUTDOWN TRANSACTIONAL` (we are not sure if it's a first cousin or second cousin; Oracle has not defined this within the body of the Oracle

documentation yet). The main difference here is that the SHUTDOWN TRANSACTIONAL command will wait for active transactions to complete (commit) before shutting down those sessions. As a result, the SHUTDOWN TRANSACTIONAL command can take a while longer to complete its task, but on the positive side, users might be a little bit happier (if they are actually able to be happy anytime the database comes down). Here is an example of the SHUTDOWN TRANSACTIONAL command:

```
SQL> shutdown transactional
```

The bad boy of database shutdowns is the SHUTDOWN ABORT command. If you want your database to come down without debate, this is the way to do it. This is like pulling the power cord on your database; it is a crash of the database, shutting it down in an inconsistent manner. Here is an example of the SHUTDOWN ABORT command:

```
SQL> shutdown abort
```

The SHUTDOWN ABORT Command: The Truth Is Out There

As long as the SHUTDOWN ABORT command has been around it has been surrounded in controversy. It is believed by some that the Seven-Day War was actually started as the result of a disagreement between DBAs over the SHUTDOWN ABORT command (they conveniently ignore the fact that Oracle didn't even exist then). The truth is that the SHUTDOWN ABORT command is the fastest way to shut down your database. Because the database will be shut down in an inconsistent manner, it may result in a delayed database startup because of the recovery process that Oracle must go through internally. Often, though, SHUTDOWN ABORT may well be the way to get your database shut down and started back up in the shortest amount of time possible.

The ALTER DATABASE Command The ALTER DATABASE command is used to move the database from one state to another. For example, if the instance was started with the STARTUP NOMOUNT command, you may want to mount the database. To do so, you would use the ALTER DATABASE MOUNT command, as shown here:

```
SQL> alter database mount;
```

If the database is already mounted and you want to open it, then the ALTER DATABASE OPEN command would be appropriate, as shown in this example:

```
SQL> alter database open;
```

Performing Database Restricted-Mode Operations

Sometimes it's nice to have the house to yourself, isn't it? Oracle allows you the equivalent of having the house to yourself when you put the database in restricted mode. When the database is in restricted mode, only those with the restricted session privilege can access it. Since DBA accounts have restricted session privileges, this means you can log into the database and do your work, feeling secure that other users won't get in and cause problems. You may find that during certain recovery operations a restricted session will help when you need to get into the database to perform some DBA-related activities but you don't want other users to log in yet.

To open the database in restricted mode, you issue the `STARTUP RESTRICT` command. If your database is already open, you can put it in restricted mode with the `ALTER DATABASE ENABLE RESTRICTED SESSION` command. This will not impact existing users, but new users will not be able to connect unless they have the restricted session privilege. To disable the restricted session and allow users to connect to the database, use the `ALTER DATABASE DISABLE RESTRICTED SESSION` command.

Performing Backup and Recovery Operations and Getting Database Status

So, what kind of operations would you perform given the different open or closed combinations of the database? Here are some examples:

Operations while the instance is down and the database is not open:

- Copy the spfile to a pfile
- Copy the pfile to an spfile
- Perform manual cold backups

Operations while the instance is open and the database is not open:

- Create a database
- Create a database control file
- Restore the database control file or spfile from RMAN

Operations while the instance is mounted and the database is not open:

- Cold backup with RMAN
- Recovery of critical data files (`SYSTEM`, `UNDO` tablespaces)
- Offline recovery of entire database

Operations while the instance is mounted and the database is open:

- Online data file or tablespace recovery of noncritical tablespaces
- Online backups of the database

Configuring the Database for Backup and Recovery

The ARCHIVELOG and NOARCHIVELOG modes of the database really boil down to what your backup and recovery needs and requirements are. If your recovery needs are simple, and you just want to restore to the point of your last backup, then you can leave your database in NOARCHIVELOG mode and do offline backups. If your database uptime requirements provide time to shut down the database for your backup, then NOARCHIVELOG mode will work fine for you.

If you are going to do offline backups of your database in NOARCHIVELOG mode, then you can pretty much ignore this section. All you will need to do there (as we will cover in the next section) is determine the location of the files you need to back up and the location to which you want to back them up, shut down the database, and back it up.

When you want to perform online backups, or if you want to be able to recover offline backups beyond the time of the backup, then the database needs to be in ARCHIVELOG mode. Putting the database in ARCHIVELOG mode requires some configuration, which we will discuss first. Once you have configured the database, you will then actually put it in ARCHIVELOG mode, which we will cover next.

You will need to understand both NOARCHIVELOG and ARCHIVELOG mode for the OCP exam. While manual database backup and recovery–related questions will only ask you about NOARCHIVELOG mode situations, RMAN questions may well ask you about backup and recovery in both modes.

In the following sections, we will discuss configuring the database for ARCHIVELOG mode operations (no additional configuration for NOARCHIVELOG mode operations is required unless you want to configure the Fast Recovery Area for RMAN, which is covered in Chapter 3), and then we will discuss actually putting the database in ARCHIVELOG mode. Finally, we will discuss some database views that will be useful when managing a database in ARCHIVELOG mode.

Configuring for ARCHIVELOG Mode

The first step to putting the database in ARCHIVELOG mode is to set the database parameters. You need to set the database parameters so that the ARCH process will work correctly when it needs to archive the online redo logs, creating archived redo logs.

You will want to consider setting several parameters when you are going to put the database in ARCHIVELOG mode. A number of parameters are directly associated with user-managed backup and recovery in ARCHIVELOG mode. Table 1.2 describes these parameters.

Real World Scenario

Mixing NOARCHIVELOG and ARCHIVELOG Modes

In the real world, you might find that some of your databases are running in NOARCHIVELOG mode and some of your databases are running in ARCHIVELOG mode. For example, your development and test databases may be able to be shut down at night for backups. Additionally, they might not have a need for point-in-time or point-of-failure recovery. Thus, NOARCHIVELOG mode is just fine for them.

You may find that your production databases have different requirements. First, it may be that shutting down your production systems for backups at any time is not acceptable to your customer. Further, you probably will also find that people prefer not to lose data in production and that they would prefer to be able to restore the database and then recover all work that occurred after the last backup. You will have to put your database in ARCHIVELOG mode to satisfy those requirements.

Perhaps you are asking yourself, "What about this Fast Recovery Area thing I've been hearing about?" Since the Fast Recovery Area (FRA, previously called the Flash Recovery Area) typically is not used for manual backups, we will defer discussion of the FRA until we discuss RMAN in later chapters. Thus, the parameters related to the RMAN and the FRA are not included in Table 1.2.

TABLE 1.2 Oracle Parameters Associated with User-Managed Backup and Recovery

Parameter Name	Description
LOG_ARCHIVE_DEST	Indicates the destination to copy archived redo logs to. Typically this parameter is not set and the LOG_ARCHIVE_DEST_*n* parameter is set instead.
LOG_ARCHIVE_DEST_*n*	Indicates one of up to 10 destinations to copy archived redo logs to. The first destination starts with 1 (LOG_ARCHIVE_DEST_01).
LOG_ARCHIVE_DEST_STATE_*n*	Indicates the state of LOG_ARCHIVE_DEST_*n* (ENABLED, DEFERRED, or ALTERNATE).
LOG_ARCHIVE_FORMAT	Indicates the format of the archived redo log filenames.

Assume that you have a database called orcl and you have decided that you want Oracle to back up your archived redo logs to a directory called c:\oracle\archivelog\orcl (in Unix, perhaps it's called /oracle/archivelog/orcl). You would first have to create the file system directory structures, and then you would need to set the appropriate parameters. In this case, you would use the ALTER SYSTEM command to set the LOG_ARCHIVE_DEST_1 parameter to point to c:\oracle\archivelog\orcl, as shown in this code example:

```
Alter system set log_archive_dest_1='location=c:\oracle\archivelog\orcl';
```

You can also clear this parameter setting by just using blank quotes, as shown in this example:

```
Alter system set log_archive_dest_1='';
```

With the LOG_ARCHIVE_DEST_*n* parameter, you can configure up to 31 different archive-log destination directories. This feature can be used to provide redundant backup locations for your archive logs to protect them in the event of a failure of one or more of those locations. For example, you could archive to a local disk, and you could archive to an NFS-mounted disk. In that case, you would have two LOG_ARCHIVE_DEST_*n* parameters set like this:

```
-- Local mount on C: drive
Alter system set log_archive_dest_1='location=c:\oracle\archivelog\orcl';
-- NFS Mount on Z: drive
Alter system set log_archive_dest_2='location=Z:\oracle\archivelog\orcl';
```

Oracle will archive to both destinations, in parallel. This type of configuration is also used in more advanced database setups such as standby databases.

Archive-log destination directories can take on different states, such as ENABLED, DEFERRED, or ALTERNATE (I tried to get the folks at Oracle to add a state of EXAUSTION or FRUSTRATION; they said no, but they did seem to like the suggestion of a state of CONFUSION). They can also be defined as MANDATORY or OPTIONAL. You will not need to be aware of these advanced settings for your OCP exam, but you might need to use these options as a part of your normal duties. You can find more information on these different attributes in the Oracle documentation.

You may also want to control how Oracle names the archived redo logs. This is done with the LOG_ARCHIVE_FORMAT parameter. For example, you may want to put the database name in the name of the archive logs being created, but you also want them to be numbered in such a way that the name will always be unique. You can set the LOG_ARCHIVE_FORMAT string to a value of orcl_%s_%t_%r_%d.arc, as shown in this example:

```
Alter system set log_archive_format='orcl_%s_%t_%r_%d.arc' scope=spfile;
```

You may wonder what the %s, %t, %r, and %d represent. These are variables that represent values for particular components of the archived redo log. The %s represents the sequence number, which is always unique for a given database (until a RESETLOGS command occurs, which we will discuss in Chapter 2). The %t is the thread number that represents an individual node on a cluster when your database is running on Oracle's Real Application Clusters (RAC). The %r represents the reset logs number (see Chapter 2). Finally, the %d represents the DBID that should be unique for each database. Together, this string of variables will make the archive log filenames unique for every database on your system.

Every database in Oracle has a DBID, which is a unique identifier for the database (see the DBID column in V$DATABASE to see your DBID). Be careful, though! It is possible to have databases on two different boxes with the same name and even with the same DBID. If you will be sharing an ARCHIVELOG mount point between boxes (say, via NFS), you will need to make sure you do not accidentally overwrite archived redo logs originating from databases with the same name and/or DBID! You won't need to know about the DBID for your OCP exam, but we thought we'd tell you anyway!

Putting the Database in ARCHIVELOG Mode

Putting the database in ARCHIVELOG mode is as easy as following these steps:

1. Configure archiving-related parameters as shown in the previous section.
2. Shut down the database in a consistent state using the SHUTDOWN, SHUTDOWN IMMEDIATE, or SHUTDOWN TRANSACTIONAL command.
3. Mount the database with the STARTUP MOUNT command.
4. Put the database in ARCHIVELOG mode with the ALTER DATABASE ARCHIVELOG command.
5. Open the database with the ALTER DATABASE OPEN command.

One thing to be aware of when the database is in ARCHIVELOG mode is that if you have not configured archiving correctly, you could find yourself with a database that just stops running. If Oracle cannot archive the online redo logs, it will suspend all database operations once it cycles through all the online redo log groups. So, for example, if your database does log switches every 10 minutes and has three redo log groups, your database will mysteriously freeze after 30 minutes. Lack of configuration is not as big of a problem in Oracle Database 12*c* as it was in earlier versions since Oracle defaults to using the Fast Recovery Area (discussed in Chapter 3).

Similar problems can occur if the archive-log destination directory runs out of space or if the permissions are not set correctly. If Oracle starts to have problems of this sort, it will

log an error in the alert log of the database. Here is an example of an error you might see in the alert log (we discuss finding the alert log of the database later in this chapter):

```
All online logs needed archiving
ARCH: Archival stopped, error occurred. Will continue retrying
ORA-16014: log 2 sequence# 29 not archived, no available destinations
ORA-00312: online log 2 thread 1: 'C:\ORACLE\ORADATA\ORCL\REDO02.LOG'
```

Also, users logging into the database will find their logins just hanging until the archive-log problems are solved.

We've covered the parameters needed to put the database in ARCHIVELOG mode and the basic steps involved in the process. Let's look at an example of actually putting the database in ARCHIVELOG mode. Exercise 1.1 provides an example of doing just that.

Real World Scenario

Online Redo Logs Stop Being Archived

We can't tell you how many times this has happened to us as DBAs. You are busy designing some cool model and the operations guys call. "Hey," they say, "we are getting calls. The database isn't working anymore."

"What?" you respond. "What do you mean it's not working?"

"The user sessions are just stalled, sitting there not doing anything. It's like the database has gone out to lunch or something," the operations guy says.

Immediately you are pretty sure you know what's wrong. So you ask the operations guy, "So, the mount point for the archived redo logs. Are you possibly getting an alert that it's full?"

The operations guy fumbles around to look at the alerts. Sure enough, the archived redo log destination is filled up. "Oh, yeah … I was going to call you about that but I forgot."

So, you proceed to back up the archived redo logs and then remove them from the system to free up space. You also follow up to make sure some additional disk space is added to the file system.

This kind of situation happens a lot if you do not have enough space available for your archived redo logs or if your backups stop working. I've seen this happen frequently in shops where the database was originally designed for a certain amount of use and that usage has increased significantly.

EXERCISE 1.1

Putting a Database in ARCHIVELOG Mode

In this exercise you will take a database that is in NOARCHIVELOG mode and put it in ARCHIVELOG mode.

1. First, validate that the database is in NOARCHIVELOG mode using the V$DATABASE column LOG_MODE:

```
SQL> Select log_mode from v$database;
LOG_MODE
------------
NOARCHIVELOG
```

2. Next, look at the settings for the parameters (note that we have removed some of the LOG_ARCHIVE_DEST_*n* parameter to save trees) LOG_ARCHIVE_DEST_1 and LOG_ARCHIVE_FORMAT:

```
SQL> show parameter log_archive_dest_1
NAME                                 TYPE        VALUE
------------------------------------ ----------- -----------
log_archive_dest_1                   string
log_archive_dest_10                  string
SQL> show parameter log_archive_format
NAME                                 TYPE        VALUE
------------------------------------ ----------- -----------
log_archive_format                   string      ARC%S_%R.%T
```

3. Create the archive log directory c:\oracle\arch\orcl:

```
SQL> host mkdir c:\oracle\arch\orcl
```

4. You want to modify LOG_ARCHIVE_DEST_1 and LOG_ARCHIVE_FORMAT so that they are set correctly. LOG_ARCHIVE_DEST_1 should be set to c:\oracle\arch\orcl and LOG_ARCHIVE_FORMAT should be orcl_%r_%t_%s.arc. You will use the ALTER SYSTEM command to set these parameters. You will then check to make sure they are set correctly.

```
Alter system set log_archive_dest_1='location=c:\oracle\arch\orcl';
-- Note that we have to use the scope=spfile on this next parameter.
-- This is because it's not dynamic!
Alter system set log_archive_format='orcl_%r_%t_%s.arc' scope=spfile;
```

EXERCISE 1.1 *(continued)*

5. Next, shut down the database in a consistent manner with the `SHUTDOWN IMMEDIATE` command:

```
SQL> shutdown immediate
Database closed.
Database dismounted.
ORACLE instance shut down.
```

6. Now mount the database with the `STARTUP MOUND` command:

```
SQL> startup mount;
ORACLE instance started.
Total System Global Area  418484224 bytes
Fixed Size                  1333592 bytes
Variable Size             348128936 bytes
Database Buffers           62914560 bytes
Redo Buffers                6107136 bytes
Database Mounted.
```

7. Put the database in ARCHIVELOG mode with the `ALTER DATABASE ARCHIVELOG` command:

```
SQL> alter database archivelog;
Database altered.
```

8. Open the database for operations:

```
SQL> alter database open;
Database altered.
```

9. Make sure the database is in ARCHIVELOG mode:

```
SQL> Select log_mode from v$database;
LOG_MODE
------------
ARCHIVELOG
```

10. It is a good idea to make sure that everything is configured correctly and that the archived redo logs are getting generated in the place where you expect them to be generated. So, first you will force an archive-log switch with the `ALTER SYSTEM`

EXERCISE 1.1 *(continued)*

SWITCH LOGFILE command. This will cause a log switch to the next redo log group, and ARCH will need to copy the redo log to an archived redo log:

```
SQL> alter system switch logfile;
System altered.
```

11. Look in the c:\oracle\arch\orcl directory. You should see a file in that directory:

```
SQL> host dir c:\oracle\arch\orcl
 Volume in drive C has no label.
 Volume Serial Number is 08DE-E1AB
 Directory of c:\oracle\arch\orcl
08/02/2013  12:44 PM    <DIR>          .
08/02/2013  12:44 PM    <DIR>          ..
08/02/2013  12:44 PM        41,032,192 ORCL_658485967_1_2.ARC
               1 File(s)     41,032,192 bytes
               2 Dir(s)  17,065,476,096 bytes free
```

The ORCL_659495967_1_2.ARC file is your archive log file, so ARCH is copying the log file to the correct location. Excellent job!

What If the Archived Redo Logs Are Not Getting Created?

So, what if you don't see an archived redo log in the directory where you think it's supposed to be? Double-check the LOG_ARCHIVE_DEST_1 parameter and make sure it's set correctly. This is usually the problem. You can issue the command SHOW PARAMETER LOG_ARCHIVE_DEST_1 from SQL*Plus to do this. Make sure the directory exists, check the security permissions on the directory, and make sure you have enough space available on the file system.

If the archive logs are not getting created correctly, you will need to quickly figure out why. If Oracle switches through all of the available online redo logs and tries to switch into one that has previously been used and is waiting to be archived, all database activity will be suspended until the archived redo log can be completely written out.

Using ARCHIVELOG Mode Data Dictionary Views

Oracle provides several data dictionary views that can be used to monitor and manage the online and archived redo logs. Table 1.3 describes those views.

TABLE 1.3 Oracle Dynamic Performance Views Associated with User-Managed Backup and Recovery

View Name	Description
V$ARCHIVE	The V$ARCHIVE view provides information on redo logs that are in need of being archived.
V$ARCHIVE_DEST	The V$ARCHIVE_DEST view provides information on each individual archive-log destination. Typically this view is used for Oracle Data Guard.
V$ARCHIVE_DEST_STATUS	The V$ARCHIVE_DEST_STATUS view provides status information on each of the individual archive-log destination directories.
V$ARCHIVE_PROCESSES	The V$ARCHIVE_PROCESSES view provides information on the different ARCH processes running on your system.
V$ARCHIVED_LOG	The V$ARCHIVED_LOG view provides information on individual archived redo logs.
V$LOG	The V$LOG view provides information on the online redo log groups.
V$LOGFILE	The V$LOGFILE view provides information on specific online redo logs.
V$LOG_HISTORY	The V$LOG_HISTORY view provides historical information on all online/archived redo logs.

Using the V$ views is easy to do, and they can tell you a lot about the status of both online and archived redo logs, as shown in Exercise 1.2.

EXERCISE 1.2

Putting the V$ Views to Work

The V$ views are very useful when you want to find out something about your database related to backup or recovery. In this exercise, we will look at some V$ views related to the database online redo logs.

1. Let's look at the current redo logs that have been archived:

```
SQL> select name, thread#, sequence# from v$archived_log;
NAME                                          THREAD#  SEQUENCE#
--------------------------------------------- ---------- ----------
C:\ORACLE\ARCH\ORCL\ORCL_658485967_1_2.ARC             1          2
```

EXERCISE 1.2 *(continued)*

```
C:\ORACLE\ARCH\ORCL\ORCL_658485967_1_3.ARC              1          3
C:\ORACLE\ARCH\ORCL\ORCL_658485967_1_4.ARC              1          4
C:\ORACLE\ARCH\ORCL\ORCL_658485967_1_5.ARC              1          5
```

In the output, you will find the name of the archived redo log. You also see the thread number (in case you are running RAC) and the log sequence number. Note that since you have put the database in ARCHIVELOG mode, you have generated four archived redo logs.

2. You can see where your online redo logs are by using the `V$LOGFILE` view, as shown in this example:

```
SQL> select group#, status, member from v$logfile;
    GROUP# STATUS  MEMBER
---------- ------- --------------------------------
         3         C:\ORACLE\ORADATA\ORCL\REDO03A.LOG
         2         C:\ORACLE\ORADATA\ORCL\REDO02.LOG
         1         C:\ORACLE\ORADATA\ORCL\REDO01.LOG
         3         C:\ORACLE\ORADATA\ORCL\REDO03B.LOG
```

In this output, you can see you have three online redo log groups. It is interesting to note that group 3 actually has two members, whereas groups 1 and 2 have one member each.

3. You can see which is the current online redo log group by querying the `V$LOG` view, as shown here:

```
SQL> select group#, sequence#,  status from v$log;
    GROUP#  SEQUENCE# STATUS
     ----------     ----------  ----------------
         1         13 CURRENT
         2         11 ACTIVE
         3         12 ACTIVE
```

In this example, log group 1 (marked with a `CURRENT` status) is the group that Oracle is currently writing to. Note that sequences 11 and 12 are marked active. This implies that they have not been archived yet or that they are being archived. They will be marked inactive once ARCH has finished archiving them.

Performing Oracle Offline Backups

We have been talking a lot about ARCHIVELOG mode and preparing for online backups, but we first need to talk about how to do offline backups in Oracle. Offline backups are actually quite easy to do, as you will see in Exercise 1.3, where you will be backing up a database with an offline backup.

This is the kind of manual backup you may be asked about in the OCP exam.

EXERCISE 1.3

Executing an Offline Backup

In this exercise, you will be executing an offline backup of your database. Follow these steps to back up a database with an offline backup:

1. First, you need to determine which files to back up. You will need to know the location of the data files, the control file, and the online redo logs. You use the `FILE_NAME` column of the `DBA_DATA_FILES` view to find the data files:

```
SQL> Select file_name from dba_data_files;
FILE_NAME
-----------------------------------------
C:\ORACLE\ORADATA\ORCL\USERS01.DBF
C:\ORACLE\ORADATA\ORCL\UNDOTBS01.DBF
C:\ORACLE\ORADATA\ORCL\SYSAUX01.DBF
C:\ORACLE\ORADATA\ORCL\SYSTEM01.DBF
C:\ORACLE\ORADATA\ORCL\REVEAL_DATA_01.DBF
C:\ORACLE\ORADATA\ORCL\REVEAL_INDEX_01.DBF
C:\ORACLE\ORADATA\ORCL\USERS02.DBF
7 rows selected.
```

2. You use the `MEMBER` column in the `V$LOGFILE` view to find the location of all the online redo logs:

```
SQL> select member from v$logfile;
MEMBER
---------------------------------
C:\ORACLE\ORADATA\ORCL\REDO03A.LOG
```

EXERCISE 1.3 *(continued)*

```
C:\ORACLE\ORADATA\ORCL\REDO02.LOG
C:\ORACLE\ORADATA\ORCL\REDO01.LOG
C:\ORACLE\ORADATA\ORCL\REDO03B.LOG
```

3. You use the NAME column in V$CONTROLFILE to find the control files:

```
SQL> select name from v$controlfile;
NAME
------------------------------------
C:\ORACLE\ORADATA\ORCL\CONTROL01.CTL
C:\ORACLE\ORADATA\ORCL\CONTROL02.CTL
C:\ORACLE\ORADATA\ORCL\CONTROL03.CTL
C:\ORACLE\ORADATA\ORCL\CONTROL04.CTL
```

4. Having found all the files you will need for your backup, create a directory to back up all your files to. Of course, you might back up your files to tape or a thumb drive or some such thing. In this case, you will just copy the files to a directory that you will create called c:\backup\orcl\backup1:

```
SQL> host mkdir c:\backup\orcl\backup1
```

5. Having created your backup directory, you need to shut down the database with the SHUTDOWN IMMEDIATE command before you start your backup:

```
SQL> shutdown immediate
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL> exit
```

6. Now copy the files that you found in steps 1, 2, and 3 to the backup directory created in step 4. Notice that all the files in this example reside in one directory, c:\oracle\oradata\orcl, so the COPY command is quite easy. Backups can take a while, so be patient. It's probably a good time to go grab a cool refreshment from the vending machine!

```
C:\>copy c:\oracle\oradata\orcl\*.* c:\backup\orcl\backup1
c:\oracle\oradata\orcl\CONTROL01.CTL
c:\oracle\oradata\orcl\CONTROL02.CTL
c:\oracle\oradata\orcl\CONTROL03.CTL
c:\oracle\oradata\orcl\CONTROL04.CTL
c:\oracle\oradata\orcl\REDO01.LOG
```

EXERCISE 1.3 *(continued)*

```
c:\oracle\oradata\orcl\REDO02.LOG
c:\oracle\oradata\orcl\REDO03A.LOG
c:\oracle\oradata\orcl\REDO03B.LOG
c:\oracle\oradata\orcl\REVEAL_DATA_01.DBF
c:\oracle\oradata\orcl\REVEAL_INDEX_01.DBF
c:\oracle\oradata\orcl\SYSAUX01.DBF
c:\oracle\oradata\orcl\SYSTEM01.DBF
c:\oracle\oradata\orcl\TEMP01.DBF
c:\oracle\oradata\orcl\UNDOTBS01.DBF
c:\oracle\oradata\orcl\USERS01.DBF
c:\oracle\oradata\orcl\USERS02.DBF
       16 file(s) copied.
```

7. Once the copy is complete, verify that the backup is where you expect it to be:

```
C:\>dir c:\backup\orcl\backup1
 Volume in drive C has no label.
 Volume Serial Number is 08DE-E1AB
 Directory of c:\backup\orcl\backup1
08/02/2013  02:16 PM    <DIR>          .
08/02/2013  02:16 PM    <DIR>          ..
08/02/2013  02:02 PM        10,174,464 CONTROL01.CTL
08/02/2013  02:02 PM        10,174,464 CONTROL02.CTL
08/02/2013  02:02 PM        10,174,464 CONTROL03.CTL
08/02/2013  02:02 PM        10,174,464 CONTROL04.CTL
08/02/2013  02:02 PM        52,429,312 REDO01.LOG
08/02/2013  02:02 PM        52,429,312 REDO02.LOG
08/02/2013  02:02 PM       104,858,112 REDO03A.LOG
08/02/2013  02:02 PM       104,858,112 REDO03B.LOG
08/02/2013  02:02 PM        15,736,832 REVEAL_DATA_01.DBF
08/02/2013  02:02 PM        15,736,832 REVEAL_INDEX_01.DBF
08/02/2013  02:02 PM       851,386,368 SYSAUX01.DBF
08/02/2013  02:02 PM       754,982,912 SYSTEM01.DBF
08/02/2013  02:02 PM        50,339,840 TEMP01.DBF
08/02/2013  02:02 PM       519,053,312 UNDOTBS01.DBF
08/02/2013  02:02 PM       581,246,976 USERS01.DBF
08/02/2013  02:02 PM        10,493,952 USERS02.DBF
              16 File(s)  3,154,249,728 bytes
               2 Dir(s)  13,330,685,952 bytes free
```

EXERCISE 1.3 *(continued)*

8. Start the database. Your backup is complete!

```
SQL> startup
ORACLE instance started.
Total System Global Area  418484224 bytes
Fixed Size                  1333592 bytes
Variable Size             348128936 bytes
Database Buffers           62914560 bytes
Redo Buffers                6107136 bytes
Database mounted.
Database opened.
```

You could always decide to compress the backup files with a utility like PKZIP to save space if you wanted. By the way, RMAN can do this for you!

That's all there is to an offline database backup. In the next chapter, you will see that recovering the database using this backup is just as easy!

Temporary Tablespaces and Backups

Temporary tablespaces created with the `CREATE TEMPORARY TABLESPACE` command do not need to be backed up. The tempfiles associated with temporary tablespaces can be recreated on the fly as needed. This is true with both online backups and offline backups. If you are using the old-style temporary tablespaces that are not using tempfiles, you will still need to back up those data files.

To recreate tempfiles, simply use the `ALTER TABLESPACE` command with the `ADD TEMPFILE` keyword, as shown here:

```
Alter tablespace my_temp
Add tempfile '/u01/db01/mytempfile01.dbf' size 100m;
```

Performing Oracle Online Backups

Oracle online backups are not difficult to do; they just require a few additional steps. In this section, we will introduce you to Oracle online backups. First, we will discuss online backups and generally how to do them. We will then present an example of performing an online backup.

The Mechanics of Online Backups

To do Oracle online backups, your database must be in ARCHIVELOG mode. You can back up the entire database or you can choose to back up a specific tablespace or set of tablespaces. If you choose to back up only specific tablespaces, you will not be able to recover your database until you have at least a base backup of all of its tablespaces. That said, you can back up the tablespaces at different times if you prefer (though this is not common practice). For example, you could back up the `SYSTEM` tablespace on Monday, the `USERS` tablespace on Tuesday, and so on. As long as you have a complete cumulative backup of the database (taken at different times), you can recover it.

To start an online backup, you will need to put each tablespace in hot backup mode. This can be done by using the `ALTER DATABASE BEGIN BACKUP` command, or you can individually put tablespaces in hot backup mode with the `ALTER DATABASE BEGIN BACKUP` command. After you have put the tablespaces in hot backup mode, you back up the underlying data files of that tablespace. If you need to know where the data files related to that tablespace reside, you can use the `DBA_DATA_FILES` view.

When a Tablespace Is in Hot Backup Mode

When you put a tablespace in hot backup mode, Oracle will start writing block-sized records to the redo logs. These records are much bigger than the normal-sized records, so this can cause performance problems.

One odd misconception we hear from time to time is that Oracle will stop writing to the database data files during a hot backup. In fact Oracle will continue to write changes to the data files; however, it will not update the data file headers until the backup is complete.

When you put a tablespace in hot backup mode, you are really putting the underlying data files of that tablespace in hot backup mode. You can determine if a data file is in hot backup mode by querying the `V$BACKUP` view. The `STATUS` column will indicate `ACTIVE` if

the given data file is in hot backup mode. Here is an example of such a query where our USERS tablespace is in hot backup mode, as indicated by the ACTIVE status:

```
SQL>  select a.tablespace_name, b.status
  2   from dba_data_files a, v$backup b
  3   where a.file_id=b.file#
  4  order by tablespace_name;

TABLESPACE_NAME                STATUS
------------------------------ -----------
REVEAL_DATA                    NOT ACTIVE
REVEAL_INDEX                   NOT ACTIVE
SYSAUX                         NOT ACTIVE
SYSTEM                         NOT ACTIVE
UNDOTBS1                       NOT ACTIVE
USERS                          ACTIVE
USERS                          ACTIVE
```

Another thing to be aware of is what happens if the database is shut down while data files are in hot backup mode. First, Oracle will not allow you to shut down a database with most shutdown commands (SHUTDOWN, SHUTDOWN IMMEDIATE, or SHUTDOWN TRANSACTIONAL) while a tablespace is in hot backup mode. Instead it will generate an error, as shown here:

```
ORA-01149: cannot shutdown - file 4 has online backup set
ORA-01110: data file 4: 'C:\ORACLE\ORADATA\ORCL\USERS01.DBF'
```

This error identifies the data file that is in hot backup mode. You would need to determine which tablespace the data file is assigned to by looking at the DBA_DATA_FILES view. You would then issue the ALTER TABLESPACE END BACKUP command to take it out of hot backup mode.

If you issue a SHUTDOWN ABORT, STARTUP FORCE, or if the database crashes for some reason or the server shuts down without shutting down the database in a natural fashion, Oracle will not restart with a data file in hot backup mode. You will see the following error when you try to restart the database:

```
ORA-10873: file 4 needs end backup before opening a database
ORA-01110: data file 4: 'C:\ORACLE\ORADATA\ORCL\USERS01.DBF'
```

You simply issue the command ALTER DATABASE END BACKUP to take the data files out of hot backup mode and then ALTER DATABASE OPEN to open the database.

You can take the tablespaces out of hot backup mode with the ALTER DATABASE END BACKUP command, or you can individually take each tablespace out of hot backup mode by issuing the ALTER TABLESPACE END BACKUP command.

Once you complete the online backup, one more very important step is to back up the archived redo logs that were generated during the backup. You will need each log that was generated from the time you issued the `ALTER DATABASE BEGIN BACKUP` command until you issued the `ALTER DATABASE END BACKUP` command. After the backup, use the `ALTER SYSTEM SWITCH LOGFILE` command to force a log switch to cause the current online redo log (which contains redo generated during the backup) to be archived after you have completed the backup. You will need the redo in this log file, and any other archived redo logs that might have been generated during the backup, in order to recover the database from the backup you just completed.

In addition to regular online backups, you will want to schedule regular backups of your archived redo logs to protect them as much as possible. For example, if the online backup in the exercise ended at 4 p.m., you would be able to restore the database up to 4 p.m. with the archived redo logs you backed up. Archived redo logs will continue to be generated, though, and if you want to be able to recover your database to a point beyond 5 p.m., you will need to have those later-generated archived redo logs available (more on recovery in the next chapter). Thus it is a good idea to have regular backups of your archived redo logs!

In Exercise 1.4, we walk you through the process of doing an online backup.

EXERCISE 1.4

Executing an Online Backup

In this exercise you will be performing an online database backup. As mentioned in the text, your database will need to be in ARCHIVELOG mode to successfully execute this backup.

1. We assume your database is already running in ARCHIVELOG mode. If it's not, return to Exercise 1.1 and put your database in ARCHIVELOG mode.
2. As with the previous offline/cold backup, you need to know what data files need to be backed up:

```
SQL> Select file_name from dba_data_files;
FILE_NAME
-----------------------------------------
C:\ORACLE\ORADATA\ORCL\USERS01.DBF
C:\ORACLE\ORADATA\ORCL\UNDOTBS01.DBF
C:\ORACLE\ORADATA\ORCL\SYSAUX01.DBF
C:\ORACLE\ORADATA\ORCL\SYSTEM01.DBF
C:\ORACLE\ORADATA\ORCL\REVEAL_DATA_01.DBF
C:\ORACLE\ORADATA\ORCL\REVEAL_INDEX_01.DBF
C:\ORACLE\ORADATA\ORCL\USERS02.DBF
7 rows selected.
```

EXERCISE 1.4 *(continued)*

3. Having determined which data files need to be backed up, you need to know where the archived redo logs are being copied to.

```
SQL> show parameter log_archive_dest_1
NAME                          TYPE        VALUE
----------------------------- ----------- ------------------------------
log_archive_dest_1            string      location=c:\oracle\arch\orcl
```

4. You should note the current online redo log sequence number at this point. You will need this, plus all log sequences generated during the backup, to be able to perform your recovery. You can get this number from the V$LOG view:

```
SQL> select group#, sequence#, status from v$log;
    GROUP#  SEQUENCE# STATUS
---------- ---------- ----------------
         1         13 INACTIVE
         2         14 CURRENT
         3         12 INACTIVE
```

   In this case, you see that you will need all log files from sequence number 14 on in order to restore the backup you are preparing to use.

5. You now need to put the database in hot backup mode. Oracle Database 12*c* provides the command ALTER DATABASE BEGIN BACKUP for this purpose. You can also back up specific tablespaces with the ALTER TABLESPACE BEGIN BACKUP command:

```
SQL> alter database begin backup;
Database altered.
-- ALTERNATE - Run this for each tablespace to be backed up.
-- alter tablespace users begin backup;
```

6. The database data files are now ready to be backed up. You will copy the files to a directory that you will create called c:\backup\orcl\backup2:

```
SQL> host mkdir c:\backup\orcl\backup2
```

7. Now copy all the database data files to this directory. In this case, all the files are in the directory c:\oracle\oradata\orcl, and the filenames all end with an extension of .DBF, so the command to copy them is pretty easy. Once you have started the data file copy, go get something to eat. It might take a while.

```
SQL> host copy c:\oracle\oradata\orcl\*.dbf c:\backup\orcl\backup2
c:\oracle\oradata\orcl\REVEAL_DATA_01.DBF
```

EXERCISE 1.4 *(continued)*

```
c:\oracle\oradata\orcl\REVEAL_INDEX_01.DBF
c:\oracle\oradata\orcl\SYSAUX01.DBF
c:\oracle\oradata\orcl\SYSTEM01.DBF
c:\oracle\oradata\orcl\TEMP01.DBF
c:\oracle\oradata\orcl\UNDOTBS01.DBF
c:\oracle\oradata\orcl\USERS01.DBF
c:\oracle\oradata\orcl\USERS02.DBF
        8 file(s) copied.
```

8. Having patiently waited for the backup to complete, you now need to take the database out of hot backup mode. Oracle Database 12*c* provides the command `ALTER DATABASE END BACKUP` for this purpose. You can also back up specific tablespaces with the `ALTER TABLESPACE END BACKUP` command:

```
SQL> alter database end backup;
Database altered.
-- ALTERNATE - Run this for each tablespace to be backed up.
-- alter tablespace users end backup;
```

9. Next, you need to determine the current log file sequence number. You will need the earlier log file that you identified and all log files generated during the backup up to the current log file to be able to restore this backup. The query is the same as the query against the `V$LOG` view that we showed you earlier in this chapter:

```
SQL> select group#, sequence#, status from v$log;
    GROUP#  SEQUENCE# STATUS
---------- ---------- ----------------
         1         13 INACTIVE
         2         14 ACTIVE
         3         15 CURRENT
```

   In this example, you can see that during the backup you had a log file switch, from sequence number 14 to sequence number 15. You see that 15 is the current sequence number. You know now that you will need to back up the logs with sequence numbers 14 and 15 in order to be able to restore this backup.

10. You now need to force a log switch so the log with sequence number 15 (the current online redo log sequence number) will be archived. To do this, you issue the `ALTER SYSTEM SWITCH LOGFILE` command. This will cause Oracle to switch to the next log file (sequence 16), and the current archive log (sequence 15) will be copied to the archive-log directory by the ARC*n* processes.

```
SQL> Alter system switch logfile;
System altered.
```

EXERCISE 1.4 *(continued)*

11. Having switched log files, you need to wait for ARCH to complete copying the last log file to the archive-log directory. You can check for this completion by looking at the V$ARCHIVED_LOG view:

```
SQL> Select sequence#, archived, status from v$archived_log
  2  Where sequence# between 14 and 15;
 SEQUENCE# ARC S
---------- --- -
        14 YES A
        15 YES A
```

Here you see that the logs with sequence numbers 14 and 15 (already identified as critical to restoring this backup) have been archived successfully. The ARCHIVED column indicates this with the use of the YES value.

12. Now back up all archived redo logs, ensuring that all logs with numbers between sequence *x* and sequence *y* are backed up. You will simply copy all archived redo logs from the directory identified in step 3 (c:\oracle\arch\orcl) to your backup directory:

```
SQL> Host copy c:\oracle\arch\orcl\*.* c:\backup\orcl\backup2
c:\oracle\arch\orcl\ORCL_658485967_1_10.ARC
c:\oracle\arch\orcl\ORCL_658485967_1_11.ARC
c:\oracle\arch\orcl\ORCL_658485967_1_12.ARC
c:\oracle\arch\orcl\ORCL_658485967_1_13.ARC
c:\oracle\arch\orcl\ORCL_658485967_1_14.ARC ⇓ Log sequence 14
c:\oracle\arch\orcl\ORCL_658485967_1_15.ARC ⇓ Log sequence 15
c:\oracle\arch\orcl\ORCL_658485967_1_2.ARC
c:\oracle\arch\orcl\ORCL_658485967_1_3.ARC
c:\oracle\arch\orcl\ORCL_658485967_1_4.ARC
c:\oracle\arch\orcl\ORCL_658485967_1_5.ARC
c:\oracle\arch\orcl\ORCL_658485967_1_6.ARC
c:\oracle\arch\orcl\ORCL_658485967_1_7.ARC
c:\oracle\arch\orcl\ORCL_658485967_1_8.ARC
c:\oracle\arch\orcl\ORCL_658485967_1_9.ARC
       14 file(s) copied.
```

You can tell that the logs with sequence numbers 14 and 15 were backed up since you know that the log sequence number is part of the filename (it's the number right before the extension). We also marked them for you in the output just because we are nice guys. After copying the archived redo logs to the backup location, you can delete the source location if you want to save space. Once the backup of the archived redo logs is complete, your database backup is done.

We hope you also realize that you will need to back up files like the database parameter file, any other Oracle-related configuration files (such as for networking), and the Oracle Database software itself. Backing up these structures (except for the spfile, which is a special RMAN case we will cover in Chapter 3) is beyond the scope of the OCP exam.

Backing Up the Control File

Finally, we need to talk about control-file backups. In Oracle there are three ways to manually back up a control file (again, RMAN methods will be covered in Chapter 3):

- Backing up the original control file during a cold backup
- Creating a backup control file
- Creating a trace file with the `CREATE CONTROL FILE` command in it

We have already covered the first method in this chapter. Let's look at the remaining two methods in some more detail. We will address recovering from a lost control file in Chapter 2.

Creating a Backup Control File

The backup control file is almost the same as a regular control file. It has some areas in it that are marked such that Oracle recognizes that it's a backup control file. When a backup control file is used, some form of recovery will be required (typically just involving the use of the archived and online redo logs if the database is otherwise intact).

To create the backup control file, simply issue the `ALTER DATABASE BACKUP CONTROLFILE TO` command, indicating at the end of the command where you want the control file to be created.

For example, if you wanted to create a backup control file after the online backup you performed in Exercise 1.4, you would simply need to issue the following command:

```
SQL> alter database backup controlfile to
'c:\backup\orcl\backup2\backup_control.ctl';
Database altered.
```

The result is the creation of a backup control file called `backup_control.ctl` found in the `c:\backup\orcl\backup2` directory, as you can see here:

```
SQL> host dir c:\backup\orcl\backup2\backup_control.ctl
 Volume in drive C has no label.
 Volume Serial Number is 08DE-E1AB
 Directory of c:\backup\orcl\backup2
```

```
08/02/2013  03:24 PM        10,174,464 BACKUP_CONTROL.CTL
               1 File(s)     10,174,464 bytes
               0 Dir(s)   9,930,571,776 bytes free
```

We will cover recovering from control-file loss using a backup control file in Chapter 2.

Creating a Trace File with the *Create CONTROLFILE* Command in It

If all else fails and you do not have a backup control file, don't worry; you have another option, the `CREATE CONTROLFILE` command. Normally, manually executing the command can be challenging because you need to know a lot of information about your database (like the names and locations of all the database data files). However, you can prepare for the possibility of having to use the `CREATE CONTROLFILE` command by creating one in advance. The `ALTER DATABASE BACKUP CONTROLFILE TO TRACE` command will create a trace file with the `CREATE CONTROLFILE` command in it for you. The trace file is stored in the new diagnostic directory structure in Oracle Database 12*c*.

The diagnostic directory structure is a new standard introduced in Oracle Database 12*c* that defines where Oracle stores files related to database troubleshooting and diagnostics. The base directory of this structure is defined by the parameter `DIAGNOSTIC_DEST`. Here is an example of the setting of `DIAGNOSTIC_DEST` on an Oracle Database:

```
SQL> show parameter diag
NAME                                 TYPE        VALUE
------------------------------------ ----------- ---------
diagnostic_dest                      string      C:\ORACLE
```

A whole book could be written on the new 12*c* diagnostic capabilities, but what we are interested in is where user-generated trace files get created because when we issue the `ALTER DATABASE BACKUP CONTROLFILE TO TRACE` command, the resulting file will be a user-generated trace file.

We would be remiss if we didn't mention that the database alert log is contained in the diagnostic destination directory structure. It is contained in the same directory as the trace files we are discussing in this section. Each database (or database instance if you are running RAC) will have its own alert log. The filename format for the alert log is `alert_{instance name}.log`. The alert log is an important diagnostic and monitoring tool, so you should become familiar with it.

In this case, the trace file will be created in a directory structure under `DIAGNOSTIC_DEST\diag\rdbms\orcl\orcl\trace`, as shown in this code example:

```
SQL> alter database backup controlfile to trace;
Database altered.
```

```
C:\oracle\diag\rdbms\orcl\orcl\trace>dir
 Volume in drive C has no label.
 Volume Serial Number is 08DE-E1AB
 Directory of C:\oracle\diag\rdbms\orcl\orcl\trace
08/02/2013  03:38 PM    <DIR>          .
08/02/2013  03:38 PM    <DIR>          ..
08/02/2013  03:38 PM         1,027,520 alert_orcl.log
08/02/2013  03:38 PM             9,572 orcl_ora_12120.trc
08/02/2013  03:38 PM                91 orcl_ora_12120.trm
               4 File(s)      1,037,183 bytes
               4 Dir(s)   9,964,507,136 bytes free
```

The trace file is called `orcl_ora_12120.trc` (it's easy to tell since there are no other trace files in the directory). If you have a number of trace files in the directory, you can sort the contents ascending by date and time and generally the correct trace file will be at the top (i.e., the LS -ALT command). Also, the `V$PROCESS` view has a TRACEFILE column in it that will display the name of the trace file for that process.

Another option with the `ALTER DATABASE BACKUP CONTROLFILE TO TRACE` command is to define an alternate location for the trace file. The syntax for this command is as follows:

```
alter database backup controlfile to trace as '/tmp/my_control_trace.trc';
```

If you look in the file, you will find a trace-file header in it first. Later down the trace file, you will find two different versions of the `CREATE CONTROLFILE` command. Here is an example of the `CREATE CONTROLFILE` command that you might find in this file:

```
CREATE CONTROLFILE REUSE DATABASE "ORCL" NORESETLOGS  ARCHIVELOG
    MAXLOGFILES 16
    MAXLOGMEMBERS 3
    MAXDATAFILES 100
    MAXINSTANCES 8
    MAXLOGHISTORY 292
LOGFILE
  GROUP 1 'C:\ORACLE\ORADATA\ORCL\REDO01.LOG'  SIZE 50M,
  GROUP 2 'C:\ORACLE\ORADATA\ORCL\REDO02.LOG'  SIZE 50M,
  GROUP 3 (
    'C:\ORACLE\ORADATA\ORCL\REDO03A.LOG',
    'C:\ORACLE\ORADATA\ORCL\REDO03B.LOG'
  ) SIZE 100M
-- STANDBY LOGFILE
DATAFILE
  'C:\ORACLE\ORADATA\ORCL\SYSTEM01.DBF',
  'C:\ORACLE\ORADATA\ORCL\SYSAUX01.DBF',
```

```
  'C:\ORACLE\ORADATA\ORCL\UNDOTBS01.DBF',
  'C:\ORACLE\ORADATA\ORCL\USERS01.DBF',
  'C:\ORACLE\ORADATA\ORCL\REVEAL_DATA_01.DBF',
  'C:\ORACLE\ORADATA\ORCL\REVEAL_INDEX_01.DBF',
  'C:\ORACLE\ORADATA\ORCL\USERS02.DBF'
CHARACTER SET WE8MSWIN1252;
```

You will notice that this output includes the data file names, the location and names of the online redo logs, and other information needed by the `CREATE CONTROLFILE` command. The trace file contains other output that will be required to complete the recovery process, so you should back up the trace file as it is. In Chapter 2, we will address the process of recovering from a control-file loss using the output contained in the trace files.

Exam objectives are subject to change at any time without prior notice and at Oracle's sole discretion. Please visit Oracle's Training and Certification website (`http://www.oracle.com/education/certification/`) for the most current exam-objectives listing.

Summary

Backups in Oracle can be a big deal. There are a number of different situations you will find yourself in, and finding a strategy to use depends on your knowledge of how to do backup and recovery. That's why the OCP exam has a great deal of backup and recovery focus.

We started this chapter introducing you to the Oracle architecture. This knowledge is critical to understanding what is required for any kind of Oracle database backup and recovery.

At the time we wrote this book, the Oracle 12*c* OCP exam contents with respect to manual backups and recovery specifically say "Backup and recover a NOARCHIVELOG database."

In this chapter we covered that particular topic in depth.

Then, we moved a little deeper to give you a firm understanding of more complex backup procedures, such as online backups and point-in-time recovery. We did this because later chapters will be discussing these kinds of backups and restores in the context of Oracle's backup tool called Recovery Manager (RMAN). You will understand RMAN operations much better if you understand the basic things that Oracle does under the covers to facilitate database recovery.

It is probably clear to you that recovery in Oracle can be quite complex. It becomes even harder when you are actually doing it under the gun, when people are breathing down your neck to get that database up. So, build your skills in this area now. They will serve you well in the future.

Exam Essentials

The following topics are specifically listed as essential test items and part of the content of the Oracle Database 12*c* OCP Exam.

Describe types of database failures. Understand the kinds of database failures that can occur. Also understand what causes those failures and what kind of recovery is required in the event those failures occur.

Describe tools available for backup and recovery tasks. Understand the various tools/methods (ie: shell scripts) that are available to backup and recover and Oracle database. Describe generally how each tool works.

Describe RMAN and maximum availability architecture. Understand and be able to describe the benefits of Oracle's Maximum Availability Architecture (MAA). Also be able to describe what RMAN is and what it's used for.

Configure your control file and online redo logs for recoverability. Understand the benefit of multiplexing the control files and the online redo logs. Understand why it is important to back up your control file. Understand the different methods of backing up your control file. Understand what a backup control file is.

Describe and tune instance recovery. Understand what instance recovery is and why it's critical to database availability. Describe how to monitor and configure the database for optimal instance recovery performance.

Perform backup and recovery in NOARCHIVELOG mode. Understand the steps that need to be performed to do an offline backup. Understand the difference between an offline backup and an online backup and how these differences can be used to decide the optimal backup strategy.

Enable ARCHIVELOG mode. Understand what ARCHIVELOG mode is, and how to configure the database so that it's running in ARCHIVELOG mode. Understand what archived redo logs are and why they are important, how they are created and where they are created.

Back up a control file to trace. Understand how to create a backup of the control file to a trace file, and where that trace file is located.

Review Questions

1. What are the different archiving modes available in the Oracle Database? (Choose all that apply.)
   A. LOGGING mode
   B. NOLOGGING mode
   C. ARCHIVELOG mode
   D. ARCHIVING mode
   E. NOLOGFILE mode
   F. NOARCHIVELOG mode
   G. RECOVERY mode
2. Your database is in NOARCHIVELOG mode. You start to do a backup, but your users complain that they don't want you to shut down the database to perform the backup. What options are available to you?
   A. Put the database in hot backup mode and perform an online backup, including backing up the archived redo logs.
   B. Just back up the database data files without shutting down the database.
   C. You will have to wait until you can shut down the database to perform the backup.
   D. Mark each data file as backup in progress, back them up individually, and then mark them as backup not in progress. No archived redo logs will need to be backed up.
   E. Back up only the data files that the user will not be touching. Once the user has finished what they were doing, you can shut down the database and back up the data files the user changed during the course of the remaining backup.
3. Your database backup scripts keep failing on the `ALTER TABLESPACE BEGIN BACKUP` command. Your junior DBA informs you that the failure is because the database is in ARCHIVELOG mode. Is he correct?
   A. Yes
   B. No
   C. There is insufficient information to decide the answer to this question.
4. What parameter is used to tune instance recovery in Oracle Database 12*c*?
   A. `INSTANCE_RECOVERY_TIME`
   B. `FAST_START_RECOVERY_TIME`
   C. `FAST_MTTR_RECOVERY`
   D. `FAST_START_MTTR_RECOVERY`
   E. None of the above

5. Which files do you need to back up for a database that is in ARCHIVELOG mode to ensure recovery? (Choose all that apply.)
   - **A.** Database data files
   - **B.** Online redo logs
   - **C.** Archived redo logs
   - **D.** Backup control file
   - **E.** Control file from a backup
6. What does the SCN represent?
   - **A.** The system change number, which is a marker indicating a point in time relative to transactions within a given database.
   - **B.** A number that represents time. Thus, at 1300 hours, the SCN is the same on all databases.
   - **C.** The security change number, which represents the security code that is needed to access any database structure.
   - **D.** A conversion factor that converts internal database time to external clock time.
   - **E.** UTC time in the database, providing a standardized way of tracking time in Oracle.
7. Which command(s) are used to back up the control file? (Choose all that apply.)
   - **A.** `ALTER SYSTEM BACKUP CONTROLFILE`
   - **B.** `ALTER DATABASE BACKUP CONTROLFILE TO TRACE`
   - **C.** `ALTER DATABASE BACKUP CONTROL FILE`
   - **D.** `ALTER DATABASE BACKUP CONTROLFILE TO 'FILENAME'`
   - **E.** `ALTER CONTROLFILE BACKUP`
8. What is the purpose of MAA?
   - **A.** MAA is a set of practices that Oracle has created to help you better administer your databases.
   - **B.** MAA is a set of practices that standardizes Oracle database creation, file naming conventions and operating system directory names.
   - **C.** MAA is a set of practices that align around providing maximum availability for Oracle databases.
   - **D.** MAA is a new form of Oracle martial arts.
   - **E.** MAA is an online document available from oracle.com that provides a list of current critical issues related to database backup and recovery.

9. Which of the following will cause instance recovery to occur? (Choose all that apply.)
   A. Power failure causes the database server to crash.
   B. You issue a `SHUTDOWN IMMEDIATE` command from the SQL prompt.
   C. You issue the `ALTER DATABASE CHECKPOINT` command.
   D. You issue the `SHUTDOWN ABORT` command from the SQL prompt.
   E. You issue the `SHUTDOWN TRANSACTIONAL` command from the SQL prompt.

10. What parameter controls how often the database checkpoints and also tries to reduce the time that instance recovery will take?
   A. `FAST_START_RECOVERY`
   B. `MTTR_TARGET`
   C. `RESTART_FAST`
   D. `CHECKPOINT_TIME`
   E. `FAST_START_MTTR_RECOVERY`

11. Which of the following database processes have some purpose in Oracle database operations that apply to database backup and recovery? (Choose all that apply.)
   A. DBW
   B. LGWR
   C. DNNR
   D. THREADn
   E. PQOn

12. What is most impacted by the delay between committing a transaction and the time that the data associated with that change is written to disk?
   A. Media recovery
   B. Archiving
   C. Instance recovery
   D. Transaction completion time
   E. Database creation time

13. Which of the following files are likely to be found in the diagnostic destination directory of an Oracle database?
   A. Database trace files
   B. Database data files
   C. Listener files
   D. Database parameter files
   E. Database alert log

14. Your database is ARCHIVELOG mode. Given the following steps, which option provides the steps to do this backup in the right order and the most efficiently?
    1. `ALTER DATABASE BEGIN BACKUP.`
    2. `ALTER TABLESPACE {TABLESPACE NAME} BEGIN BACKUP.`
    3. Back up the database data files.
    4. `ALTER DATABASE END BACKUP.`
    5. `ALTER TABLESPACE {TABLESPACE NAME} END BACKUP`
    6. Back up the online redo logs.
    7. Perform a redo log switch.
    8. Back up the archived redo logs.

    A. 2,3,4,6,8
    B. 1,2,3,5,4,7,8
    C. 1,3,4,7,8
    D. 8,7,4,3,1
    E. 1,7,8,9

15. From the answers available below, choose the two types of backups that can be taken of an Oracle database.
    A. Metadata backup
    B. Logical backup
    C. Physical backup
    D. Consistent backup
    E. Inconsistent backup

16. What database files would you need to back up if you are doing a NOARCHIVELOG recovery?
    A. Database data files
    B. Online redo logs
    C. Archived redo logs
    D. Database control file
    E. Database command file

17. What database view would you query to determine if the database is in ARCHIVELOG mode?
    A. `DBA_ARCHIVELOG_MODE`
    B. `V$LOG_MODE`
    C. `V$LOGGING_MODE`
    D. `V$ARCHIVELOG_MODE`
    E. `V$DATABASE`

18. What database view would you query to determine what the current archiving status of an online redo log is?
    A. `V$LOGFILE`
    B. `DBA_ARCHIVED_LOG`
    C. `DBA_ARCHIVLOG_HISTORY`
    D. `V$ARCHIVELOG_HISTORY`
    E. `V$ARCHIVED_LOG`

19. Which of the following are the names of an Oracle database process architecture? (Choose all that apply.)
    A. Multithreaded
    B. Multigenerational
    C. Multiprocess
    D. Multiplied
    E. Multitasked

20. Which of the following is not the name of an Oracle Database process?
    A. DBW
    B. LGWR
    C. ARCH
    D. ORCL
    E. All are database processes