CERTIFICATION OBJECTIVES

1.01 Explain Oracle Backup and Recovery Solutions

✓ Two-Minute Drill

Q&A Self Test
Automatic Storage Management (ASM) is a key Oracle Database technology that you should be using in your environment, even if it has only one database and one database instance. ASM is the key to all robust and complete backup solutions. The integration of the server file system and a volume manager built specifically for Oracle Database files makes your disk management and tuning task a breeze: Every file object is striped and mirrored to optimize performance. In addition, nearly all ASM volume management tasks can occur while a volume is online. For example, you can expand a volume or even move a volume to another disk while users are accessing it, with minimal impact to performance. The multiplexing features of an ASM cluster minimize the possibility of data loss and are generally more effective than a manual scheme that places critical files and backups on different physical drives. And if that isn’t enough, you can use an ASM instance and its disk groups to service more than one database instance, further optimizing your investment in disk hardware.

Before beginning a detailed explanation of how ASM works and how you can leverage it in your environment, this chapter discusses the available Oracle backup solutions and then offers a brief overview of the Oracle Database architecture, including instance memory structures, logical database structures, and physical database structures. A thorough understanding of the Oracle Database architecture (if you don’t already have this from previous coursework) is required to fully understand and appreciate how ASM works and how ASM contributes to Oracle’s extensive list of backup and recovery solutions.
this chapter gives an overview of the Oracle-recommended backup and recovery solutions available in Oracle Database 12c.

**Oracle Logical Storage Structures**

The datafiles in Oracle Database are grouped together into one or more tablespaces. *Datafiles* are physical structures that are subdivided into *extents* and *blocks*. Each *tablespace* is a little like a logical wrapper for one or more datafiles. Within each tablespace are finer-grained logical database structures, such as *tables* and *indexes*. Another term used is *segments*, which in Oracle Database is used to describe the physical space occupied by a table or an index. The way in which Oracle Database is compartmentalized allows for more efficient control over disk space usage. Figure 1-1 shows the relationship between the logical storage structures in a database.

**Tablespaces**

An Oracle *tablespace* consists of one or more datafiles; a datafile can be part of one and only one tablespace. For an installation of Oracle Database 12c, a minimum of two tablespaces must be created: the SYSTEM tablespace and the SYSAUX tablespace. A default installation of Oracle 12c creates six tablespaces.

Oracle Database 12c (since Oracle Database 10g) allows you to create a special kind of tablespace called a *bigfile tablespace*, which can be as large as 128 terabytes.
(TB). Using bigfiles makes tablespace management completely transparent to the database administrator (DBA); in other words, the DBA can manage the tablespace as a unit without worrying about the size and structure of the underlying datafiles.

Using Oracle Managed Files (OMF) can make tablespace datafile management even easier. With OMF, the DBA specifies one or more locations in the file system where datafiles, control files, and redo log files will reside, and Oracle automatically handles the naming and management of these files.

If a tablespace is temporary, only the segments saved in the tablespace are temporary; the tablespace itself is permanent. A temporary tablespace can be used for sorting operations and for table contents that exist only for the duration of the user's session. Dedicating a tablespace for these kinds of operations helps to reduce the I/O contention between temporary segments and permanent segments stored in another tablespace, such as tables.

Tablespaces can be either dictionary managed or locally managed. In a dictionary-managed tablespace, extent management is recorded in data dictionary tables. Therefore, even if all application tables are in the USERS tablespace, the SYSTEM tablespace will still be accessed for managing Data Manipulation Language (DML) on application tables. Because all users and applications must use the SYSTEM tablespace for extent management, this creates a potential bottleneck for write-intensive applications. In a locally managed tablespace, Oracle maintains a bitmap in the header of each datafile (inside a tablespace) to track space availability. Only quotas are managed in the data dictionary, dramatically reducing the contention for data dictionary tables.

Since Oracle Database 9i, if the SYSTEM tablespace is locally managed, then all other tablespaces must be locally managed if both read and write operations are to be performed on them. Dictionary-managed tablespaces must be read-only in databases with a locally managed SYSTEM tablespace.

**Blocks**

A database block is the smallest unit of storage in Oracle. The size of a block is a specific number of bytes of storage within a given tablespace, within the database.

To facilitate efficient disk I/O performance, a block is usually a multiple of the operating system block size. The default block size is specified by the Oracle initialization parameter DB_BLOCK_SIZE. Most operating systems will allow as many as four other block sizes to be defined for other tablespaces in the database. Some high-end operating systems will allow five block sizes. The blocks in the SYSTEM, SYSAUX, and any temporary tablespaces must be of the size DB_BLOCK_SIZE.
Explain Oracle Backup and Recovery Solutions

**Extents**
The extent is the next level of logical grouping in the database. An extent consists of one or more database blocks. When you enlarge a database object, the space added to the object is allocated as an extent. Extents are managed by Oracle at the datafile level.

**Segments**
The next level of logical grouping is the segment. A segment is a group of extents that form a database object that Oracle treats as a unit, such as a table or index. As a result, this is typically the smallest unit of storage that an end user of the database will deal with. Four types of segments are found in an Oracle database: data segments, index segments, temporary segments, and undo segments.

Every table in a database resides in a single data segment consisting of one or more extents; Oracle allocates more than one segment for a table if it is a partitioned table or a clustered table. Data segments include large object (LOB) segments that store LOB data referenced by a LOB locator column in a table segment (if the LOB is not stored inline in the table).

Each index is stored in its own index segment. As with partitioned tables, each partition of a partitioned index is stored in its own segment. Included in this category are LOB index segments. A table’s non-LOB columns, a table’s LOB columns, and the LOB’s associated indexes can all reside in their own tablespace (different segments) to improve performance.

When a user’s SQL statement needs disk space to complete an operation, such as a sorting operation that cannot fit in memory, Oracle allocates a temporary segment. Temporary segments exist only for the duration of the SQL statement.

From Oracle 10g onward, manual rollback segments exist only in the SYSTEM tablespace, and typically the DBA does not need to maintain the SYSTEM rollback segment. In previous Oracle releases, a rollback segment was created to save the previous values of a database DML operation in case the transaction was rolled back and to maintain the “before” image data to provide read-consistent views of table data for other users accessing the table. Rollback segments were also used during database recovery for rolling back uncommitted transactions that were active when the database instance crashed or terminated unexpectedly.

In Oracle Database 10g through 12c, Automatic Undo Management handles the automatic allocation and management of rollback segments within an undo tablespace. Within an undo tablespace, the undo segments are structured similarly to rollback segments, except that the details of how these segments are managed is
under control of Oracle, instead of being managed (often inefficiently) by the DBA. Automatic undo segments were available starting with Oracle Database 9i, but manually managed rollback segments were still available in Oracle 10g. However, this functionality is deprecated as of Oracle 10g and later releases. In Oracle Database 12c, Automatic Undo Management is enabled by default; in addition, a Procedural Language/Structured Query Language (PL/SQL) procedure is provided to help you size the UNDO tablespace.

If you’re starting out with Oracle Database 12c, all you really need to know is that manual rollback is redundant and will be unavailable in a future release. In addition, automatic undo is standard in Oracle Database 12c.

Oracle Physical Storage Structures

Oracle Database uses a number of physical storage structures on the disk to hold and manage the data from user transactions. Some of these storage structures, such as the datafiles, redo log files, and archived redo log files, hold real user data. Other structures, such as control files, maintain the state of the database objects. Text-based alert and trace files contain logging information for both routine events and error conditions in the database. Figure 1-2 shows the relationship between these physical structures and the logical storage structures reviewed in the section “Oracle Logical Storage Structures.” A database file is one of either a control file, datafile, or online redo log file. All other files that are not important to the successful running of the database include parameter files, password files, backup files, archived redo log files, trace files, and alert log files.

Datafiles

Oracle Database must contain at least one datafile. One Oracle datafile corresponds to one physical operating system file on the disk. Each datafile in Oracle Database is a member of one and only one tablespace; a tablespace, however, can consist of many datafiles. The exception is a bigfile tablespace, which consists of exactly one datafile.

An Oracle datafile can automatically expand when it runs out of space, if the DBA created the datafile with the AUTOEXTEND parameter. A datafile can be manually expanded using the ALTER DATABASE DATAFILE command. The DBA can limit the amount of expansion for a given datafile by using the MAXSIZE parameter. In any case, the size of the datafile is ultimately limited by the disk volume on which it resides.
The datafile is the ultimate resting place for all data in the database. Frequently accessed blocks in a datafile are cached in memory. Similarly, new data blocks are not immediately written out to the datafile but are written to the datafile depending on when the database writer process is active. Before a user’s transaction is considered complete, however, the transaction’s changes are written to the redo log files.

**Redo Log Files**
Whenever data is added, removed, or changed in a table, index, or other Oracle object, an entry is written to the current redo log file. Oracle Database must have at least two redo log files because Oracle reuses redo log files in a circular fashion. When one redo log file is filled with redo log entries, the current log file is marked as ACTIVE, if it is still needed for instance recovery, or INACTIVE, if it is not needed for instance recovery. The next log file in the sequence is reused from the beginning of the file and is marked as CURRENT.
Ideally, the information in a redo log file is never used. However, when a power failure occurs or some other server failure causes the Oracle instance to fail, the new or updated data blocks in the database buffer cache might not yet have been written to the datafiles. When the Oracle instance is restarted, the entries in the redo log file are applied to the database datafiles in a roll-forward operation to restore the state of the database up to the point at which the failure occurred.

To be able to recover from the loss of one redo log file within a redo log group, multiple copies of a redo log file can exist on different physical disks. Later in this chapter, you will see how redo log files, archived log files, and control files can be multiplexed to ensure the availability and data integrity of the Oracle database. Multiplexing, in a nutshell, means you have more than one, or many more than one, copies of a structure for performance and availability.

**Control Files**

Oracle Database has at least one control file that maintains the metadata of the database. Metadata in the control file is the data about the physical structure of the database itself. Among other things, the control file contains the name of the database, when the database was created, and the names and locations of all datafiles and redo log files. In addition, the control file maintains information used by Recovery Manager (RMAN), such as the persistent RMAN settings and the types of backups that have been performed on the database. Whenever any changes are made to the structure of the database, the information about the changes is immediately reflected in the control file.

Because the control file is so critical to the operation of the database, it can also be multiplexed (one or more control files can be copied). However, no matter how many copies of the control file are associated with an instance, only one of the control files is designated as primary for the purposes of retrieving database metadata.

The ALTER DATABASE BACKUP CONTROLFILE TO TRACE command is another way to back up the control file. It produces a SQL script that you can use to re-create the database control file in case all multiplexed binary versions of the control file are lost because of a catastrophic failure.

You can also use this trace file, for example, to re-create a control file if the database needs to be renamed or to change various database limits that could not otherwise be changed without re-creating the entire database.

**Archived Log Files**

Oracle Database can operate in one of two modes: ARCHIVELOG or NOARCHIVELOG. When the database is in NOARCHIVELOG mode, the
circular reuse of the redo log files (also known as the online redo log files) means that redo entries (the contents of previous transactions) are no longer available in case of a failure to a disk drive or another media-related failure. Operating in NOARCHIVELOG mode does protect the integrity of the database in the event of an instance failure or system crash because all transactions that are committed but not yet written to the datafiles are available in the online redo log files only. So, crash recovery is limited to entries currently in online redo logs. If your last backup of datafiles fails before your earliest redo log file, you cannot recover your database.

In contrast, ARCHIVELOG mode sends a filled redo log file to one or more specified destinations and can be available to reconstruct the database at any given point in time in the event that a database media failure occurs. For example, if the disk drive containing the datafiles crashes, the contents of the database can be recovered to a point in time before the crash, given the availability of a recent backup of the datafiles, the redo log files, and the archived log files that were generated since the backup occurred.

The use of multiple archived log destinations for filled redo log files is critical for one of Oracle’s high-availability features known as Oracle Data Guard, formerly known as Oracle Standby Database.

Initialization Parameter Files

When a database instance starts, the memory for the Oracle instance is allocated, and one of two types of initialization parameter files is opened: either a text-based file called init<SID>.ora (known generically as init.ora or a PFILE) or a server parameter file (SPFILE). The instance first looks for an SPFILE in the default location for the operating system ($ORACLE_HOME/dbs on Unix, for example) as either spfile<SID>.ora or spfile.ora. If neither of these files exists, the instance looks for a PFILE with the name init<SID>.ora. Alternatively, the STARTUP command can explicitly specify a PFILE to use for startup of Oracle.

Initialization parameter files, regardless of their format, specify file locations for trace files, control files, filled redo log files, and so forth. They also set limits on the sizes of the various structures in the System Global Area (SGA), as well as how many users can connect to the database simultaneously.

Until Oracle Database 9i, using the init.ora file was the only way to specify initialization parameters for the instance. Although it is easy to edit with a text editor, the file has some drawbacks. If a dynamic system parameter is changed at the command line with the ALTER SYSTEM command, the DBA must remember to change the init.ora file so that the new parameter value will be in effect the next time the instance is restarted.
An SPFILE makes parameter management easier and more effective for the DBA. If an SPFILE is in use for the running instance, any ALTER SYSTEM command that changes an initialization parameter can change the initialization parameter automatically in the SPFILE, change it only for the running instance, or both. No editing of the SPFILE is necessary or even possible without corrupting the SPFILE.

Although you cannot mirror a parameter file or SPFILE per se, you can back up an SPFILE to an init.ora file. Both the init.ora file and the SPFILE for the Oracle instance should be backed up using conventional operating system commands or, in the case of an SPFILE, using Recovery Manager.

When the Database Configuration Assistant (DBCA) is used to create a database, an SPFILE is created by default.

**Alert and Trace Log Files**

When things go wrong, Oracle can and often does write messages to the alert log and, in the case of background processes or user sessions, trace log files.

The alert log file, located in the directory specified by the initialization parameter BACKGROUND_DUMP_DEST, contains the most significant routine status messages as well as critical error conditions. When the database is started up or shut down, a message is recorded in the alert log, along with a list of initialization parameters that are different from their default values. In addition, any ALTER DATABASE or ALTER SYSTEM command issued by the DBA are recorded. Operations involving tablespaces and their datafiles are recorded here, too, such as adding a tablespace, dropping a tablespace, and adding a datafile to a tablespace. Error conditions, such as tablespaces running out of space, corrupted redo logs, and so forth, are also recorded here—all critical conditions.

The trace files for the Oracle instance background processes are also located in BACKGROUND_DUMP_DEST. For example, the trace files for PMON (process monitor) and SMON (system monitor) contain an entry when an error occurs or when SMON needs to perform instance recovery; the trace files for QMON (queue monitor) contain informational messages when it spawns a new process.

Trace files are also created for individual user sessions or connections to the database. These trace files are located in the directory specified by the initialization parameter USER_DUMP_DEST. Trace files for user processes are created in two situations: They are created when some type of error occurs in a user session because of a privilege problem, running out of space, and so forth. A trace file can be created for a user session on demand, to aid with problem diagnosis, with this command:

```
ALTER SESSION SET SQL_TRACE=TRUE;
```
Trace information is generated for each SQL statement that the user executes, which can be helpful when tuning a user's SQL statement.

The alert log file can be deleted or renamed at any time; it is re-created the next time an alert log message is generated. The DBA will often set up a daily batch job (through either an operating system mechanism, the Oracle Database internal scheduling mechanism, or Oracle Enterprise Manager's scheduler) to rename and archive the alert log on a daily basis.

As of Oracle Database 11g Release 1, the diagnostics for an instance are centralized in a single directory specified by the initialization parameter DIAGNOSTIC_DEST; USER_DUMP_DEST and BACKGROUND_DUMP_DEST are ignored.

Backup Files
Backup files can originate from a number of sources, such as operating system copy commands or Oracle RMAN. If the DBA performs a "cold" backup, the backup files are simply operating system copies of the datafiles, redo log files, control files, archived redo log files, and so forth.

In addition to bit-for-bit image copies of datafiles (the default in RMAN), RMAN can generate full and incremental backups of datafiles, control files, archived redo log files, and SPFILEs that are in a special format, called backupsets, readable only by RMAN. RMAN backupset backups are generally smaller than the original datafiles because RMAN does not back up unused blocks. RMAN is the standard for backup and recovery management, except in situations where RMAN backup processing has a detrimental effect on performance.

Oracle Memory Structures
Oracle uses the server's physical memory to hold many things for an Oracle instance: the Oracle executable code itself, session information, individual processes associated with the database, and information shared between processes (such as locks on database objects). In addition, the memory structures contain user and data dictionary SQL statements, along with cached information that is eventually permanently stored on disk, such as data blocks from database segments and information about completed transactions in the database. The data area allocated for an Oracle instance is called the System Global Area (SGA). The Oracle executables reside in the software code area. In addition, an area called the Program Global Area (PGA) is private to each server and background process; one PGA is allocated for each user session or server process.

Figure 1-3 shows the relationships between these Oracle memory structures.
System Global Area

The SGA is a group of memory structures for an Oracle instance, shared by the users of the database instance. When an Oracle instance is started, memory is allocated for the SGA based on the values specified in the initialization parameter file or hard-coded in the Oracle software. Many of the parameters that control the size of the various parts of the SGA are dynamic (can be changed immediately while the instance is running); however, if the parameter SGA_MAX_SIZE is specified, the total size of all SGA areas must not exceed the value of SGA_MAX_SIZE. If SGA_MAX_SIZE is not specified but the parameter SGA_TARGET is specified, Oracle automatically adjusts the sizes of the SGA components so that the total amount of memory allocated is equal to SGA_TARGET. SGA_TARGET is a dynamic parameter; it can be changed while the instance is running. The parameter MEMORY_TARGET, new as of Oracle 11g, balances all memory available to Oracle between the SGA and the PGA to optimize performance.
Memory in the SGA is allocated in units of granules. A granule can be either 4MB or 16MB, depending on the total size of the SGA. If the SGA is less than or equal to 128MB, a granule is 4MB; otherwise, it is 16MB. The next few subsections cover the highlights of how Oracle uses each section in the SGA.

### Buffer Caches

The database buffer cache holds blocks of data from disk that have been recently read to satisfy a SELECT statement or that contain modified blocks that have been changed or added from a DML statement. As of Oracle Database 9i, the memory area in the SGA that holds these data blocks is dynamic. This is a good thing, considering that there may be tablespaces in the database with block sizes other than the default block size. Oracle allows for tablespaces with up to five different block sizes (one block size for the default and up to four others). Each block size requires its own buffer cache. As the processing and transactional needs change during the day or during the week, the values of DB_CACHE_SIZE and DB_nK_CACHE_SIZE can be dynamically changed without restarting the instance to enhance performance for a tablespace with a given block size.

Oracle can use two additional caches with the same block size as the default (DB_CACHE_SIZE) block size: the KEEP buffer pool and the RECYCLE buffer pool. As of Oracle Database 9i, both pools allocate memory independently of other caches in the SGA.

When a table is created, you can specify the pool where the table's data blocks will reside by using the BUFFER_POOL_KEEP or BUFFER_POOL_RECYCLE clause in the STORAGE clause. For tables that you use frequently throughout the day, it would be advantageous to place the tables into the KEEP buffer pool to minimize the I/O needed to retrieve blocks in the tables.

### Shared Pool

The shared pool contains two major subcaches: the library cache and the data dictionary cache. The shared pool is sized by the SHARED_POOL_SIZE initialization parameter. This is another dynamic parameter that can be resized as long as the total SGA size is less than SGA_MAX_SIZE or SGA_TARGET.

The library cache holds information about SQL and PL/SQL statements that are run against the database. In the library cache, because it is shared by all users, many different database users can potentially share the same SQL statement.

Along with the SQL statement, the execution plan of the SQL statement is stored in the library cache. The second time an identical SQL statement is run, by the same user or a different user, the execution plan is already computed, improving the execution time of the query or DML statement.
Chapter 1: Database Backup Solutions and Automatic Storage Management

If the library cache is sized too small, then frequently used execution plans can be flushed out of the cache, requiring just as frequent reloads of SQL statements into the library cache.

The data dictionary is a collection of database tables, owned by the SYS and SYSTEM schemas, which contain the metadata about the database, its structures, and the privileges and roles of database users. The data dictionary cache holds a subset of the columns from data dictionary tables after first being read into the buffer cache. Data blocks from tables in the data dictionary are used continually to assist in processing user queries and other DML commands.

If the data dictionary cache is too small, requests for information from the data dictionary will cause extra I/O to occur; these I/O-bound data dictionary requests are called recursive calls and should be avoided by sizing the data dictionary cache correctly.

**Redo Log Buffer**  The redo log buffer holds the most recent changes to the data blocks in the datafiles. When the redo log buffer is one-third full, or every 3 seconds, Oracle writes redo log records to the redo log files. Additionally, as of Oracle Database 10g, the Log Writer (LGWR) process will write the redo log records to the redo log files when 1MB of redo is stored in the redo log buffer. The entries in the redo log buffer, once written to the redo log files, are critical to database recovery if the instance crashes before the changed data blocks are written from the buffer cache to the datafiles. A user’s committed transaction is not considered complete until the redo log entries have been successfully written to the redo log files.

**Large Pool**  The large pool is an optional area of the SGA. It is used for transactions that interact with more than one database, message buffers for processes performing parallel queries, and RMAN parallel backup and restore operations. As the name implies, the large pool makes available large blocks of memory for operations that need to allocate large blocks of memory at a time.

The initialization parameter LARGE_POOL_SIZE controls the size of the large pool and is a dynamic parameter.

**Java Pool**  The Java pool is used by the Oracle Java Virtual Machine (JVM) for all Java code and data within a user session. Storing Java code and data in the Java pool is analogous to SQL and PL/SQL code cached in the shared pool.

**Streams Pool**  Starting with Oracle 10g, the streams pool is sized by using the initialization parameter STREAMS_POOL_SIZE. The streams pool holds data and control structures to support the Oracle Streams feature of Oracle Enterprise Edition.
Oracle Streams manages the sharing of data and events in a distributed environment. If the initialization parameter STREAMS_POOL_SIZE is uninitialized or set to zero, the memory used for Streams operations is allocated from the shared pool and can use up to 10 percent of the shared pool.

**Program Global Area**  The PGA is an area of memory allocating dynamic sections of itself, privately for one set of connection processes. The configuration of the PGA depends on the connection configuration of the Oracle database: either *shared server* or *dedicated server*.

In a shared server configuration, multiple users share a connection to the database, minimizing memory usage on the server but potentially affecting response time for user requests. In a shared server environment, the SGA holds the persistent session information for a user instead of the PGA. Shared server environments are ideal for a large number of simultaneous connections to the database with infrequent or short-lived requests.

In a dedicated server environment, each user process gets its own connection to the database; the PGA contains the session memory for this configuration. The PGA also includes a sort area that is used whenever a user request requires a sort, bitmap merge, or hash join operation.

As of Oracle Database 9i, the PGA_AGGREGATE_TARGET parameter, in conjunction with the WORKAREA_SIZE_POLICY initialization parameter, can ease system administration by allowing the DBA to choose a total size for all work areas and let Oracle manage and allocate the memory between all user processes. As mentioned earlier in this chapter, the parameter MEMORY_TARGET manages the PGA and SGA memory as a whole to optimize performance. The MEMORY_TARGET parameter can help to manage the sizing of PGA and SGA as a whole. In general, PGA was automated starting in Oracle Database 9i. SGA was automated in 10g. Starting with Oracle Database 11g, the sum of SGA and PGA is now automated as well. Even experienced DBAs find that automated memory structuring is more effective at managing memory allocations.

**Software Code Area**  Software code areas store the Oracle executable files that are running as part of an Oracle instance. These code areas are static in nature and change only when a new release of the software is installed. Typically, the Oracle software code areas are located in a privileged memory area separate from other user programs.

Oracle software code is strictly read-only and can be installed as either sharable or nonsharable. Installing Oracle software code as sharable saves memory when
multiple Oracle instances are running on the same server, at the same software release level.

**Background Processes**

When an Oracle instance starts, multiple background processes start. A *background process* is a block of executable code designed to perform a specific task. Figure 1-4 shows the relationship between the background processes, the database, and the Oracle SGA. In contrast to a foreground process, such as a SQL*Plus session or a web browser, a background process works behind the scenes. Together, the SGA and the background processes make up an Oracle instance.
**SMON**  In the case of a system crash or instance failure due to a power outage or CPU failure, SMON, the *system monitor* process, performs crash recovery by applying the entries in the online redo log files to the datafiles. In addition, temporary segments in all tablespaces are purged during system restart.

One of SMON’s routine tasks is to coalesce the free space in tablespaces on a regular basis if the tablespace is dictionary managed (which should be rare or nonexistent in an Oracle Database or 12c database).

**PMON**  If a user connection is dropped or a user process otherwise fails, PMON, the *process monitor*, does the cleanup work. It cleans up the database buffer cache along with any other resources that the user connection was using. For example, suppose a user session is updating some rows in a table, placing a lock on one or more of the rows. A thunderstorm knocks out the power at the user’s desk, and the SQL*Plus session disappears when the workstation is powered off. Within milliseconds, PMON will detect that the connection no longer exists and perform the following tasks:

- Roll back the transaction that was in progress when the power went out
- Mark the transaction’s blocks as available in the buffer cache
- Remove the locks on the affected rows in the table
- Remove the process ID of the disconnected process from the list of active processes

PMON will also interact with the listeners by providing information about the status of the instance for incoming connection requests.

**DBWn**  The *database writer* process, known as DBWR in older versions of Oracle, writes new or changed data blocks (known as *dirty blocks*) in the buffer cache to the datafiles. Using a least recently used (LRU) algorithm, DBWn writes the oldest, least active blocks first. As a result, the most commonly requested blocks, even if they are dirty blocks, are in memory.

Up to 100 DBWn processes can be started: DBW0 through DBW9, DBWa through DBWz, and BW36 through BW99. The number of DBWn processes is controlled by the `DB_WRITER_PROCESSES` parameter.

**LGWR**  LGWR, or *Log Writer*, is in charge of redo log buffer management. LGWR is one of the most active processes in an instance with heavy DML activity. A transaction is not considered complete until LGWR successfully writes the redo
information, including the commit record, to the redo log files. In addition, the dirty buffers in the buffer cache cannot be written to the datafiles by DBWn until LGWR has written the redo information.

If the redo log files are grouped and one of the multiplexed redo log files in a group is damaged, LGWR writes to the remaining members of the group and records an error in the alert log file. If all members of a group are unusable, the LGWR process fails and the entire instance hangs until the problem can be corrected.

**ARCn** If the database is in ARCHIVELOG mode, then the archiver process, or ARCn, copies redo logs to one or more destination directories, devices, or network locations whenever a redo log fills up and redo information starts to fill the next redo log in sequence. Optimally, the archive process finishes before the filled redo log is needed again; otherwise, serious performance problems occur—users cannot complete their transactions until the entries are written to the redo log files, and the redo log file is not ready to accept new entries because it is still being written to the archive location. At least three potential solutions to this problem exist: make the redo log files larger, increase the number of redo log groups, and increase the number of ARCn processes. Up to 10 ARCn processes can be started for each instance by increasing the value of the LOG_ARCHIVE_MAX_PROCESSES initialization parameter.

**CKPT** The checkpoint process, or CKPT, helps to reduce the amount of time required for instance recovery. During a checkpoint, CKPT updates the header of the control file and the datafiles to reflect the last successful system change number (SCN). A checkpoint occurs automatically every time one redo log file fills and Oracle starts to fill the next one in a round-robin sequence.

The DBWn processes routinely write dirty buffers to advance the checkpoint from where instance recovery can begin, thus reducing the mean time to recovery (MTTR).

**RECO** The recoverer process, or RECO, handles failures of distributed transactions (that is, transactions that include changes to tables in more than one database). If a table in the CCTR (contact center) database is changed along with a table in the WHSE (data warehouse) database and the network connection between the databases fails before the table in the WHSE database can be updated, RECO will roll back the failed transaction.

**ASM Overview**

ASM is a multiplexing solution that automates the layout of datafiles, control files, and redo log files by distributing them across all available disks. When new disks are
added to the ASM cluster, the database files are automatically redistributed across all disk volumes for optimal performance. The multiplexing features of an ASM cluster minimize the possibility of data loss and are generally more effective than a manual scheme that places critical files and backups on different physical drives. One of the key components of an ASM disk is a disk group, a collection of disks that ASM manages as a unit.

When creating a new tablespace or other database structure, such as a control file or redo log file, you can specify a disk group as the storage area for the database structure instead of an operating system file. ASM takes the ease of use of OMF and combines it with mirroring and striping features to provide a robust file system and logical volume manager that can even support multiple nodes in an Oracle Real Application Cluster (RAC). ASM eliminates the need to purchase a third-party logical volume manager.

ASM not only enhances performance by automatically spreading out database objects over multiple devices but also increases availability by allowing new disk devices to be added to the database without shutting down the database; ASM automatically rebalances the distribution of files with minimal intervention.

The following sections review the ASM architecture, show you how to create a special type of Oracle instance to support ASM, and show how to start up and shut down an ASM instance.

**ASM Architecture**

ASM divides the datafiles and other database structures into extents, and it divides the extents among all the disks in the disk group to enhance both performance and reliability. Instead of mirroring entire disk volumes, ASM mirrors the database objects to provide the flexibility to mirror or stripe the database objects differently depending on their type. Optionally, the objects may not be striped at all if the underlying disk hardware is already RAID enabled, part of a storage area network (SAN), or part of a network-attached storage (NAS) device.

Automatic rebalancing is another key feature of ASM. When an increase in disk space is needed, additional disk devices can be added to a disk group, and ASM moves a proportional number of files from one or more existing disks to the new disks to maintain the overall I/O balance across all disks. This happens in the background while the database objects contained in the disk files are still online and available to users. If the impact to the I/O subsystem is high during a rebalance operation, the speed at which the rebalance occurs can be set using an initialization parameter.
ASM requires a special type of Oracle instance to provide the interface between a traditional Oracle instance and the file system; the ASM software components are shipped with the Oracle Database software and are always available as a selection when you’re selecting the storage type for the entire database while creating a database.

Using ASM does not, however, prevent you from mixing ASM disk groups with manual Oracle datafile management techniques. For example, you might have all of your tablespaces in ASM storage but have one tablespace created on your server’s file system to make it easier to transport to another database. Still, the ease of use and performance of ASM makes a strong case for eventually using ASM disk groups for all your storage needs.

Two Oracle background processes introduced in Oracle Database 10g support ASM instances: the rebalancer (RBAL) and ARBn. RBAL coordinates the disk activity for disk groups, performing rebalancing when a disk is added or removed. ARBn, where n can be a number from 0 to 9, performs the actual extent movement between disks in the disk groups.

For RDBMS instances that use ASM disks, two new background processes exist as of Oracle Database 10g: ASMB and RBAL. ASMB performs the communication between the database and the ASM instance, whereas RBAL performs the opening and closing of the disks in the disk group on behalf of the database. This is the same process as RBAL in an ASM instance but performs a different, but related, function. In other words, the process behaves differently depending on the type of instance.

**EXERCISE 1-1**

Find New ASM-Related Processes in ASM and RDBMS Instances

For this exercise, identify the new background processes on a Linux server for both the RDBMS instance and the ASM instance. On Linux, every Oracle process has its own thread. You can either join the Oracle dynamic performance views V$BGPROCESS and V$SESSION or use the Linux `ps -ef` command and search for command names containing either the ASM or the RDBMS instance names.

1. Query `/etc/oratab` for the name of the ASM and RDBMS instances:

```
[oracle@oel63 ~]$ tail /etc/oratab
#   # Multiple entries with the same $ORACLE_SID are not allowed.
#   #
+ASM:/u01/app/product/12.1.0/grid:N:          # line added by Agent
```
Explain Oracle Backup and Recovery Solutions

2. Set the ORACLE_SID environment variable for the RDBMS instance; in this case, choose `complref`:

   [oracle@dw ~]$ export ORACLE_SID=complref

3. Connect to the RDBMS instance and query V$SESSION and V$BGPROCESS to get the list of running processes:

   [oracle@dw ~]$ sqlplus / as sysdba

   SQL*Plus: Release 12.1.0.1.0 Production on Sun Feb 2 22:40:52 2014
   Copyright (c) 1982, 2013, Oracle.  All rights reserved.

   Connected to:
   Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 - 64bit Production
   With the Partitioning, Automatic Storage Management, OLAP, Advanced Analytics and Real Application Testing options

   SQL> select sid, serial#, process, name, description
       2> from v$session join v$bgprocess using(paddr);

   SID SERIAL# PROCESS    NAME           DESCRIPTION
   ---------- ---------- --------- ------ ---------------------------------
   --
       237      1 2886  PMON   process cleanup
       1       1 2893  VKTM   Virtual Keeper of TiMe process
      119      1 2897  GEN0   generic0
       . . .
       238      1 2899  MMAN   Memory Manager
       358      1 2912  DBW0   db writer process 0
       127      3 3005  TMON   Transport Monitor
       3       1 2914  LGWR   Redo etc.
      121      1 2916  CKPT   checkpoint
       6       7 3011  FBDA   Flashback Data Archiver Process
       4       1 2922  SMON   System Monitor Process
       362      1 3009  SMCO   Space Manager Process
       . . .
       360      1 2928  RBAL   ASM Rebalance master
       5       1 2930  ASMB   ASM Background
      124      1 2942  MARK   mark AU for resync coordinator
      123      1 2932  MMON   Manageability Monitor Process
      242      1 2934  MMNL   Manageability Monitor Process 2

   24 rows selected.
   SQL>

   Note the processes RBAL and ASMB near the end of the list.
Chapter 1: Database Backup Solutions and Automatic Storage Management

4. You can use the PID column to identify the Linux process number and query the Linux process directly:

   SQL> ps -f -p 2928
   UID   PID  PPID  C  STIME TTY          TIME CMD
   oracle 2928     1  0 21:31 ?        00:00:00 ora_rbal_complref
   SQL>

5. Next, check for the ASM background processes by setting the ORACLE_SID environment variable for the ASM instance (+ASM):

   [oracle@oel63 ~]$ . oraenv
   ORACLE_SID = [+ASM] ? +ASM

6. Connect to the ASM instance and query V$SESSION and V$BGPROCESS to get the list of running processes:

   [oracle@dw ~]$ sqlplus / as sysasm
   . . .
   SQL> select sid, serial#, process, name, description
   2>     from v$session join v$bgprocess using(paddr);

   SID  SERIAL# PROCESS  NAME                                DESCRIPTION
   ---------- ---------- -------- ------------------------------
   283      1 2810     PMON  process cleanup
   1        1 2814     VKTM  Virtual Keeper of TiMe process
   3        1 2832     CKPT  checkpoint
   4        1 2840     GMON  diskgroup monitor
   142      1 2818     GEN0  generic0
   425      1 2812     PSP0  process spawner 0
   145      1 2842     MMON  Manageability Monitor Process
   427      1 2830     LGWR  Redo etc.
   284      1 2820     MMAN  Memory Manager
   285      1 2828     DBW0  db writer process 0
   428      1 2838     RBAL  ASM Rebalance master
   287      1 2844     MMNL  Manageability Monitor Process 2
   144      1 2834     SMON  System Monitor Process
   143      1 2826     DIA0  diagnosibility process 0
   2        1 2824     DIAG  diagnosibility process
   286      1 2836     LREG  Listener Registration
   147      1 2868     ASMB  ASM Background
   17 rows selected.
   SQL>

   Note the processes RBAL and ASMB in the list. The ARBn process starts when a rebalance operation is initiated.
Database Failures and Backup Solutions

Eventually, your database will have some kind of failure whether it be a network failure, user error, disk drive failure, or memory corruption issue causing an instance failure. The best way to prepare for this nearly inescapable occurrence is to have a backup and recovery plan in place. This backup and recovery plan should also be tested on a regular basis to make sure that the recovery techniques will be successful if they are ever needed.

To create a successful backup and recovery strategy, you must first understand the types of failures and how Oracle responds to each of them: Some require no immediate user intervention, and some do. Which backup and recovery solutions you leverage depend on many factors, including how fast you need to recover from a failure and how many resources you want to dedicate to the solution. Your investment in recovery infrastructure is proportional to the cost of lost productivity or business income if the database is unavailable for a day, an hour, or a minute.

Failure Categories

The types of failures or errors you may encounter fall into two general categories: physical and logical. Physical errors are generally hardware errors or software errors in the applications using the database, while logical errors are typically at the end user level (database users or administrators). The categories of failures are as follows:

- **Statement failure**  A user’s SELECT or DML statement failed because of permissions, syntax, or resource limits.
- **User error**  The user mistakenly dropped a table or deleted the wrong rows in a table.
- **User process failure**  The connection to the database failed because of a client disconnect or unexpected shutdown.
- **Network failure**  As the name implies, the network connection between the client and the server (database) failed because of network hardware or protocol errors.
- **Instance failure**  The database instance crashed because of a bug, OS errors, memory corruption, or even power loss to the server.
- **Media failure**  There were disk drive physical errors or a controller hardware failure.

These failure categories are in the order of least serious and easiest to recover from to the most serious and harder to recover from. Specifically, statement failures are
the easiest to recover from since they are almost always user managed: Rewrite the SELECT or INSERT statement so it does not generate a syntax error! Accidentally dropping an important table can be recovered by the user or by the database administrator depending on whether the dropped table is still in the recycle bin or whether the DBA has given the user privileges to use various flashback features. Recovery from network, instance, or media failure will necessitate varying actions, the choice of which will depend on the criticality and severity of any lost data.

Your backup and recovery strategy will need to account for all of these failures, even though some of these failures are easier to recover from than others. For example, an instance failure may be because of a power outage. The recovery of committed transactions is relatively painless and automatic as long as your online redo log files are intact and multiplexed.

**Oracle Backup and Recovery Solutions**

Your *recovery time objective* (RTO) is the target in which a recovery operation must be completed to meet your customer’s or client’s service level agreement (SLA). Various Oracle tools and technologies will apply depending on the RTO:

- **Days or hours** RMAN (and optionally Oracle Secure Backup) can recover your database in days or hours if the entire database is lost because of natural disaster or disk failures.
- **Hours or minutes** Various flashback technologies, either user-initiated or DBA-initiated, can recover database objects usually while the database is still available to other users. Although the database needs to be temporarily shut down, by using a flashback database the DBA can roll back the entire database to a point in time earlier in the day or even weeks ago depending on the storage available in the fast recovery area and the restore points defined.
- **Minutes or seconds** If the database must be available continuously with downtimes no longer than a few minutes, Oracle solutions such as Data Guard or Active Data Guard can fail over to a backup database with minimal or no intervention from the DBA.
- **Recovery analysis** Regardless of the type of failure or RTO, the Oracle Data Recover Advisor makes it easy to quickly determine the type of failure and the fastest way to recover from a specific type of failure.

**Recovery Manager** Recovery Manager is the primary tool you use to back up, restore, and recover database objects from the table level (new to Oracle Database 12c)
to the datafile, tablespace, and of course database level. RMAN has many uses outside of backup and recovery, including the cloning or duplication of a database to another location.

A key component of RMAN is a special location for backup and recovery objects called the fast recovery area (FRA), explained in detail in Chapter 2. While this area is ideally a disk group in ASM, it can also be located in an operating system (OS) file system. Regardless of location, it is a centralized place for all backup and recovery objects. The FRA is managed based on size and your recovery objectives, whether that’s based on the recovery window or on the number of backups you need to retain.

**Oracle Secure Backup** In conjunction with RMAN, Oracle Secure Backup (OSB) will take RMAN backups from the FRA and copy them to a tape device or to cloud storage to prevent loss of data from a catastrophic failure at a data center. OSB also provides an extension to RMAN at the OS level to back up Linux servers as well as any attached storage such as in a NAS appliance.

**Oracle Data Guard** Oracle Data Guard is one of Oracle’s high-availability (HA) solutions to ensure near-real-time availability because of a failure of the primary database or to prevent database corruptions. A standby database receives archived redo log files from the primary database and maintains a secondary copy of the database that can be used in a number of scenarios beyond just disaster recovery. For example, during a scheduled maintenance window, the standby database can be switched to the role of primary database, while a new disk array is added to the primary database’s server. A standby database can also play the temporary role of a read-only copy of the database for reporting purposes and therefore free up resources on the primary database for better response time in an OLTP environment. This configuration is a special type of Data Guard configuration called Active Data Guard.

One primary database can update up to 30 standby databases (31 if you count the local archived redo log destination defined by the LOG_ARCHIVE_DEST_n parameter). One standby database can be updated in real time; another standby database can be updated with a lag time of 30 minutes or more to guard against the propagation of logical errors from the primary database.

Another type of standby database is called a logical standby database. Instead of continuously applying archived redo log files to a physical copy of the primary database, a logical standby database receives only the equivalent DML SQL statements that were submitted to the primary database. Therefore, the standby database is logically equivalent to the standby database but will almost certainly not have the identical physical structure of the primary database.
Chapter 1: Database Backup Solutions and Automatic Storage Management

CERTIFICATION SUMMARY

This chapter started with a review of the Oracle Database architecture. Some of this material has been covered in previous coursework, but a refresher course is always helpful for day-to-day database administration. More importantly, understanding the basic Oracle tablespace architecture is a prerequisite for understanding how ASM disk groups will store and manage Oracle tablespaces. In addition, understanding the basic Oracle background process architecture will dovetail nicely into a discussion of the ASM-related background processes available in both an ASM instance and an RDBMS instance.

Oracle provides a number of backup and recovery tools to meet the needs of any recovery time objective whether it be days, hours, or minutes. The tools center on Recovery Manager and include the fast recovery area and extensions to RMAN such as Oracle Secure Backup. Oracle Data Guard continuously protects a primary database by sending physical (archived redo log files) or logical (SQL DML statements) updates to one or more standby databases that can be used for both failover and offloading of reporting requests.
TWO-MINUTE DRILL

Explain Oracle Backup and Recovery Solutions

- Oracle database logical structures include tablespaces, segments, extents, and blocks, in order of increasing granularity.
- At a minimum, a database must have a SYSTEM tablespace and a SYSAUX tablespace.
- Oracle database physical structures include datafiles, redo log files, control files, archived log files, initialization parameter files, alert/trace files, and backup files.
- Oracle memory structures include the System Global Area, the Program Global Area, and the software code area.
- The primary Oracle background processes are SMON, PMON, DBWN, LGWR, ARCh, CKPT, and RECO.
- The background processes that support ASM instances are RBAL and ARBn; RDBMS instances that use ASM disks have the ASMB and RBAL background processes.
- ASM requires a dedicated instance for managing shared disks, called, not surprisingly, an ASM instance.
- Automatic rebalancing of disks in an ASM disk group happens in the background when disks are added or removed from an ASM disk group.
- The RBAL background process in an ASM instance coordinates disk activity for disk groups; the ARBn processes perform the actual extent movement between the disks in a disk group.
- The ASMB background process in an RDBMS instance performs the communication between the database and the ASM instance; the RBAL background process performs the opening and closing of the disks in the disk group for the RDBMS instance.
- An ASM instance has an initialization parameter file and a password file, but since there are no datafiles in an ASM instance, there is therefore no data dictionary; all connections to an ASM instance use operating system authentication.
The new SYSASM privilege in an ASM instance facilitates the separation of database administration and storage administration in an ASM instance.

Recovery Manager provides a flexible set of tools to back up and recover a database as well as to clone a database.

Oracle Secure Backup can copy RMAN backups, local file systems, and objects from NAS devices to tape or cloud backup.

Oracle Data Guard can send continuous changes from a primary database to up to 30 locations (standby databases) plus a local destination.

Standby databases can be physical or logical. Active Data Guard (physical) databases are maintained with archived redo log files, whereas a logical standby database uses SQL DML statements.

Active Data Guard databases can be used in read-only mode to support reporting requirements to free up resources on the primary database.
SELF TEST

The following questions will help you measure your understanding of the material presented in this chapter. Read all the choices carefully because there might be more than one correct answer. Choose all correct answers for each question.

**Explain Oracle Backup and Recovery Solutions**

1. Which of the following tablespaces are required in an installation of Oracle Database 12c? (Choose all that apply.)
   - A. USERS
   - B. SYSTEM
   - C. SYSAUX
   - D. TEMP
   - E. UNDOTBS1
   - F. RMAN

2. What is the maximum number of database writer processes (DBWn) in an Oracle database instance?
   - A. 1.
   - B. 100.
   - C. 20.
   - D. None; database writer processes exist only in an ASM instance.

3. Which of the following background processes exist in both an ASM instance and an RDBMS instance and also support ASM disk groups? (Choose all that apply.)
   - A. ASMB
   - B. RBAL
   - C. ARBn
   - D. LGWR
   - E. ARCn

4. At which level does ASM perform mirroring?
   - A. At the database object level.
   - B. At the tablespace level.
   - C. At the disk volume level.
   - D. ASM does not perform mirroring; it only supports disk hardware that is already RAID-enabled.
5. What is the value for INSTANCE_TYPE in the init.ora file or SPFILE for an ASM instance?
   A. RDBMS.
   B. ASM.
   C. +ASM.
   D. NOMOUNT.
   E. There is no such initialization parameter INSTANCE_TYPE.

6. You connect to an ASM instance with connected RDBMS instances as SYSOPER and run this command:

   SQL> shutdown immediate

   What happens?
   A. The ASM instance shuts down immediately, and all connected RDBMS instances shut down with the ABORT option.
   B. The ASM instance shuts down immediately, and all connected RDBMS instances shut down with the IMMEDIATE option.
   C. The command is ignored, since the SYSOPER privilege does not include starting up or shutting down an ASM instance.
   D. The ASM instance is not shut down because there is at least one connected RDBMS instance.

7. The value of the initialization parameter ASM_DISKGROUPS on your ASM instance is the following:

   DATA, RECOV, DATA2

   What happens when the ASM instance starts? (Choose the best answer.)
   A. Nothing happens until you issue ALTER DISKGROUP MOUNT commands.
   B. The ASM instance automatically mounts the disk groups, and you can manually mount any disk groups not in the list.
   C. ASM_DISKGROUPS is valid only for RDBMS instances.
   D. The disk devices DATA, RECOV, and DATA2 are available to create new disk groups.

8. Which of the following parameters are required for an ASM instance? (Choose all that apply.)
   A. INSTANCE_NAME
   B. INSTANCE_TYPE
   C. ASM_DISKGROUPS
   D. ASM_POWER_LIMIT
   E. ASM_PREFERRED_READ_FAILURE_GROUPS
9. Which of the following failures would be considered user errors? (Choose all that apply.)
   A. The intern just got a user account on the database and tries to update her own salary in the HR.EMPLOYEES table.
   B. Because of a power outage, the report server goes down during the overnight report batch window and is not able to generate most of the daily reports.
   C. Several users think the database has been upgraded to Oracle Database 12c and try to create a table with a VARCHAR2 column of more than 4,000 characters.
   D. The Linux administrator accidentally kills an OS process belonging to a database user who is trying to run a SELECT statement against the data warehouse.
   E. A data warehouse programmer enters the server room and removes a network card from the primary database server.

10. Which Oracle HA technology would be best suited for near-real-time failover in the case of a complete media failure of all disks in the primary database?
   A. Logical standby database
   B. Oracle Active Data Guard read-only access
   C. Oracle flashback database
   D. Oracle Active Data Guard physical standby
SELF TEST ANSWERS

Explain Oracle Backup and Recovery Solutions

1. ☑ B and C. Both the SYSTEM and SYSAUX tablespaces are required.
   ☒ A, D, E, and F are incorrect. While the USERS tablespace is highly desirable for placing application tables in its own tablespace, it is not required; TEMP, USERS, and UNDOTBS1 are created in a default installation of Oracle Database 11g. No RMAN tablespace is created, nor is it required in an installation of Oracle Database 11g.

2. ☑ B. The database writer processes are DBW0 through DBW9 and, if needed, DBWa through DBWz and BW36 through BW99 (for a total of 100) on most operating system platforms.
   ☒ A, C, and D are incorrect. Database writers exist only in an RDBMS instance.

3. ☑ B. Only the RBAL process exists in both ASM and RDBMS instances for ASM operations. RBAL coordinates the disk activity for disk groups in an ASM instance. RBAL performs the opening and closing of the disks in a disk group in an RDBMS instance, on behalf of the database.
   ☒ A, C, D, and E are incorrect. A is incorrect because ASMB exists only in an RDBMS instance that uses ASM disks. C is incorrect because ARBn exists only in an ASM instance and performs the extent movement between disks in disk groups. D is incorrect because LGWR exists only in an RDBMS instance and is not ASM related; it writes redo information to the online redo log files. E is incorrect because ARCn exists only in an RDBMS instance and is not ASM related; it writes online redo log files to archive redo log files when the database is in ARCHIVELOG mode.

4. ☑ A. ASM mirrors database objects only.
   ☒ B, C, and D are incorrect. ASM mirrors database objects to provide the flexibility to mirror or stripe each database object differently depending on their type. ASM does not need to mirror a given object if an underlying object is already mirrored by RAID hardware or the operating system.

5. ☑ B. As you might expect, INSTANCE_TYPE has a value of ASM for an ASM instance.
   ☒ A, C, D, and E are incorrect. A is valid only for an RDBMS (database) instance. C is the value for DB_UNIQUE_NAME in an ASM instance. D is an option to the STARTUP command. E is wrong because there is an initialization parameter called INSTANCE_TYPE and it defaults to RDBMS.

6. ☑ D. All connected RDBMS instances must be shut down before you can shut down an ASM instance with the IMMEDIATE option. If you stop an ASM instance with the ABORT option, all connected RDBMS instances are stopped.
   ☒ A, B, and C are incorrect. A is incorrect because RDBMS instances shut down with ABORT only if the ASM instance shuts down with the ABORT option or the ASM instance
10. **☑ D.** Oracle Active Data Guard physical standby continuously applies archived redo log files on one or more (up to 30) remote locations (standby locations) and can be configured to almost instantaneously take over the role of the primary database in case of a catastrophic failure of the primary database. Any standby location can be configured to apply the archived redo logs after a predefined delay to avoid potential logical corruptions to the database even if there is not a catastrophic failure of the primary database.

   **☒ A, B, and C** are incorrect. **A** is suitable for read-write access to report writers or developers but will not be an exact physical copy of the primary database. **B** is incorrect because using an Active Data Guard database for read-only queries is not providing a failover after a catastrophic failure but instead supplements the primary database for offloading some or all of the reporting workload. **C** is a viable option for recovering a database, tablespace, or individual database object to a previous state but does not provide real-time failover from a failure of the primary database.

---

**7. ☑ B.** The ASM instance automatically mounts the specified disk groups, and you can manually mount any disk groups not in the list.

   **☒ A, C, and D** are incorrect. **A** is incorrect because `ASM_DISKGROUPS` facilitates automatic mounting of the specified disk groups at startup. **C** is incorrect because `ASM_DISKGROUPS` is valid only for ASM instances. **D** is incorrect because the parameter `ASM_DISKGROUPS` contains existing disk groups, not raw devices available for disk groups.

---

**8. ☑ B.** Only the `INSTANCE_TYPE` parameter is required, and its value must be `ASM`.

   **☒ A, C, D, and E** are incorrect. `ASM_DISKGROUPS` can be empty, but then you must mount disk groups manually after starting an ASM instance. `ASM_POWER_LIMIT` defaults to 1 if it is not set; `ASM_PREFERRED_READ_FAILURE_GROUPS`, new to Oracle Database 11g, specifies a preferred failure group that is closest to the instance’s node to improve performance in a clustered ASM environment.

---

**9. ☑ A and C.** User errors are typically logical errors with SQL syntax, permissions on database objects, or trying to use features not available in the current version of the database.

   **☒ B, D, and E** are incorrect. **B** is a process failure since the client (in this case, the batch report generator) has failed and disconnects from the database. **D** is a user process failure except that the user’s process fails because of an OS administrator killing the incorrect process. If the OS administrator had killed a global database process, the failure would likely be considered an instance failure instead. **E** is a hardware or network failure, not a user failure. The user will likely be looking for a job somewhere else.

---

**10. ☑ D.** Oracle Active Data Guard physical standby continuously applies archived redo log files on one or more (up to 30) remote locations (standby locations) and can be configured to almost instantaneously take over the role of the primary database in case of a catastrophic failure of the primary database. Any standby location can be configured to apply the archived redo logs after a predefined delay to avoid potential logical corruptions to the database even if there is not a catastrophic failure of the primary database.

   **☒ A, B, and C** are incorrect. **A** is suitable for read-write access to report writers or developers but will not be an exact physical copy of the primary database. **B** is incorrect because using an Active Data Guard database for read-only queries is not providing a failover after a catastrophic failure but instead supplements the primary database for offloading some or all of the reporting workload. **C** is a viable option for recovering a database, tablespace, or individual database object to a previous state but does not provide real-time failover from a failure of the primary database.