Exam
Manual

LearnSmart

Oracle 10g
(1Z0-043) DBA II

This LearnSmart exam manual is designed to enhance your confidence and help you pass the Oracle 10g DBA II exam (IZ0-043). By studying this guide, you will become familiar with a wealth of exam-related topics and objectives, including:

- Using Globalization Support Objectives
- Securing the Oracle Listener
- Diagnostic Sources
- Flashback Database
- And more!

Give yourself the competitive edge necessary to further your career as an IT professional and purchase this exam manual today!

Oracle 10g DBA II (1Z0-043) LearnSmart Exam Manual

Product ID: 010335
Production Date: July 18, 2011
Total number of Questions: 25

Warning and Disclaimer

Every effort has been made to make this document as complete and as accurate as possible, but no warranty or fitness is implied. The publisher and authors assume no responsibility for errors or omissions. The information provided is on an "as is" basis. The authors and the publisher shall have neither liability nor responsibility to any person or entity with respect to any loss or damages arising from the information contained in this document.

Volume, Corporate, and Educational Sales

Favorable discounts are offered on all products when ordered in quantity. For more information, please contact us directly:

1-800-418-6789
solutions@preplogic.com

International Contact Information

International: +1 (813) 769-0920

United Kingdom: (0) 20 8816 8036

Table of Contents

Abstract

This Exam Manual prepares you for the Oracle 10g certification exam #1Z0-043,"Oracle Database 10g: Administration II." This test is popularly referred to as the "10g OCP Test." Passing it awards you the database certification title Oracle Certified Professional, or OCP. Many refer to this certification as the "OCP-DBA."

Topics on this exam include: backup and recovery (through RMAN, user-managed, and Enterprise Manager methods), Flashback Database and the various kinds of flashback recovery Oracle 10g supports, how to recover from user errors, handling block corruption, 10g's automatic database and storage management capabilities, globalization support, how to manage resources and use the new job Scheduler, monitoring and managing storage, and how to secure the Oracle listener and use the basic diagnostic tools.

What to Know

- Understand how to back up Oracle databases and their components. For backup and recovery tasks, know how to use Recovery Manager's (RMAN's) command line interface (CLI), the Enterprise Manager (EM), graphical user interface (GUI) panels, and user-managed (self-directed) commands.
- Understand the kinds of recovery situations you face when managing an Oracle database, and the basic ways to recover in each situation. These include recovering from the loss of one (or all) control files; recovering from the loss of one (or all) online redo log files; recovering from the loss of a non-critical datafile; recovering from the loss of a critical datafile (in the **SYSTEM** or **UNDO** tablespaces); and, recovering a database in an inconsistent state.
- Know the benefits and drawbacks of both Archivelog and Noarchivelog modes, and how to switch between them.
- Know the differences between complete and incomplete recovery, and how to perform both.
- Know all the basic RMAN commands. Know how to login to RMAN to issue these commands, and how to develop scripts of RMAN commands.
- Know what it means to recover a database using the **RESETLOGS** option. How does 10g now allow you to recover "across a **RESETLOGS?**"
- Understand Oracle's backup and recovery terminology thoroughly, and what the terms mean in real-world situations. For example: what are the two meanings of the word "recovery?" How do whole database backups, partial database backups, consistent backups, and inconsistent backups differ? What's the difference between an online redo log and an archived redo log? What's the difference between redo and undo? When you understand the finer points of difference between these definitions, you have reached the point where you are ready to take the exam.
- What is block corruption and how do you recover from it? What are the advantages and drawbacks of each recovery method? Know the commands for each approach.
- Understand the basics of automatic database management, including the automatic workload repository (AWR), automatic statistics collection and analysis, the automatic database diagnostic monitor (ADDM), and how these relate to the new 10g Advisors.
- Know how automatic storage management (ASM) works, how to set it up and how to use it. Along with ASM, know about monitoring and managing traditional database storage objects, such as tablespaces, segments, special kinds of tables (like index-organized tables or IOT's and clustered tables), and resumable space allocation.
- Understand Oracle's globalization support, and the aspects of the database that support international use of the system.
- Know how to use the Database Resource Manager, or DRM, to manage hardware resources and be able to use the new 10g job Scheduler to submit and manage tasks.

Tips

- The test contains both multiple-choice questions with one correct answer, and multiple-choice questions where you select 2 or 3 correct items out of a list of 5 or 6 possible answers. The latter are usually harder to answer correctly.
- Be sure to answer *every question* -- there is no penalty for guessing.
- Read each question carefully and read all alternatives before picking an answer. Often the differences between answers are very slight and you don't want to pick a wrong answer just because you didn't read the question carefully.
- The answers to many questions will not be immediately evident. In this case it helps to use a process of elimination. Eliminate answers you know are wrong, and you'll often be able to take your best guess from the remaining alternatives.
- You can always mark a question you're not sure about for later review. It is not unusual to find information in other test questions that will later help you answer the questions you've marked for review.
- You must know how to accomplish backup and recovery tasks in 3 ways: through RMAN's command line interface or CLI, through the new Enterprise Manager or EM GUI, and through self-directed means (referred to by Oracle Corp as user-managed backup and recovery).
- There is some overlap between this Exam and the OCA Exam (exam "1Z0-042 Oracle 10g Administration I"). The topics of overlapping coverage include:
    - Backup and Recovery.
    - Flashback features.
    - Automatic database management -- including the Automatic Workload Repository (AWR), Automatic Database Diagnostic Monitor (ADDM), and Automatic Shared Memory Management (ASMM).
- In the areas of Backup/Recovery and Flashback, this Exam probes much more deeply than the OCA Administration I test. You'll have to know a lot more detail about these two topics for this exam as opposed to the OCA Administration I exam.
- Regarding automatic database management, the level of detail you're expected to know in this test is about the same as in the OCA Exam. Of course, the questions will differ.

To pass any Oracle exam, you need to do four things –

1. Verify Oracle Corp's exam requirements
2. Get hands-on experience
3. Study this exam manual, written specifically to help candidates pass the test
4. Work with practice questions

Verify Oracle Corp's Exam Requirements:
To verify Oracle Corp's exam requirements, go to their certification home page at www.oracle.com/education/certification, or go directly to the page for this test at http://education.oracle.com/pls/web_prod-plqdad/db_pages.getpage?page_id=47#2.

You need to verify the exam requirements because Oracle reserves the right to change them at any time (usually they only do so when a new release comes out, but if you're going to put in the effort to pass this test you'll want to make sure). The web site also lists the exact topics on the test, information about how long the test takes and how many questions it contains, and where and how to sign up to take the test.

Hands-on Experience:
To get hands-on experience with Oracle 10g, either get it at work, or take advantage of Oracle's free 10g Database download. Go to http://www.oracle.com/database/index.html to get the free download, or go to www.oracle.com and enter "free download" in the Search Box. You can install the product on your Windows or Linux PC. The free license will give you what you need to pass the test.

Backup and Recovery

Terminology

Understanding Oracle Corp's backup and recovery terminology may appear too basic to the uninitiated. But, in fact, if you thoroughly understand Oracle's backup and recovery terminology you are well on the way to understanding how backup and recovery works for Oracle databases.

Oracle backups may be consistent (with all internal System Change Numbers or SCNs synchronized). This means the database, its control files and datafile headers are all in sync and represent a single, common transactional state of the database. Consistent backups (cold backups) are made with the database down or "offline," so they are also called offline backups.

Inconsistent, or online backups, are made while the database is up (hot backups). Since control file SCNs may vary from datafile header SCNs, this kind of backup is first restored (copied in from the flash area or tape backup), then recovered (by applying the logs to bring the database to a point of internal consistency). At the conclusion of recovery, all SCNs match.

A whole database backup includes all datafiles and the control file (online redo logs are never backed up). A partial database backup lacks all the files required to make it a whole database backup. Note that a whole database backup may -- or may not -- be a consistent database backup. In other words, the presence of all files in a database backup does not imply anything about whether the transactional state of all those files is consistent.

A full backup includes all blocks of every datafile included in a whole or partial database backup. This differs from a whole database backup -- the full backup does *not* necessarily mean that the complete database has been backed up. In contrast to a full backup, an incremental backup makes a copy of all data blocks changed since a previous backup. Incremental backups or "incrementals" come in 5 levels, from the level 0 or baseline backup, up through the level 4 backup.

A differential incremental backup backs up only data blocks modified since the most recent backup at the same level or lower. It is the default kind of incremental backup.

A cumulative incremental backup is a type of incremental backup that backs up all the data blocks that have been changed since the most recent backup of the next lower level. If this backup is level **n**, then the cumulative incremental backup will backup the data blocks changed since level **n-1** or lower. A cumulative incremental backup is useful because having it may mean you will be able to apply fewer incremental backups when recovering a database, thus reducing the mean time to recover (MTTR). The MTTR is one of several key measures of backup success and may be part of a service level agreement (SLA) with the user community.

A target database is any database that has been backed up so that it may later be recovered. When a target database becomes lost or damaged in some way, we talk about *recovering the database*. This is a loose way of saying we need to fix some damage that has occurred to data or other files in the database. Be aware that Oracle Corp. also has another, very specific use of the word recover. In Oracle's terminology, to fix a damaged database, you will often perform these 2 steps:

1. Restore data from backup (copy data over from a backup file of some sort, possibly transforming it into the proper format for a database datafile, and putting it into proper location as a datafile within the database).
2. Recover the database by applying redo log data to bring any lost datafiles up to consistency with the rest of the database.

So, as used here, recover specifically means to apply logs to one or more lost datafiles in order to bring them to a point of consistency (to the same System Change Number or SCN as the rest of the database). "Recovering a database" in the general sense means performing the two specific steps of *restoring* and then *recovering* datafiles.

Recall that every Oracle database runs in either of two possible modes:

- Archivelog mode
- Noarchivelog mode

When a database is in *Archivelog* mode, if the current online redo log group members become filled up with redo entries, a log switch occurs. At this point, the Archiver background process (ARCn) steps in and writes the contents of full online redo log file members to archived redo logs. This preserves all redo records.

If a database is in *Noarchivelog* mode, no Archiver process runs, and the full online redo logs are *not* archived by ARCn. During a log switch, redo information is over-written and lost.

Archivelog mode means that no transactions are ever lost. A database or its components can be recovered to any point in time, including up to the last transaction committed (also called a recovery to currency). Databases or their components can be backed up while online and accessible to users. Recoveries of datafiles outside of the critical **SYSTEM** and **UNDO** tablespaces can be performed online, while the rest of the system is up and available to users.

Noarchivelog mode databases require that the entire database be recovered to a specific point in time -- the time at which the database backup was made. This could result in lost transactions. The entire database must be shut down to perform this recovery. (This contrasts to Archivelog mode, where many kinds of recovery permit the database to be up and operating while recovery occurs). You must shut down Noarchivelog mode databases to back them up.

To change the archivelog mode:

1. If using a textual parameter file (**PFILE**), set parameter **LOG_ARCHIVE_START = TRUE**. If using a binary parameter file (**SPFILE**), issue:
   SQL> alter system set log_archive_start=true scope=spfile;
2. Shutdown the database
3. **STARTUP MOUNT** (you change the archivelog mode in the **MOUNT** state)
4. Issue **ALTER DATABASE ARCHIVELOG** or **ALTER DATABASE NOARCHIVELOG**
5. Open the database by **ALTER DATABASE OPEN**

Set the archivelog destination(s) by initialization parameter **LOG_ARCHIVE_DEST_n**. Name the archive logs by setting initialization parm **LOG_ARCHIVE_FORMAT**. Determine the reliability level for the writing of the archive logs by init parm **LOG_ARCHIVE_MIN_SUCCEED_DEST**.

Backup and Recovery Alternatives

You can work with backup and recovery through 4 basic mechanisms in Oracle:

1. **User-managed**. This means that you manage backup and recovery yourself, manually, and issue any necessary commands to do it. You hand-code any scripts you want to automate backup and recovery tasks, which use operating system commands such as **copy** or **cp**, and **compress**, for example.
2. **Recovery Manager (RMAN)**. RMAN is Oracle Corp's bundled backup and recovery tool. Use RMAN if you work on the command line or write (generate) scripts. This is Oracle Corp's recommended approach.
3. **Enterprise Manager (EM)**. EM is the 10g GUI that runs from within a web browser. It includes complete provisions for backup and recovery and is the easiest interface to use. You can learn how it operates by pressing its **Show SQL** button on many screens, which displays the line-oriented SQL or RMAN commands the EM generates to implement the actions you selected on the GUI panels. EM is really a web-based GUI interface into RMAN, when it comes to backup and recovery.
4. **Export Utility**. The Export utility performs backups on logical database entities (like tablespaces or tables). It creates logical backups, as opposed to the physical backups you get when you back up physical database files (by copying the datafiles, control files, and archived redo logs). 10g gives you the new **expdp**/**impdp** utilities. These have more capabilities than and supersede the older **exp**/**imp** utilities of Oracle 9i and previous releases. (The older **exp**/**imp** utilities are still present in 10g if you want them).

The exam emphasizes RMAN and its line commands (the test developers correctly figure that if you know how to accomplish tasks using RMAN's command-line interface (CLI) you can perform those same tasks through the easier EM GUI). The Exam does not much cover user-managed and Export methods outside of the few words we give them here.

This chart gives you a rough comparison of the basic features of the main backup/recovery methods. EM capabilities are basically the same as RMAN, since the EM is really just an RMAN GUI interface when it comes to backup and recovery:

Feature:	User-managed:	RMAN:	Exports:
Password file backups	Yes	Not supported	Not supported
PFILE backups	Yes	Not supported	Not supported
SPFILE backups	Yes	Yes	Not supported
Closed database backups	Yes	Yes	Not supported
Open database backups	Yes	Yes	Yes
Incremental backups	Not supported	Yes	Not supported
Detects corrupt blocks	Not supported	Yes	Yes
Automatic backup file	Not supported	Yes	Yes

Backup catalogs	Not supported	Yes	Yes
Media Manager	Yes	Yes	Yes
Platform independent	Not supported	Yes	Yes

This chart shows that RMAN backs up everything in an Oracle database for you except:

1. Password files
2. Text-based initialization parameter files (**PFILE**s)
3. The Oracle binaries themselves

Most sites that use RMAN easily pick up these items in backups by relying on operating-system level disk backups to back them up. But RMAN itself does not back them up.

RMAN Components and Set-up

Recovery Manager (RMAN) is the Oracle database's backup and recovery feature. Its basic components are:

- **EM's web-based GUI interface** – This GUI into RMAN generates RMAN line commands and scripts as needed to carry out the user's directives.
- **RMAN line commands** – A complete set of over 50 commands managing all aspects of Oracle backup and recovery.
- **RMAN Recovery Catalog** – An Oracle database that provides centralized control for backup and recovery. The RMAN recovery catalog exists as a set of tables and tablespaces that may be placed in their own dedicated "RMAN database," or they may co-reside with other applications within any other Oracle database you choose. The Recovery Catalog is sometimes referred to as the "RMAN Catalog" or just as the "Catalog."
- **Media Management Layer (MML)** - The RMAN MML is a component that provides a generic interface (or application programming interface, or API) into external software that controls tape drives and automated tape libraries (ATLs). Oracle Corp does not provide the tape management software; 3rd parties do, and the MML is the RMAN interface into the 3rd party software.

The RMAN Recovery Catalog is optional but most sites use it since it provides extra RMAN features not otherwise available. The only benefit to *not* using it is that you don't have the overhead of supporting it and you save a few hundred megabytes of disk space. If you do not use an "RMAN Catalog," Oracle stores necessary backup/recovery information for any given target database in that database's control files. This causes the control file to grow in size. The initialization parameter **CONTROL_FILE_RECORD_KEEP_TIME** tells how long RMAN info is kept in the control file. It defaults to 7 days and has a maximum of 365 days.

Control files consist of non-reusable and reusable sections. The reusable sections include both circular reuse records and non-circular reuse records. The circular reuse records contain non-critical information that can be over-written. The non-circular reuse records include information on datafiles and redo logs for the target database.

If you use an RMAN Recovery Catalog it contains:

- Backup and recovery information about target databases, including their datafiles and archive logs
- Physical or structural information about target databases
- Scripts, programs of RMAN commands that can be stored, modified, re-used and potentially applied to multiple target databases

The RMAN Catalog is preferably stored in a separate database than any of its target databases and on a separate server.

To create a Recovery Catalog, login to the database server on which you want it to reside, and follow these steps:

1. Create an RMAN user account:

```
SQL> connect / as sysdba
SQL> create  user  rman_user identified by  rman_user
          default  tablespace  rman_data
          temporary  tablespace  temp;
```

2. Grant required permissions to the RMAN user account (including **RECOVERY_CATALOG_OWNER**):

```
SQL> grant connect, resource, recovery_catalog_owner to rman_user;
```

3. Start the RMAN utility and connect as the RMAN user id:

```
c:>  rman
RMAN>  connect catalog  rman_user/rman_user ;
```

4. Create the recovery catalog by the RMAN **CREATE CATALOG** command. Specify the tablespace you want to store the catalog in:

```
RMAN>  create  catalog  tablespace  rman_data ;
RMAN>  exit ;
```

For any database you want to backup through RMAN (any target database), you must first register that database to RMAN. Another way of saying this is that you will create an incarnation of that target database in the catalog (a reference to a version of that database). These steps show how to start the RMAN utility and connect to the target database, how to connect to the recovery catalog, and how to register the target database with the recovery catalog:

1. c:> rman target / (the slash indicates that you are connecting rman to a target database on the same system you are running rman from)
2. RMAN> connect catalog rman_user/rman_user@orarc ;
3. RMAN> register database ;

orarc is the host server where the recovery catalog resides in this example, as accessed through SQL*Net.

When using RMAN, you can connect to the target database only, the recovery catalog database only, or to both the target and catalog databases. This results in some confusing syntax so we summarize it here with a few connection examples:

Example Connection:	**What It Does:**
c:> rman	Starts RMAN utility only, does not connect to any database
c:> rman target /	Starts RMAN, connects to target database only on the local system
c:> rman target / nocatalog	Starts RMAN, connects to target database only (**nocatalog** is the default) on the local system
c:> rman target / catalog rman_user/rman_user@orarc	Starts RMAN, connects to both a target database on the local system and a recovery catalog (specify which recovery catalog following the keyword **catalog**)
RMAN> connect target	Connects to a target database from within RMAN
RMAN> connect catalog rman_user/rman_user@orarc	Connects to a recovery catalog database from within RMAN

You need the **SYSDBA** privilege to connect to and work with a target database. RMAN can also connect to an auxiliary database. This is a standby database, duplicate database, or auxiliary instance (for standby or tablespace point in time recovery (TSPITR)).

RMAN Commands

RMAN has over 50 commands you can issue through the RMAN command line, or that you can automatically generate and issue through the EM GUI panels. We'll walk through some example RMAN commands shortly.

The chart below provides a reference for basic commands. The RMAN commands are listed in upper case while command clauses are in mixed case. You will have to be at least familiar with all these commands for the test:

RMAN Command (or clause)	**Usage:**
@	Runs a command file
@@	Runs a command file in the same directory from which another command file is already running
ALLOCATE CHANNEL	Creates a channel, a connection between RMAN, the target instance, and disk or tape storage

ALLOCATE CHANNEL FOR MAINTENANCE	Allocates a channel for subsequent RMAN maintenance commands
allocOperandList	Clause for specifying channel control options
ALTER DATABASE	Mounts or opens a database
archivelogRecordSpecifier	Specifies a range of archive redo logs
BACKUP	Backs up database files or copies of them, archived logs, or backup sets
BACKUP AS COPY	Backs up database files as image copies (exact duplicates) for quicker recovery
BACKUP AS COMPRESSED	Backs up database files in compressed format
BLOCKRECOVER	Recover individual data block(s)
CATALOG	Adds info to the Recovery Catalog
CHANGE	Changes info in the Recovery Catalog
completedTimeSpec	Specifies a time period in which the backup or copy completes
CONFIGURE	Configures RMAN
CONNECT	Connects RMAN utility to a target database, an auxiliary database, or a RMAN recovery catalog
connectStringSpec	Connect string for connecting to a database
CONVERT	Converts datafile formats for transportable tablespaces across platforms
CREATE CATALOG	Creates the recovery catalog
CREATE SCRIPT	Stores a script in the recovery catalog. Scripts are stored as editable text files of type ***.rcv**
CROSSCHECK	Determines whether RMAN-managed files still exist
datafileSpec	Specifies a datafile by name or number
DELETE	Deletes backups or copies from the recovery catalog
DELETE SCRIPT	Deletes a script in the recovery catalog
deviceSpecifier	The type of storage device for a backup or copy
DROP CATALOG	Drops the recovery catalog
DROP DATABASE	Drops the target database and un-registers it

DUPLICATE	Uses backups to create a duplicate database
EXECUTE SCRIPT	Runs a script in the recovery catalog
EXIT or QUIT	Exit the RMAN utility
fileNameConversionSpec	Specifies how to transform source to target filenames during **BACKUP AS COPY**, **CONVERT**, and **DUPLICATE**
FLASHBACK	Puts the database back to a previous point in time or System Change Number (SCN)
FORMAT formatSpec	The **BACKUP** command parameter that specifies the pattern for filenames for backups or copies
HOST	Allows you to run a host operating system command from within the RMAN utility
keepOption	Whether a backup or copy should be exempt from the current retention policy
LIST	Lists backup sets and copies in a report
listObjList	Items to list via the **LIST** command
maintQualifier	Specifies options for maintenance commands like **CHANGE** and **DELETE**
maintSpec	Specifies the files used in a **CHANGE**, **CROSSCHECK** or **DELETE** command
obsOperandList	A subclause for obsolete backups and copies
PRINT SCRIPT	Displays a stored script
recordSpec	Which objects maintenance commands apply to
RECOVER	Applies redo logs to recover datafiles
REGISTER	Registers the target database with the recovery catalog, for subsequent backups, etc.
RELEASE CHANNEL	Releases a channel previously dedicated by an **ALLOCATE CHANNEL** command
releaseForMaint	Releases a channel allocated for maintenance
REPLACE SCRIPT	Replaces a stored script, or adds it, if new
REPORT	Reports recovery catalog contents

RESET DATABASE	Sets the incarnation of a target database (a state or version of the database after a database **OPEN RESETLOGS** command has been executed)
RESTORE	Brings in files from backups to their proper disk locations in the database
RESYNC	Resynchronizes the snapshot control file with the recovery catalog
RUN	Runs one or more RMAN commands enclosed in braces **{ }**
SEND	Sends a quoted string to a channel
SET	Sets any of many RMAN attributes for a session
SHOW	Shows current configuration settings
SHOW ALL	Shows all the current configuration settings
SHUTDOWN	Shuts down the target database
SPOOL	Writes RMAN output to a file
SQL	Executes a SQL statement within RMAN
STARTUP	Starts up the target database
SWITCH	Switches a datafile copy to be the current datafile
TAG	**BACKUP** command parameter that names the backup file
UNREGISTER DATABASE	Un-registers a target database in the recovery catalog
untilClause	Specifies a time, SCN, or log sequence number
UPGRADE CATALOG	Upgrades the recovery catalog to a newer version
VALIDATE	Validates a backup set

RMAN Backups and Examples

RMAN backs up database files in either of two ways:

- As image copies - Duplicates of datafiles.
- As backup sets - Backups in an RMAN-internal format.

Image copies can be quicker for recovery because they must only be copied to the relevant disk location to be used (they need not be translated out of the internal format RMAN uses for backup sets when re-covering). But image copies cannot be compressed and require more disk space than backup sets. Image copies can only be written to disk (not to tape) via rman.

Backup sets are logical groupings of files in RMAN's proprietary format. A backup piece is the actual file within the backup set. Backup sets may be either written to disk or tape. They may optionally be compressed. Compression ratios vary by the data contained in the database but compression ratios of 5:1 are typical. So a compressed backup set might only be 20% of the size of the same backup set in un-compressed format. You must process a backup set with the RMAN **RESTORE** command before it is usable in a recovery situation (image copies do not require **RESTORE** because they're already in a ready-to-use format).

Backup sets are inherently multiplexed (image copies are not). Multiplexing means that multiple input files are read and then may be written out to a single backup piece. The blocks from the various datafiles read as input may be interspersed in the output backup.

This example code uses the **BACKUP AS COPY** command to back up an entire database in image copy format. The keyword **TAG** assigns the *tag* or name of "101005 backup**"** to the backup. Of course, you must connect RMAN to the target database prior to backing it up:

```
RMAN> connect target
RMAN> backup as copy tag "101005 backup" database ;
```

This command backs up only the **SYSTEM** datafile (datafile number 1) and the control file to image copies:

```
RMAN> run { allocate channel ch1 type disk ;
            copy datafile 1 to 'c:\oracle\bkups\SYSTEM01.DBF' ,
                 current controlfile to 'c:\oracle\bkups\CONTROL01.CTL' ; }
```

In this case we used the older, Oracle 9i RMAN **COPY** command (still supported) instead of the newer, preferred **BACKUP AS COPY** command. We just list the files we want to image copy and the location to which to write them on disk. The new 10g **BACKUP AS COPY** is better than Oracle 9i's **COPY** command because now you can create image copies of the entire database, multiple tablespaces, datafiles, or archive logs without having to list all the individual files.

We used the RMAN **RUN** command in order to execute the series of RMAN commands enclosed within the **RUN** command's braces (**{ }**). This allows you to group and run a series of RMAN commands as a single block. You can save **RUN** blocks as scripts in the Recovery Catalog for use later, perhaps with other target databases. Manipulate scripts in the RMAN catalog by the **CREATE SCRIPT**, **EXECUTE SCRIPT**, **DELETE SCRIPT**, **REPLACE SCRIPT**, and **PRINT SCRIPT** RMAN commands.

The **ALLOCATE CHANNEL** command establishes a channel, or connection, between RMAN and the instance and is used as the conduit for the copy. The channel also specifies an output type of disk or tape. We named the channel **ch1** in the example below. The keywords **TYPE DISK** indicate that the backup will be written to disk (as opposed to tape or other media).
Here is an example backup for a complete database backup written to backup sets:

```
RMAN> run {
            allocate channel ch1 type disk ;
            backup database format 'db_%d_%u_%s' tag monthly_backup ;
            backup format 'log_t%t_s%s_p%p'
                  (archivelog all) ;      }
```

The **ALLOCATE** channel command allocates a single channel for this backup to write to disk. **BACKUP DATABASE** backs up the entire database as a backup set. The first **FORMAT** keyword assigns a naming convention for the backup pieces. The second **FORMAT** assigns the naming convention for the backup pieces for the archive logs.

The keywords **(ARCHIVELOG ALL)** ensure that all archive logs are backed up. When you specify that archive logs be backed up like this, RMAN performs the backup in this order:

1. Back up the database
2. Archive all active online redo logs
3. Back up all available archive logs in the file system

This chart lists the specifiers you can use for the **FORMAT** option of the **BACKUP** command. You specify a pattern that RMAN uses to create backup file names in the style you request:

BACKUP Command FORMAT Option:	**Meaning:**
%a	Database activation ID
%c	Copy number of a backup piece within a set of duplexed backup pieces
%d	Database name
%D	Day of month
%e	Archived log sequence number
%f	Absolute file number
%F	A generated name (based on database ID or DBID, day, month, year, and sequence number)
%h	Archived redo log thread number
%I	Database ID or DBID
%M	Month
%N	Tablespace name
%n	Left-justified 8-character database name; Pads on the right
%p	Piece number within a backup set
%s	Backup set number
%t	Backup set timestamp
%T	Year, month, day
%u	8-character generated filename
%U	System-generated unique filename

We mentioned earlier that backup sets only (*not* image copies) can be written in compressed format. Use the **BACKUP AS COMPRESSED** command for this purpose. This example backs up an entire database as a compressed backup set:

```
RMAN> backup as compressed backupset database ;
```

Previous versions of RMAN reduced backup set size by backing up only used blocks and skipping unused blocks. 10g can now also physically compress the blocks that it backs up.

Incremental backups typically run faster than full backups because they include only changed data blocks. If you want to use incremental backups, you are first required to make a level 0 backup. A level 0 backup is a full backup, except that as a level 0 backup it functions as the required starting point or "parent" for subsequent incremental backups.

Here is one way to create the required level 0 backup for the entire database:

```
RMAN> backup incremental level 0 database ;
```

Now you can run a level 1 backup that picks up all changed data blocks since the level 0 backup:

```
RMAN> backup incremental level 1 database ;
```

Remember that, by default, we just created a differential incremental backup (see the "Terminology" section at the start of this Exam Manual). A cumulative incremental backup backs up all the data blocks that have been changed since the most recent backup of the next lower level. The cumulative incremental backup has the benefit that it makes for faster recovery, but the downside is that it typically runs longer than the differential incremental backup. Here's how to run a cumulative incremental backup:

```
RMAN> backup incremental level 1 cumulative database ;
```

Just specify the correct backup level and the keyword **DATABASE** to cover the entire database with the backup.

The table below shows a few other options on the **BACKUP** command to be aware of:

BACKUP command option:	**Usage:**
DURATION	The maximum time a backup runs before automatic termination.
FORMAT	Names the backup output files according to some given pattern.
MAXSETSIZE	Limits the size of a backup set.
MAXPIECESIZE	Limits the size of a backup piece.
RATE	Limits I/O bandwidth RMAN can use.
TAG	Names a backup set or image copy.

This example limits RMAN to reading only 3 M per second by specifying the **RATE**:

```
RMAN> configure channel device type disk rate 3 M ;
```

Once you have configured the channel with the **CONFIGURE** command, all subsequent backups that use that channel will use the configuration characteristics you have set for that channel.

This example limits the size of the output datafile to 5 G. *If the output needs to be larger than this maximum, the backup will fail:*

```
RMAN> backup database maxsetsize=5G ;
```

This next example limits backup pieces to 2 G. By default RMAN puts the entire backup set into one backup piece (datafile). This is a way around that limitation, for example, if your MML requires smaller files:

```
RMAN> configure channel device type disk maxpiecesize=2G ;
```

You can reduce the time it takes to produce backups by parallelizing the process with multiple channels. Achieve backup parallelism in either of 2 ways:

- Manually configure parallelism by assigning different channels to different parts of the backup
- Automatically parallelize the backup by defining RMAN's **PARALLELISM** configuration setting

This example manually configures parallelism. Each set of datafiles is specifically assigned its own channel:

```
RMAN> run {
        allocate channel ch1 type disk ;
        allocate channel ch2 type disk ;
        backup
            (datafile 1, 2, 3 channel ch1)
            (datafile 5, 6, 7 channel ch2) ;  }
```

Here's an example using automatic parallelism. First, set the **PARALLELISM** configuration parameter for RMAN through RMAN's **CONFIGURE** command. Parallelism will automatically kick in for the multiple items you list for backup:

```
RMAN> configure device type disk parallelism 3 ;
RMAN> backup (datafile 1, 2)
             (datafile 5, 6)
             (archivelog all) ;
```

When you specify multiple RMAN channels, RMAN also gives you automatic channel failover. If one channel fails, RMAN automatically finishes the backup using the other valid or working channels.

A new feature in 10g that speeds backups is called block change tracking. This feature keeps track of changed data blocks so that RMAN then only has to back up changed blocks (instead of scanning the database to find the changed blocks). The new block change tracking file keeps track of changed data blocks as changes are made to the database.

By default, block change tracking is disabled. To enable it, issue this command and specify a block change tracking file:

```
SQL>  alter database  enable block change tracking
                        using file  'c:\oracle\bct\bct_log.log'  ;
```

Enabling block change tracking starts the new background process **CTRW** to track changes. To disable block change tracking issue:

```
SQL>  alter  database  disable  block  change  tracking ;
```

Configuring RMAN

We've already shown how you can dynamically configure channels and other RMAN features through RMAN commands for a specific backup. An alternative approach is to permanently configure various RMAN behaviors so as to affect all subsequent backups.

The basic RMAN command for this purpose is **CONFIGURE**. To see the current configuration parameters, issue the RMAN **SHOW ALL** command:

```
RMAN>  show all ;
```

You'll see output similar to the following. *This output shows what RMAN parameters you can configure yourself, as well as what most of the initial defaults are*:

```
RMAN configuration parameters are:
CONFIGURE RETENTION POLICY TO REDUNDANCY 1; # default
CONFIGURE BACKUP OPTIMIZATION OFF;              # default
CONFIGURE DEFAULT DEVICE TYPE TO DISK;          # default
CONFIGURE CONTROLFILE AUTOBACKUP OFF:           # default
CONFIGURE CONTROLFILE AUTOBACKUP FORMAT FOR DEVICE
        TYPE DISK TO '%F';                                  # default
CONFIGURE DEVICE TYPE DISK PARALLELISM 1 BACKUP TYPE TO
        BACKUPSET;                                                  # default
CONFIGURE DATAFILE COPIES FOR DEVICE TYPE DISK TO 1; # default
CONFIGURE ARCHIVELOG BACKUP COPIES FOR DEVICE TYPE DISK
        TO 1;                                                               # default
CONFIGURE CHANNEL DEVICE TYPE DISK FORMAT
        '/oracle/flash_recovery_area/ora101c/%rec_area_%s_%p.bak' ;
CONFIGURE MAXSETSIZE TO UNLIMITED;                  # default
CONFIGURE ARCHIVELOG DELETION POLICY TO NONE; # default
CONFIGURE SNAPSHOT CONTROLFILE NAME TO
   'C:\ORACLE\PRODUCT\10.1.0\DB_1\DATABASE\SNCFORA101C.ORA';
                                                                    # default
```

The following are some examples of how to configure RMAN.

To configure for automatic channel allocation, issue various forms of the **CONFIGURE DEVICE**, **CONFIGURE DEFAULT DEVICE**, and **CONFIGURE CHANNEL** commands. Here are their formats:

```
CONFIGURE DEVICE TYPE DISK  backup | clear |  parallelism  n
CONFIGURE DEFAULT DEVICE TYPE  to  |  clear
CONFIGURE CHANNEL DEVICE TYPE  disk  |  equal
CONFIGURE CHANNEL n DEVICE TYPE disk  |  equal
```

Use the **CLEAR** keyword to un-set a previous device type configuration.

Here are some examples. To configure the default backup device to disk issue:

```
RMAN> configure default device type to disk ;
```

To configure the default backup device to tape use:

```
RMAN> configure default device type to sbt ;
```

SBT is the keyword RMAN uses for tape devices.

To set the default backup type to image copies, issue this command:

```
RMAN> configure device type disk backup type to copy ;
```

To re-set this value back to backup sets issue:

```
RMAN> configure device type disk backup type to backupset ;
```

To configure a default tape backup device to use a compressed backup:

```
RMAN> configure device type sbt backup type to compressed backupset;
```

To configure a default disk backup device to use a compressed backup:

```
RMAN> configure device type disk backup type to compressed backupset;
```

To configure tape parallelism to 5 use:

```
RMAN> configure device type sbt parallelism 5 ;
```

To set MML parameters, you typically issue a **CONFIGURE** command with a string parameter labeled **PARMS**:

```
RMAN> configure channel device type sbt PARMS='ENV=mml_settings' ;
```

The specific strings you can send to the MML depends on what that software expects and can process. RMAN just passes in the string (unaltered).

RMAN's retention policy dictates how long backups are retained for use. Configure the retention policy like this, using the **RECOVERY WINDOW** clause:

```
RMAN> configure retention policy to recovery window of 60 days ;
```

You can use the RMAN **CHANGE** command to cause a backupset to expire regardless of the current retention policy. This example expires the backupset called **monthly_backup** through the **NOKEEP** keyword:

```
RMAN> change backupset tag monthly_backup nokeep ;
```

This example keeps a backupset longer than the retention policy default specifies through the **KEEP UNTIL TIME** parameter:

```
RMAN>change backupset tag monthly_backup keep until time '31-DEC-06' logs;
```

Control file autobackup is a new feature in Oracle 10g that allows you to configure RMAN to automatically back up the control file whenever information impacting it changes. This is key because prior to 10g you (the DBA) were responsible for backing up the control file manually whenever database structural changes occurred. If you are like most of us, even though you knew to do this, there were occasions when you forgot to perform this backup. Now RMAN automates it and ensures it gets done as required.

This example enables control file autobackup:

```
RMAN> configure controlfile autobackup on ;
```

Note that the default behavior configures this useful feature to **OFF**.

This example configures control file autobackup and names the backup using the keyword **FORMAT**:

```
RMAN> configure controlfile autobackup format for device type disk to 'c:\oracle\bkups\cf_%F' ;
```

Monitoring RMAN

You can monitor and manage RMAN backups through the EM Database Control GUI. Or use RMAN commands like **SHOW**, **LIST**, and **REPORT**. Or inspect various dynamic performance views (also known as the **V$** tables).

To track backups you'll need to understand database incarnations, which are the different versions of a database. Specifically, a new incarnation of a database is generated whenever either:

- The database is opened with **RESETLOGS** (the **ALTER DATABASE OPEN** parameter that resets the redo log sequence back to a starting point)
- A database is opened with a **BACKUP CONTROLFILE** (a different control file than that expected and was previously in use)

To figure out a database's incarnation, follow these steps:

1. Get the database identifier (DBID) *from the target database*:

   ```
   SQL> select dbid from v$database ;
        DBID
        -------
        1835264768
   ```

2. Get that database's **DB_KEY** and therefore its current incarnation *from the recovery catalog database*:

   ```
   SQL> select db_key from rc_database where dbid = 1835264768 ;
   ```

3. If a database has multiple incarnations, run this query *on the recovery catalog database* to determine the current incarnation:

   ```
   SQL>  select  bs_key, backup_type, completion_time
              from  rc_database_incarnation  a, rc_backup_set  b
              where  a.db_key =  b.db_key
              and  a.current_incarnation =  'YES' ;
   ```

Use the RMAN **LIST** command to get information about backups and copies from the Recovery Catalog (also known as the RMAN Repository). Get a list with output segregated by physical database file categories:

```
RMAN> list backupset by file ;
```

Get the output by backup in summary form:

```
RMAN> list backupset by backup summary ;
```

Get the same information in more detailed form:

```
RMAN> list backupset by backup verbose ;
```

Use the RMAN **REPORT** command to find out which datafiles need to be backed up, which backups are no longer needed, what's the target database physical schema, and whether unrecoverable operations were performed on files. To ensure accuracy of the report, the RMAN repository catalog should first be synchronized with the target database's control files.

Issue this command to list obsolete backups and copies:

```
RMAN> report obsolete ;
```

Run this command to list tablespaces and their related datafiles, as well as their sizes:

```
RMAN> report schema ;
```

Finally, find out what needs to be backed up by:

```
RMAN> report need backup ;
```

The following table shows various dynamic performance views (V$ tables) and data dictionary views that you can look into for backup information:

View:	**Contains:**
V$BLOCK_CHANGE_TRACKING	Info on the block change tracking feature
V$RMAN_OUTPUT	RMAN error messages (also see the MML log and the log of the 3rd-party tape management software for further useful error messages)
V$RMAN_STATUS	Success/failure of completed RMAN jobs
V$PROCESS	Currently active processes
V$RECOVER_FILE	Datafiles that require recovery
RC_BACKUP_FILES	Backup info similar to **LIST**/**REPORT** commands
V$BACKUP_FILES	Backup info similar to **LIST**/**REPORT** commands

V$SESSION	Lists active sessions
V$SESSION_LONGOPS	Progress on long-running sessions (like RMAN backups and recoveries)
V$SESSION_WAIT	Wait events for sessions
V$BACKUP_ASYNC_IO	Shows rows when I/O is asynchronous to the backup process
V$BACKUP_SYNC_IO	Shows rows when I/O is synchronous to the backup process

RMAN Non-Critical Recoveries

Recovery scenarios can be divided into those that are non-critical and those that are critical. Non-critical situations allow failures to be addressed without database-wide impact and without affecting significant numbers of users. Kinds of non-critical events you'll want to know how to recover from include: loss of a temporary tablespace, loss of a redo log file, loss of an index tablespace, loss of an index, loss of a read-only tablespace, the loss of a password file, and, how to fix a datafile or tablespace that is stuck in "backup mode." Let's look at how to handle these situations first, then later we'll get into more critical recovery situations.

Tempfiles use local space management and support non-persistent data, such as the sort of data found in temporary tablespaces. For example, this statement creates a temporary tablespace using the **TEMPFILE** keyword:

```
SQL> create temporary tablespace my_temp tempfile
        'c:\oracle\oradata\prod\my_temp01.dbf' size 500 M
        extent management local uniform size 512 K ;
```

One way to address the loss of the temporary tablespace or its underlying datafiles is simply to create a new temporary tablespace (by a statement just like that seen above), then switch the database to use the new temporary tablespace as its default:

```
SQL> alter database default temporary tablespace my_temp ;
```

An Oracle database must always have a temporary tablespace available to operate and it must be a tablespace of type **TEMPORARY** (not **PERMANENT**).

Here's a variation on the same scenario. Say you try to start an Oracle database and you get an error message saying a tempfile is missing. You could:

1. Start the database in **MOUNT** mode:

   ```
   SQL> startup mount ;
   ```

2. Drop the tablespace with the problem:

   ```
   SQL> drop tablespace my_temp including contents and datafiles ;
   ```

3. Re-create the temporary tablespace:

   ```
   SQL> create temporary tablespace my_temp tempfile
   'c:\oracle\oradata\prod\my_temp01.dbf' size 500 M
   extent management local uniform size 512 K ;
   ```

Online redo log files track changes to the database in the form of redo entries or redo records. Background process Log Writer writes redo to all the files (members) of an online redo log group at one time. Online redo files are thus multiplexed (mirrored) for high reliability and availability. The database stays up as long as one single member is available for writing in any online redo log group. Losing an online redo log group member is therefore a non-critical recovery scenario. Here's one way to handle it:

1. Identify the missing redo log file member from the Alert log and the Oracle trace files. For this scenario, pretend that the log file member with the problem is named **redo01.log**.
2. As long as **redo01.log** is not the currently active log, you can remove it from the data dictionary:

   SQL> alter database drop logfile member 'c:\oracle\oradata\prod\redo01.log' ;

   If there is the possibility that Log Writer would attempt to make this log file active during your work, you may want to **STARTUP RESTRICT** the entire database to prevent outside transactions from making this happen.
3. Re-create the missing log file member:

   SQL> alter database add logfile member 'c:\oracle\oradata\prod\redo01.log' to group 1 ;

Indexes are data structures maintained by Oracle that support faster access to data and can also help avoid expensive sorting operations. Indexes are typically stored in their own index tablespace. If you lose an index, you normally just drop and re-create it. Take the same approach if you lose an index tablespace:

1. SQL> drop tablespace my_index including contents ;
2. SQL> create tablespace my_index datafile 'c:\oracle\oradata\prod\my_index01.dbf' size 100 M ;

If you have referential integrity constraints involved you may want to research this first, so that you create indexes in proper order if multiple indexes are involved.

Read-only tablespaces only permit read operations (no inserts or updates). You create a tablespace as read-only, or you can flip an existing tablespace into read-only mode through the **ALTER TABLESPACE . . . READ ONLY** statement. Once you have created or altered a tablespace to read only, you just need to back up the tablespace one time to have a usable backup of it. Since the System Change Number (SCN) in the datafile headers of the read-only tablespace never changes, you do not need to back it up again -- as long as it remains a read-only tablespace.

To recover a read-only tablespace, just copy the backup into the proper location in the database. For example, if you have an image copy of the damaged read-only tablespace file, first either take the tablespace offline, or shutdown the database. Then just use an operating system **copy** or **cp** command to bring back the required file. Now bring it online, and you're done.

You do not need to **RECOVER** (apply redo logs) to the read-only tablespace, because there have been no changes to it since it was backed up.

Password files are required for all remote connections for database administration using **SYSOPER** or **SYSDBA** privileges. To use them, set the initialization parameter **REMOTE_LOGIN_PASSWORDFILE** to **SHARED** or **EXCLUSIVE**.

If you lose a password file, just copy in its backup. Another approach is to create a new password file using the Oracle password utility named **ORAPWD**:

1. Shutdown and restart the database
2. Run the **ORAPWD** utility. This creates the password file named **orapw$ORACLE_SID** and puts it in directory **$ORACLE_HOME/dbs** for Unix systems and in directory **%ORACLE_HOME%\database** on Windows systems:

   os> orapwd file=orapwPROD password=my_pass entries= 40
3. Check the view **V$PWFILE_USERS** to verify the results:

   SQL> select * from v$pwfile_users ;

Backup mode is used for datafiles or tablespaces only when you're using user-managed backups. You place a datafile or tablespace into backup mode in order to create a backup of it. Then you take it out of backup mode when the backup copy has been completed.

On occasion you may encounter the problem where, for whatever reason, the datafile or tablespace was not taken out of backup mode after the copy. You can fix this problem without recovery by these steps:

1. Identify the datafiles in backup mode by querying view **V$BACKUP**
2. Take them out of backup mode by issuing the command

   SQL> alter datafile 'datafile_name' end backup ;

Oracle 10g also offers a new command that ends backup mode for all tablespaces or datafiles still in backup mode through a single command:

```
SQL> alter database end backup ;
```

RMAN Recoveries

The previous section addressed special situations where non-critical recovery is possible. Now let's address more common recovery scenarios.

Database Recovery to Currency -- Here's an example of recovering an entire database to currency (the point after the last committed transaction). To do this you would shut down the database and put it into **MOUNT** state, then run the RMAN **RESTORE** and **RECOVER** commands. Remember that the **RECOVER** command applies all outstanding (committed) transactions from the logs to the database:

1. SQL> startup mount ;
2. RMAN> run {
   allocate channel ch1 type disk ;
   restore database ;
   recover database ;
   alter database open ; }

Incomplete Database Recovery -- Here's an example where you recover the entire database to a specified point in time. This is called an incomplete recovery because you are not recovering the database to currency.

You can specify the point to which to recover by these RMAN keywords on the **RECOVER** statement:

- **UNTIL TIME** - Recovers to the specified time
- **UNTIL SEQUENCE** - Recovers to a redo log sequence number
- **UNTIL SCN** - Recovers to a System Change Number (SCN)

After shutting down the database, you would:

1. Set the environmental variable **NLS_DATE_FORMAT** to something usable:
   c:> set NLS_DATE_FORMAT='DD-MON-YYYY HH24:MI:SS'
2. Put the database into the **MOUNT** state:
   SQL> startup mount
3. **RESTORE** and **RECOVER** the database to the specified point in time:
   ```
   RMAN> run {
           set until time '15-OCT-2005' 13:05:00' ;
           restore database ;
           recover database ; }
   ```
4. Be sure to specify **RESETLOGS** when opening the database. This restarts the log sequence number and *is required to open a database after incomplete recovery*:
   SQL> alter database open resetlogs ;

Losing Control Files -- You can configure RMAN for control file autobackup by:

```
RMAN> configure controlfile autobackup on ;
```

If you lose one of the control files, use the multiplexed (mirrored) copies of these files to restore it. (Remember that an instance requires *all* of its control files be present and usable for it to run). To restore using one of the multiplexed copies, **SHUTDOWN ABORT** the instance. Copy one of valid control files to the location of the missing or corrupted one. Then **STARTUP** the instance. (The instance's automatic instance recovery (or *crash recovery*) will handle any issues created by your **SHUTDOWN ABORT** automatically upon instance startup.)

If you want to restore a controlfile from a control file autobackup, perform these steps:

1. Shutdown the database:
   SQL> shutdown abort
2. Use the NOMOUNT state when restoring a control file:
   SQL> startup nomount
3. Connect to the target database from within RMAN:
   RMAN> connect target /
4. Specify the database ID (DBID) of the database you are connecting to. You need to specify this manually because the control file that normally provides this information is unavailable:
   RMAN> set dbid 1837615283 ;
5. Restore the control file:
   RMAN> restore controlfile from autobackup ;

6. To recover the rest of the database, go to the MOUNT state:
   RMAN> alter database mount ;
7. Apply the redo logs:
   RMAN> recover database ;
8. Open the database, using the RESETLOGS parameter:
   RMAN> alter database open resetlogs ;

You should also know how to re-create a control file "from scratch," that is, using a textual file you backed up earlier. Recall that you back up the control file to an editable text file through the **BACKUP ...TO TRACE** keywords:

```
SQL> alter database backup controlfile to trace ;
```

The textual control file backup will reside in the **UDUMP** directory. It has all the commands in it you need to create a control file. These commands always start with the **STARTUP NOMOUNT** command, because you must have the database in this state to build a new control file.

You'll also notice that the textual control file commands come in two complete scenarios: one for complete recovery (ending with **ALTER DATABASE OPEN**), and the other for incomplete recovery (ending with the mandatory **ALTER DATABASE OPEN RESETLOGS** command). Before building the new control file, edit the text to eliminate (delete or comment out) the scenario you do not want to use. For example, if you'll do a complete recovery, delete the incomplete recovery scenario text from the file.

Since all the required commands to re-build a control file are in the text file, to build the new control file just connect with proper privileges to the target database, then run the control file script you just edited:

1. SQL> connect / as sysdba
2. SQL> @control_file_script.txt

The at sign (@) runs the script from within SQL*Plus.

Datafile Recoveries -- In the Exam Manual for the OCA Exam (Exam #1Z0-042 Oracle 10g Administration I), the following topics were discussed:

- Recover non-critical lost datafile(s) with an open database in Archivelog mode.
- Recover critical lost datafiles(s) (those in the **SYSTEM** and **UNDO** tablespaces) for Archivelog mode databases. This requires shutting down the entire database prior to restore/recovery.
- Recover either non-critical or critical datafile(s) for Noarchivelog mode databases. Recoveries for Noarchivelog mode databases always require shutting down the entire database prior to recovery.

Note: These are important, basic recovery scenarios. Make sure you understand them. We do not repeat the material here to save space. Download the "Oracle 10g Administration I Exam Manual" if you are not familiar with these scenarios.

Using User-Managed Recovery Instead of RMAN

User-managed recovery means that you manually issue the necessary SQL and operating system commands to recover a database, rather than relying on RMAN to do it. Obviously, RMAN is the recommended approach, but sometimes you'll see user-managed recovery at sites that established their procedures before RMAN matured, or at sites with special or unusual backup and recovery requirements.

Here's an example scenario for user-managed recovery. Say you **STARTUP** a database and get an error message like this in response:

```
ORA-01157: cannot identify/lock data file 7 - see DBWR trace file
ORA-01110: data file 7 - 'c:\oracle\oradata\prod\data07.dbf'
```

So data file #7, **data07.dbf**, is damaged or has a problem. To fix this, first shutdown the database. Then restore the invalid file from a good backup. Since you're using user-managed recovery rather than RMAN, instead of the RMAN **RESTORE** command; you would use an operating system **copy** or **cp** command to bring back the backup file to its valid location within the database.

Then, you would **STARTUP MOUNT** the database, perform a recovery by applying the redo logs, and open the database for use.

1. SQL> startup mount
2. SQL> recover database ;
3. SQL> alter database open ;

The **RECOVER DATABASE** command will prompt you for the logs to apply. Normally, just press the **<ENTER>** key in response to each log file that the database recommends you apply or type AUTO to let Oracle do the work.

How about incomplete recovery with user-managed techniques? In this case you would recover the database to a specific time, SCN, or until you cancel the application of the redo logs. Here's one example:

1. Set the environmental variable **NLS_DATE_FORMAT** to something usable:

   c:> set NLS_DATE_FORMAT='DD-MON-YYYY HH24:MI:SS'
2. After shutting down the database, copy in the damaged file(s) from backups using operating system commands.
3. Put the database into **MOUNT** state:

   SQL> startup mount
4. Recover the database to the point in time desired:

   SQL> recover database until time '15-OCT-2005' 13:05:00' ;
5. Open the database for use. As in all cases where you have performed an incomplete recovery, you must use the **RESETLOGS** parameter when opening the database. This resets or restarts the log sequence numbers:

   SQL> alter database open resetlogs ;

Cancel-based recovery allows you to apply individual redo logs during the recovery step, stopping whenever you like. To perform cancel-based recovery, simply replace the **RECOVER** command above with this command:

```
SQL> recover database until cancel ;
```

You'll be prompted whether to apply each individual log, or to **CANCEL** the process.

One of the very few advantages of user-managed recovery over RMAN is that it supports the **UNTIL CANCEL** option; RMAN does not. The following table shows a full list of user-managed versus RMAN keywords for incomplete recovery using the **RECOVER** command.

Incomplete recovery options:

User-managed options:	RMAN options:	Meaning:
UNTIL TIME	UNTIL TIME	Apply logs until the specified time
UNTIL CHANGE	UNTIL SCN	Apply logs until the specified SCN
UNTIL CANCEL	UNTIL SEQUENCE	**User-managed:** apply logs until the user manually cancels. **RMAN:** apply logs until a specified log sequence.

Using Enterprise Manager Instead of RMAN

The EM panels support the same functionality you get by issuing RMAN line commands. To access the Enterprise Manager GUI panels, enter the appropriate URL (web address) into your browser. This should have the format:

http://hostname:5500/em
where **hostname** is your EM server's hostname,
5500 is the default EM port,
and **em** is a constant.

Here's an example of entering the URL:

http://myhost:5500/em

Login using the user id **SYS** and login with **SYSDBA** rights. You typically would always log in this way to have appropriate privileges for backup and recovery activity.

Once you have access to the main EM **Database Control panel**, select the **Maintenance** link. On this panel you'll see the heading **Backup/Recovery** with a half dozen options to perform, schedule, configure, and manage backups and recoveries. The options on those individual action panels then correspond to (and generate) appropriate RMAN scripts to fulfill those actions. This lists the options as they occur under the **Backup/Recovery** heading on the **Maintenance** panel:

Backup/Recovery
- Schedule Backup
- Perform Recovery
- Manage Current Backups
- Configure Backup Settings
- Configure Recovery Settings
- Configure Recovery Catalog Settings

From these options panels, EM will automatically generate the proper RMAN and SQL commands to perform the actions you want. Press the **SHOW SQL** button to see the RMAN and SQL commands EM generates on your behalf.

Block Corruption and Recovery

Data blocks can become corrupt due to various kinds of hardware or software bugs. You normally identify or detect block corruption through:

- Windows' **Event Viewer** messages or its Unix equivalent, the system log called **Syslog** that typically resides in **/var/adm/syslog**.
- The Oracle **Alert Log** messages.
- Oracle background process or user process **trace files** in the **BDUMP** or **UDUMP** directories (respectively).
- Running the **ANALYZE TABLE table_name VALIDATE STRUCTURE** command, such as this:

  SQL> analyze table my_schema.my_table validate structure ;
- Running the **DBVERIFY** utility against offline datafiles.
- Turning on initialization parameter **DB_BLOCK_CHECKING**, which verifies data and index blocks every time they are created or modified through a checksumming algorithm. By default, this parameter is **FALSE** for all tablespaces except for the **SYSTEM** tablespace. Turning this parameter on results in system overhead but also is a useful tool if you experience block corruption and have trouble determining why or when it is happening.
- Using the **DBMS_REPAIR** package on tables, indexes, or partitions.

DBVERIFY – Run this utility against datafiles or backups of datafiles. You can use it while the database is online or offline, but the most consistent results are from offline scans. This example runs **DBVERIFY** from the command line using its name, **dbv**:

```
c:> dbv blocksize=2048 file=c:\oracle\oradata\prod\tools01.dbf
```

The utility output shows the number of data blocks processed without error, the number of blocks processed, the number of empty blocks, and the number of blocks marked as corrupt.

Here are the **DBVERIFY** parameters:

DBVERIFY Parameter:	Usage:
BLOCKSIZE	Database blocksize as per init. parm **DB_BLOCK_SIZE**, default **8192**.
END	Ending block when verifying a range of blocks
FILE	Name of the file to verify
FEEDBACK	Displays utility progress
HELP	Code help=y for help on **DBVERIFY** parameters

LOGFILE	Name of the log file for logging the results
PARFILE	Parameter file that contains options
SEGMENT_ID	Segment identifier
START	Starting block when verifying a range of blocks
USERID	User name and password

The DBMS_REPAIR Package – This package helps identify and resolve block corruption in tables and indexes. It includes the following procedures:

DBMS_REPAIR Procedure:	**Usage:**
CHECK_OBJECT	Identifies corrupt blocks in a table or index.
FIX_CORRUPT_BLOCKS	Marks blocks as software-corrupt. You should first process those blocks with **CHECK_OBJECT**.
DUMP_ORPHAN_KEYS	Reports index key entries that point to rows in corrupt blocks.
REBULD_FREELISTS	Rebuilds object freelists.
SEGMENT_FIX_STATUS	For AUTO segment space management, this can fix corrupted bitmap entries.
SKIP_CORRUPT_BLOCKS	Ignores blocks marked as corrupt during table and index scans, thereby avoiding error ORA-1578.
ADMIN_TABLES	Administrative functions for orphan key tables (these contain keys that point to rows in corrupt blocks). These tables always reside in the **SYS** schema.

Identifying and repairing corrupt blocks by the **DBMS_REPAIR** package procedures is a complicated process that may result in loss of data in corrupt blocks. For the exam, you'll need to be familiar with the **DBMS_REPAIR** procedures. Outside of the exam, be aware that most sites use other means of repairing corrupt blocks, such as:

- Rebuilding the object (for example, dropping and recreating an index)
- Datafile recovery (which is easy for non-critical files, especially when using RMAN), and can be performed while the rest of the database remains up for Archivelog mode databases
- Block media recovery (described next below)

Block Media Recovery – Block Media Recovery, or BMR, is an RMAN feature that allows you to recover specific data block(s). Its advantage over traditional datafile recovery is that it is faster than a datafile restore/recover process, and also that you do not have to take a datafile offline while you recover its bad block(s). Its limitations are:

- BMR is an RMAN feature and must be used with RMAN.
- You must have a full RMAN backup available (not an incremental backup).
- You must perform complete recovery of data blocks (no incomplete recoveries).
- You cannot recover blocks marked as corrupt in the media backup source. These are termed media corrupt blocks.

How do you know which data block(s) need recovery? You'll see an error message similar to this one in the Alert Log or in a trace file:

```
ORA-01578: ORACLE data block corrupted (file # 6, block # 155)
ORA-01110: data file 6: 'c:\oracle\oradata\prod\data06.dbf'
```

In this example, block # 155 in file # 6 has the corruption problem.

To restore and recover it, go into RMAN and connect to the target database. Then issue:

```
RMAN>  blockrecover  datafile 6  block 155 ;
```

Flashback Technologies

The Flashback Recovery Tools

Oracle 10g Flashback is a set of features designed to enable recovery from logical errors. By logical errors, we mean human error such as running a batch update program twice by mistake or accidentally dropping a table or running an incorrect query. Oracle correctly and loyally produces the result you asked for – the trouble is that you accidentally asked for a result you did not intend. Flashback can sometimes be used for other kinds of recovery, but addressing user or logical error is its primary purpose.

While there are a variety of flashback features, all use the shared flashback recovery area (or flash recovery area). This area can be a filesystem, directory, or Automatic Storage Management (ASM) disk group. It can be shared by more than one Oracle database. Recovery from the flashback recovery area is fast because it is always disk storage.

The flash recovery area backs up all data for flashback recoveries including:

Flashback Area Resident:	**From:**
Flashback logs	Used by Flashback Database to take an entire database back to an earlier point in time. Flashback logs are written by new background process **RVWR** and contain "before images" of blocks.
Control Files	A control file copy is placed in the flashback area when the database is created.
Archived Redo Logs	Initialization parameter **LOG_ARCHIVE_DEST_10** automatically points to the flashback area so that the background processes ARCn automatically copy archived redo logs to the flashback area.
SPFILE backups	Binary server parameter files (as generated by RMAN).
RMAN backup sets	As generated by RMAN for regular datafile backups.
RMAN image copies	As generated by RMAN when using RMAN's **BACKUP AS COPY** command.

The flashback recovery area does not include the password file or Oracle binaries.

To set up the flash recovery area, you must set two dynamic initialization parameters:

- **DB_RECOVERY_FILE_DEST_SIZE** - Size of the area
- **DB_RECOVERY_FILE_DEST** - Where it's located

*You **must** specify the first parameter prior to the second.* If using Real Application Clusters (RAC), all instances in the cluster must have the same two values for these parameters. Oracle automatically uses the Oracle feature called Oracle Managed Files (OMF) for file naming in the flash recovery area. (OMF automatically names files according to standard conventions and also automates aspects of file management).

Here's an example of how to configure the flash recovery area. Assume we are using a server parameter file (**SPFILE**) to parameterize the Oracle database:

1. SQL> alter system set db_recovery_file_dest_size=50M scope=both;
2. SQL> alter system set db_recovery_file_dest = 'c:\oracle\flash_area' ;

You can manage the flashback recovery area through the EM Database Control GUI Maintenance link. Oracle automatically manages the file space in the flash recovery area. It deletes obsolete files and gives a Warning Message at 85% filled and a Critical Message at 97% filled. These appear in EM Database Control and the Alert Log.

To back up the files in the flash recovery area, issue the RMAN **BACKUP... RECOVERY FILES** command. This example backs up its files to tape:

```
RMAN> backup device type sbt recovery files ;
```

This command backs up the entire flash recovery area:

```
RMAN> backup recovery area ;
```

The view **V$RECOVERY_FILE_DEST** provides information about the flash recovery area. This includes its location, space allocation, what files are there, etc. The new **BYTES** column of view **V$BACKUP_PIECE** also indicates how big a file is in the flash recovery area. The new column, **IS_RECOVERY_FILE_DEST**, indicates whether a file exists in the flash recovery area. This new column appears in views:

- **V$BACKUP_PIECE**
- **V$DATAFILE_COPY**
- **V$CONTROLFILE**
- **V$LOGFILE**
- **V$ARCHIVED_LOG**

The 5 kinds of flashback recovery technologies and their purposes are:

Flashback Feature:	**Relies primarily on data from:**	**Purpose:**
Flashback Versions Query	Undo Tablespace	Retrieves all versions of table rows between two SCNs or Timestamps.
Flashback Transaction Query	Undo Tablespace	Retrieves all table rows affected by a specific Transaction.
Flashback Table	Undo Tablespace	Recovers one or more tables to a specific point in time -- without using traditional point-in-time recovery.
Flashback Drop	Recycle Bin	Recovers a dropped table -- without using traditional point-in-time recovery.
Flashback Database	Flashback Logs	Recovers the entire database to a prior point-in-time.

Prior to Oracle 10g, Oracle supported Flashback Query. This has now been enhanced into two different varieties:

- Flashback Versions Query
- Flashback Transaction Query

Flashback Versions Query

Flashback Versions Query displays all rows in a table between two System Change Numbers (SCNs) or Timestamps. This shows the rows regardless of whether they were inserted, updated, or deleted. Flashback Versions Query does not work for:

- Temporary tables
- External tables
- Fixed tables
- Views
- Ranges during which there were structural changes in the table (back before any period during which you changed the table structure by DDL)

This is the **SELECT** statement format you use for Flashback Versions Query:

```
SELECT [pseudo_columns]... FROM table_name
VERSIONS BETWEEN
{ SCN | TIMESTAMP {expression | MINVALUE} AND
                  {expression | MAXVALUE} }
[ AS OF {SCN | TIMESTAMP expression} ]
          WHERE ...
```

The **MINVALUE** keyword represents the oldest SCN or timestamp, while the **MAXVALUE** keyword represents the most recent. The possible **pseudo_columns** in the above statement can be:

Pseudo Column:	**Meaning:**
VERSIONS_STARTSCN	SCN when this version of row was created
VERSIONS_STARTTIME	Timestamp when this version of row was created
VERSIONS_ENDSCN	SCN when this version of row was changed or deleted
VERSIONS_ENDTIME	Timestamp when this version of row was changed/deleted
VERSIONS_XID	Transaction ID that created this version of the row
VERSIONS_OPERATION	Type of DML (I = Insert, U = Update, D = Delete)

When working with SCNs and timestamps, use the two conversion functions **SCN_TO_TIMESTAMP** and **TIMESTAMP_TO_SCN** to convert as needed.

Here is an example of a flashback versions query:

```
SQL> select  purchase_order_id  from  po_table
          versions  between  timestamp
          to_timestamp('2005-10-15  10:30:00','YYYY-MM-DD HH:MI:SS') and
          to_timestamp('2005-10-15  10:35:00','YYYY-MM-DD HH:MI:SS') ;
```

Like Flashback Transaction Query and Flashback Table, Flashback Versions Query relies on the Undo tablespace to retrieve "before change images" of older data. Set initialization parameter **UNDO_RETENTION** to a value (in seconds) to tell how long to keep undo data in the database. This does not, however, guarantee undo for that period. Only setting the new clause **RETENTION GUARANTEE** causes Oracle to guarantee that undo will always be available for the specified retention period. Specify the **RETENTION GUARANTEE** clause when creating the Undo Tablespace. The default value for **RETENTION** is **NO GUARANTEE**. View the current retention value in the view **DBA_TABLESPACES**. Or use EM.

The privileges required for Flashback Versions Query are **SELECT** and **FLASHBACK** for each user.

Flashback Transaction Query

Flashback Transaction Query retrieves all table rows affected by a specific transaction. To use it, simply retrieve information from the dictionary view **FLASHBACK_TRANSACTION_QUERY**. The columns in this view are:

- XID - Transaction ID
- START_SCN - SCN for 1st transaction DML
- START_TIMESTAMP - Timestamp for 1st transaction DML
- COMMIT_SCN - SCN when transaction was committed
- COMMIT_TIMESTAMP - Timestamp when transaction was committed
- LOGON_USER - User for the transaction
- UNDO_CHANGE# - The undo SCN
- OPERATION - Type of DML (Insert, Update, Delete, Unknown)
- TABLE_NAME - Name of table involved
- TABLE_OWNER - Owner of that table
- ROW_ID - The row modified
- UNDO_SQL - The SQL statement to undo the DML operation

You need the **SELECT ANY TRANSACTION** privilege to use this facility.
Whereas you recognize Flashback Versions Query by a **SELECT** statement with the **VERSIONS BETWEEN** clause, you recognize Flashback Transaction Query simply by the fact the query goes against the table **FLASHBACK_TRANSACTION_QUERY**. Here is an example of Flashback Transaction Query that retrieves a particular transaction:

```
SQL> select  table_name,  operation,  undo_sql
                from  flashback_transaction_query
                where  xid = '392635128F040000' ;
```

Flashback Table

Flashback Table allows you to recover one or more tables quickly to a point in time (without using traditional recovery methods). To use it, you **must** first enable row movement on the table(s) you want to flash back:

```
SQL> ALTER TABLE table_name ENABLE ROW MOVEMENT ;
```

Then flashback the table. This example goes back 10 minutes:

```
SQL> FLASHBACK TABLE table_name TO TIMESTAMP
       systimestamp - INTERVAL '10' MINUTE ;
```

You can re-run this query again using a different interval if you need to. The command acquires exclusive DML locks on the table. You cannot use this feature if a structural change has occurred to the table within the time frame you work with.

As well as by the **TO TIMESTAMP** clause, you can specify the **TO SCN** clause. Here's another example that flashes back table data to a prior SCN:

```
SQL> FLASHBACK TABLE table_name TO SCN 883651 ;
```

Triggers are disabled by default during the flashback table process. To enable them, use the **ENABLE TRIGGERS** keywords:

```
SQL> FLASHBACK TABLE table_name TO SCN 883651
       ENABLE TRIGGERS ;
```

You need the system privilege **FLASHBACK TABLE** or **FLASHBACK ANY TABLE** to use this feature, as well as proper DML privileges (**SELECT**, **INSERT**, **DELETE**) and **ALTER** object.

Flashback Drop

Flashback Drop recovers a dropped table to some previous point in time before the drop. It can be much faster to use than traditional recovery methods. Flashback Drop and its recycle bin are enabled by default in Oracle 10g.

Flashback Drop uses the recycle bin, a logical structure based on a dictionary table. You can find out what's in the recycle bin by querying view **DBA_RECYCLEBIN** or **USER_RECYCLEBIN.** You can also issue the **SHOW RECYCLEBIN** command from within SQL*Plus. You may also want to query **DBA_CONSTRAINTS** if working with more than one table, to see if there were logical relationships (dependencies) between them, before bringing back any dropped table(s).

Only tables defined within locally managed tablespaces are placed into the recycle bin when dropped. Dependent objects for eligible tables are also preserved and recovered. These include indexes, triggers, non-referential constraints, nested tables, Large Binary Object (LOB) segments, and LOB index segments. Related objects that are **not** saved include referential constraints, bitmap join indexes, materialized view logs, partitioned index-organized tables, Virtual Private Database (VPD) policies, and Fine-Grained Auditing (FGA) policies.

Limitations for Flashback Drop include:

- Only works with non-system, locally-managed tablespaces
- You cannot issue DML or DDL statements on recycle bin objects
- Use the recycle bin table name to query a table (not its original name)
- Flashback Drop automatically retrieves the dependent objects specified above
- There is no guarantee for how long an object is stored in the recycle bin

To recover a table and its dependent objects, use the **FLASHBACK TABLE... TO BEFORE DROP** command:

```
SQL> FLASHBACK TABLE table_name TO BEFORE DROP ;
```

You might also want to rename the recovered table in the process:

```
SQL> FLASHBACK TABLE table_name TO BEFORE DROP
       RENAME TO my_recovered_table ;
```

A big concern with flashback drop is: how far back can you go? In part, this depends on the size of the recycle bin and how it reuses space. Objects in the recycle bin are deleted on a First-In-First-Out (FIFO) basis. Oracle gets free space required to place a newly-dropped object into the recycle bin in this order:

1. Unoccupied free space in the recycle bin
2. Free space taken from dropped objects
3. Autoextension of the recycle bin tablespace

Note that dropped objects are purged (2) prior to autoextending the tablespace (3).

You can manually free space from the recycle bin through the **PURGE TABLE** and **PURGE INDEX** commands. Or use **PURGE TABLESPACE** to get rid of a set of objects.

The **PURGE RECYCLEBIN** (aka **PURGE USER_RECYCLEBIN**) command purges all objects a user owns from the recycle bin, while **PURGE DBA_RECYCLEBIN** purges all objects. The latter requires the **DBA** or **SYSDBA** privilege.

DBA_FREE_SPACE and **DBA_RECYCLEBIN** views are where to look to see what's in the recycle bin and their sizes.

If you want to drop a table without putting it in the recycle bin (basically bypassing the recycle bin), you'll want to use the new PURGE keyword:

```
SQL> DROP TABLE table_name PURGE ;
```

This permanently drops the table and its data. Flashback Drop will not be a available.

To get to the Flashback Drop functionality through the Enterprise Manager (EM), go to the **Maintenance** => **Recovery** => **Dropped Objects** screen.

Flashback Database

Flashback Database recovers an entire database to a prior point in time by a means that is much faster than traditional recovery. Flashback database uses the new background process RVWR (Recovery Writer) to write data from the flashback buffer in the System Global Area (SGA) to the flashback logs in the flash recovery area. These flashback logs are then used when you recover the entire database using flashback database. Here are the steps to set up flashback database;\

1. **STARTUP MOUNT EXCLUSIVE** the database
2. **ALTER SYSTEM SET DB_FLASHBACK_RETENTION_TARGET = n**; where n is specified in minutes.
3. **ALTER DATABASE FLASHBACK ON ;**
4. **ALTER DATABASE OPEN ;**

You must be in **MOUNT** state with an Archivelog mode database to enable flashback database. Set the parameter **DB_FLASHBACK_RETENTION_TARGET** to the number of minutes for eligible flashback. Changes to the database will now be written both to the redo logs *and* the flashback logs in the flashback recovery area.

Check whether flashback database is enabled via:

```
SQL> SELECT FLASHBACK_ON FROM V$DATABASE ;
```

View **V$FLASHBACK_DATABASE_LOG** helps monitor the estimated size of the flashback logs, while **V$FLASHBACK_DATABASE_STAT** shows how much flashback log is being written per time period. **V$DATABASE** tells whether flashback database is enabled or not.

You can exclude a specific tablespace from a flashback-enabled database by:

```
SQL> ALTER TABLESPACE my_ts FLASHBACK OFF;
```

Column **FLASHBACK_ON** in **V$TABLESPACE** tells whether flashback is enabled for each tablespace.

Figure 1: Recovering a Database with Flashback Database

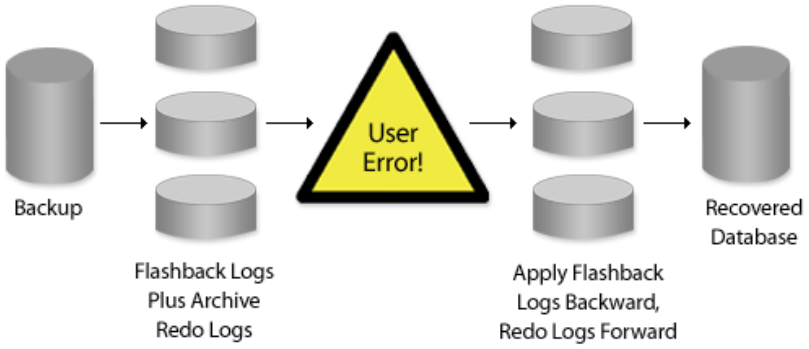

Figure 1 diagrams how recovering a database by flashback database works.

Steps:

1. **SHUTDOWN IMMEDIATE;**
2. **STARTUP MOUNT EXCLUSIVE ;**
3. **FLASHBACK DATABASE TO TIMESTAMP**(some_time) ;
4. **ALTER DATABASE OPEN RESETLOGS;**

You **must** use the Mount state and open the database with **RESETLOGS**. You can flashback to either a timestamp or an SCN.

Flashback database will not recover tables dropped after the target recovery time. You cannot use flashback database if the control file was restored or re-created after the target recovery time or if the database was opened with **RESETLOGS** after the target recovery time.

To access Flashback Database from the Enterprise Manager GUI screens, access EM choices **Database Control** => **Maintenance** => **Backup/Recovery** => **Perform Recovery**. You must login with **SYSDBA** rights to perform flashback database recovery.

Automatic Storage Management (ASM)

New in Oracle 10g, Automatic Storage Management, or ASM, offers an entirely new way to manage Oracle database storage. ASM is highly automated; it provides features like *striping* and *mirroring*. It supports all kinds of filesystems, and it often eliminates the need to purchase a third-party volume management tool or clustered filesystem support tool. ASM manages data files and provides an alternative to raw devices or filesystems that are locally attached to a database instance. It can support clustered systems, where more than one instance runs against the same database. Oracle calls their database clustering software Real Application Clusters (RAC). ASM supports RAC.

ASM requires an entire new instance, the ASM instance, to control datafiles and storage. One or more other database instances called clients use the ASM instance for their storage needs. If you use the Oracle Universal Installer (OUI) and the Database Configuration Assistant (DBCA) to create a database, it provides an option to automatically create the ASM instance for you.

The ASM instance does *not* have a data dictionary. All connections into it must be via operating system authentication. So you connect into it with the **SYSDBA** or **SYSOPER** login ids identified by operating system userids in the **dba** group. The ASM instance does *not* have a control file, so you do not supply an initialization parameter that specifies a list of control files to an ASM instance.

Start the ASM instance first, then the client instances next. Shutdown the ASM instance first, which then propagates the shutdown to its client instances. For example, issue a **SHUTDOWN NORMAL** to the ASM instance, and it propagates this command to its clients. The ASM instance waits for its clients to shutdown before completing.

The exception to this behavior is when you issue **SHUTDOWN ABORT** to the ASM instance. Issuing this command to the ASM instance causes ASM to abort without passing the command to its clients. (But they'll abort anyway, because now their storage is missing).

In a RAC environment, you might have multiple ASM instances sharing their disks. In this situation if one ASM instance fails, another ASM instance automatically performs a recovery operation for the failed instance.

When you start an ASM instance, you bring it into its typical operating state through **STARTUP MOUNT**. This allows client database(s) to connect to its disks. ASM instances are never taken into the **OPEN** state; they run in the **MOUNT** state.

You can issue **STARTUP RESTRICT** to the ASM instance to start it all the way up without allowing any client instances to connect to it. This is useful for maintenance.

For more detail, the following table shows the ASM startup options:

STARTUP option:	**Usage:**
STARTUP NOMOUNT	Starts the ASM instance but does not mount disks.
STARTUP MOUNT	Starts the ASM instance and mounts the disks. This state is for normal use in which client databases connect to the ASM instance and use the ASM-controlled storage.
STARTUP	Same as **STARTUP MOUNT** (**STARTUP** defaults to **STARTUP MOUNT**).
STARTUP RESTRICT	Starts the ASM instance but does not allow client databases to connect to it. Used for maintenance.

Flip a started ASM instance into and out of restricted state with these commands:

```
SQL> alter system enable restricted session ;
SQL> alter system disable restricted session ;
```

ASM Storage

ASM uses the concept of a disk group, a set of disk devices it manages as a logical unit. (Disk groups are analogous to the "volume groups" used by many operating systems to group and manage disks.) The most basic unit of storage in ASM is the extent. Extents are the units of data mirroring.

ASM automatically rebalances data across a disk group when space is added to or removed from a disk group. The ASM background process **RBAL** coordinates the rebalancing, while the processes **ORBn** (**n** is from 0 to 9) actual carry out the rebalancing. Rebalancing occurs transparently to users, as a background operation.

The ASM instance initialization parameter **ASM_POWER_LIMIT** dictates how fast background rebalancing occurs and how much resource it consumes. Set **ASM_POWER_LIMIT** between 1 and 11, where the default value is 1 (lowest overhead) and 11 is the highest permissible value. This parameter is dynamic, so you could stop an in-progress rebalancing operation by setting it to 0 in real-time.

ASM automatically does data striping across disk groups in either of two modes:

- **Coarse** - Units of 1 M each
- **Fine** - Units of 128 K each

Mirroring is performed with the basic unit of the extent. (It is **not** performed on the volume level as in many operating system storage management products.) There are 3 kinds of mirroring in ASM:

Mirroring Type:	Mirroring:	Failure Groups Required:
High Redundancy	3 -way mirroring	3
Normal Redundancy	2 -way mirroring	2
External Redundancy	n/a	1

A failure group is a set of disks within a disk group that would fail as a unit. An example is a set of disks on the same controller. So, high redundancy means that 2 of the 3 failure groups could fail and the data would still be accessible. Normal redundancy only allows for failure of one failure group, as it is based on a two-way mirror. External redundancy requires only a single failure group. Either the resource is not critical enough for mirroring, or you are relying on mirroring or facilities external to Oracle (for example, in the operating system or a RAID disk system or an enterprise storage subsystem).

ASM supports commands to manage its disk groups. First, here are the views to get ASM instance information. Contents of the views vary slightly between the ASM and database client instances. This table is described from the view of the ASM instance:

ASM Instance View:	Usage:
V$ASM_CLIENT	Lists the database clients using ASM's disk groups.
V$ASM_DISK	Lists all available ASM disks (whether in a disk group or not).
V$ASM_DISKGROUP	Lists the disk groups.
V$ASM_FILE	Lists the files in every mounted disk group.
V$ASM_OPERATION	One row for each long-running ASM operation. Look here to see the status of ASM operations.
V$ASM_TEMPLATE	Lists the templates for the disk groups.
V$ASM_ALIAS	Lists aliases for mounted disk groups.

To manage disks and diskgroups, query **V$ASM_DISK** to see what disks are available to ASM. Use the **CREATE DISKGROUP** statement to create a disk group:

```
SQL> CREATE DISKGROUP my_group1 HIGH REDUNDANCY
        FAILGROUP fg1    DISK '/dev/s/raw1' NAME fg1d1
        FAILGROUP fg2    DISK '/dev/s/raw2' NAME fg2d2
        FAILGROUP fg3    DISK '/dev/s/raw3' NAME fg3d3
        FAILGROUP fg4    DISK '/dev/s/raw4' NAME fg4d4 ;
```

View **V$ASM_DISKGROUP** for information on the disk groups. View **V$ASM_OPERATION** to see the status of any ongoing ASM operation.

Drop a diskgroup using the **DROP DISKGROUP** statement:

```
SQL> DROP DISKGROUP my_group1 ;
```

Include the keywords **INCLUDING CONTENTS** if the diskgroup has data:

```
SQL> DROP DISKGROUP my_group1 INCLUDING CONTENTS ;
```

This example adds a disk to an existing disk group:

```
SQL> ALTER DISKGROUP my_group1
        ADD FAILGROUP fgx '/dev/s/raw7' NAME fgxd7 ;
```

Remember that adding the disk prompts ASM to automatically rebalance data in the background, transparent to the users. You can query **V$ASM_OPERATION** to see how the rebalancing is going. Use a statement like this one to change the priority of the rebalancing operation:

```
SQL> ALTER DISKGROUP my_group1 rebalance power 10 ;
```

This example statement drops a disk from a disk group:

```
SQL> ALTER DISKGROUP my_group1 drop disk fgxd7 ;
```

UNDROP DISKS cancels a pending drop of a disk from a disk group. (If the operation is already completed, it's no longer "pending" and the **UNDROP DISKS** has no effect). This example drops a disk, then cancels the operation if it's not yet completed:

```
SQL> ALTER DISKGROUP my_group1 drop disk fgxd7 ;
SQL> ALTER DISKGROUP my_group1 undrop disks ;
```

This example drops one disk and adds another to a disk group in a single statement. The benefit is that you'll only cause one rebalance operation, instead of the two that would be caused by issuing two separate SQL statements to perform the same functions:

```
SQL> ALTER DISKGROUP my_group1
        ADD FAILGROUP fgx '/dev/s/raw7' NAME fgxd7
        DROP DISK fg1d1 ;
```

You can **DISMOUNT** a disk group, making it unavailable to client instances. You can also **MOUNT** it to make it available:

```
SQL> ALTER DISKGROUP my_group1 DISMOUNT ;
SQL> ALTER DISKGROUP my_group1 MOUNT ;
```

Verify the internal consistency of a disk group by **CHECK ALL**:

```
SQL> ALTER DISKGROUP my_group1 CHECK ALL ;
```

The following table summarizes the ASM **ALTER DISKGROUP** keywords and commands:

ALTER DISKGROUP Statement Keywords:	**Usage:**
...check all	Verifies diskgroup internal consistency
... drop disk	Drops a disk from a diskgroup (and auto-rebalances)
... drop ... add	Drops a disk and adds a disk (and auto-rebalances one time)
… add …	Adds a disk to a diskgroup (and auto-rebalances)
...mount	Renders a diskgroup available to client instances
...dismount	Renders a diskgroup unavailable to client instances

You can perform all these operations from the EM GUI. Off the main **Database Control** panel, pick the **Administration** tab, then select the **Disk Groups** link.

ASM File Types and File Naming

ASM supports all Oracle database files except those outside of the database proper (such as the Oracle executables, user traces files, etc). Here is a list of the files ASM supports:

- Datafiles
- Control files
- Online Redo logs
- Archived Redo logs
- Temp files
- RMAN files (datafile backups, datafile copies, incremental backups, etc)
- Flashback logs
- Data Pump datasets
- others

ASM uses templates to enable it to work with all these different kinds of files. There is a separate template for each file type. The templates give ASM the information it needs to manage the file type described by each template.

All ASM files are Oracle-Managed Files (or OMF files). OMF simplifies file administration. For example, when you create a tablespace, you do not code file specifics, just provide the name of the tablespace and the disk group to which it belongs. This example creates new tablespace **my_ts3** and associates it with a disk group:

```
SQL> CREATE TABLESPACE my_ts3 datafile '+my_group1';
```

There are 6 different ways to refer to an ASM file. It is important to remember the ways to refer to ASM files and when each may be used. Here are the 6 different approaches:

Fully-Qualified File Names -- Used **only** when referencing an existing file.
Format: +group/dbname/file_type/tag.file.incarnation
Example: +my_group1/prod/datafile/data02.256.1

Numeric Names -- Used **only** when referencing an existing file.
Example: +my_group1.256.1

Alias Names -- Use to create one file **or** reference an existing file. Create the alias name through the **ALTER DISKGROUP ADD ALIAS** statement.
Alias with Template Names -- Use **only** when creating a new file. Allows you to specify the template to use along with the alias.

Incomplete Names -- Use for creating a new file or group of files. This example creates a new tablespace in a disk group, and uses the default template for the kind of file it creates:

```
SQL> CREATE TABLESPACE my_ts2 DATAFILE'+my_group1';
```

Incomplete Names with Template -- Use for creating a new file or group of files. This example creates a new tablespace in a disk group and uses the template specified:

```
SQL> CREATE TABLESPACE my_ts2
       DATAFILE'+my_group1(tempfile)';
```

ASM Initialization Parameters

ASM instances use a number of ASM-unique instance initialization parameters:

Initialization Parm:	Usage:
INSTANCE_TYPE	Set to **ASM** for an ASM instance (the alternative is the default, **RDBMS**, which indicates a database instance)
DB_UNIQUE_NAME	Defaults to **+ASM**. Only set this differently if you're running multiple ASM instances on one node.
ASM_DISKSTRING	Defines what disks will be available to the ASM instance. Special value **NULL** means all disks are visible to ASM.
ASM_DISKGROUPS	Lists the diskgroups for ASM to use.
ASM_POWER_LIMIT	How much resource to dedicate to background rebalancing. Ranges from 1 (low) to 11 (high) with default of 1.
LARGE_POOL_SIZE	Must be **8 M or larger.**

New ASM Background Processes

ASM uses several new background processes you need to know about. Two run on the ASM instance, while two run on the client instances. *Note that **RBAL** has the same name on the ASM and client instances but has different functions on each*:

Background Process:	Usage:	Runs Where:
RBAL	Coordinates balancing data across disk groups.	ASM instance
ORBn	Performs the actual rebalancing of data across disk groups. (**n** is 0 thru 9).	ASM instance
OSMB	Communicates to the ASM instance.	Client instance
RBAL	Performs open/close of disks on behalf of the database instance.	Client instance

Migrating a Database to ASM

Use Recovery Manager (RMAN) to move files from a regular database instance into an ASM instance. Here are the steps:

1. Backup the control file to trace and remember datafile, control file, and online redo log file names
2. Shutdown Normal or Immediate
3. Backup the database
4. Edit the SPFILE
   - Remove **CONTROL_FILES** parameter
   - Use OMF for all file destinations
5. Run this script, editing it to cover all log files:

```
STARTUP NOMOUNT;
RESTORE  CONTROLFILE  FROM  '<controlfile location>';
ALTER DATABASE MOUNT;
BACKUP AS COPY DATABASE FORMAT
'+<disk group destination>' ;
SWITCH  DATABASE  TO  COPY;
# Repeat the next statement for all log files …
SQL "ALTER DATABASE RENAME <logfile_1>
        TO  '+<disk group destination>'  ";
ALTER  DATABASE  OPEN  RESETLOGS ;
```

The Scheduler

In Oracle 10g, a new scheduler based on package **DBMS_SCHEDULER** supersedes the old **DBMS_JOB** package (which is still present in 10g). Use the new package by coding or through the EM.
Key components of the new scheduler include:

- **Job Table** - Master table of all enabled jobs in the database
- **Job Coordinator** - New background process **CJQn** (the Job Coordinator) ensures that jobs are run according to schedule
- **Job Slave Processes** - These execute jobs assigned to them by the scheduler

Package **DBMS_SCHEDULER** offers more power than the older **DBMS_JOB** package and it has these advantages:

- Executes 3 types of jobs (as specified by the **JOB_TYPE** parameter):
  - **EXECUTABLE** - Executable scripts
  - **STORED_PROCEDURE** - Database stored procedures
  - **PLSQL_BLOCK** - Anonymous blocks
- Is based on these key components:
  - **Job** - A program to run at a specified time
  - **Program** - Metadata on what to run (program name, type, arguments)

- **Schedule** - When and how often to run the task
- **Window** - Activates different resource plans at different times
- **Window Group** - A group of Windows
- **Job Class** - A group of Jobs with the same resource requirements

Jobs, Programs, and Schedules are created in ***user schemas***.

Job Classes, Windows, and Window Groups are created in the ***SYS*** schema.

The following table shows key **DBMS_SCHEDULER** procedures through which you can manage the above components:

DBMS_SCHEDULER Package:	**Usage:**
CREATE_JOB	Creates a Job
CREATE_JOB_CLASS	Creates a Job Class
CREATE_PROGRAM	Creates a Program
CREATE_SCHEDULE	Creates a Schedule
CREATE_WINDOW	Creates a Window
DEFINE_PROGRAM_ARGUMENT	Defines an argument for a Program
SET_JOB_ARGUMENT_VALUE	Sets parameters for a Job
SET_ATTRIBUTE	Sets an attribute for a scheduler component
SET_ATTRIBUTE_NULL	Sets an attribute to null (unsets an attribute)
ENABLE	Enables a Job or Program *By default, the ENABLED attribute is set to FALSE for Jobs and Programs.*
DISABLE	Disables a Job or Program
RUN_JOB	Runs a Job. Set attribute **USE_CURRENT_SESSION** to **TRUE** to immediately run a job, or to **FALSE** (the default) to submit the job to the Scheduler for normal asynchronous execution.
STOP_JOB	Stops a Job via a graceful interrupt. Setting attribute **FORCE** to **TRUE** may stop the job more quickly.
RESET_JOB	Clears values for Job arguments
COPY_JOB	Copies all the attributes of an existing Job to a new Job

DROP_JOB	Drops a Job from the Scheduler permanently. Setting attribute **FORCE** to **TRUE** issues an implicit **STOP_JOB** to stop a currently-running instance and then drop the job.
DROP_JOB_CLASS	Drops a Job Class from the Scheduler.
DROP_PROGRAM	Drops a Program from the Scheduler.
DROP_PROGRAM_ARGUMENT	Drops arguments of a Program.
DROP_SCHEDULE	Drops a Schedule.
DROP_WINDOW	Drops a Window.
PURGE_LOG	Deletes the Scheduler's log files.

Jobs

The Job component is the only absolutely required component to run a task. A Job can belong to only one Job Class. If the Job Class is not specified, the Job is assigned to the **DEFAULT_JOB_CLASS**. A Job's **JOB_PRIORITY** attribute can be set from **1** to **5** (1 is highest priority, 5 is lowest priority).

Here's example code to create a Job:

```
EXEC   SYS.DBMS_SCHEDULER.CREATE_JOB(
job_name          => 'SCOTT.BACKUP',
job_type          => 'EXECUTABLE',
                          -- or PLSQL_BLOCK or STORED_PROCEDURE –
job_action        => '/scotts_scripts/run_backup.sh',
start_date        => to_date('08-03-2005 20:00:00','mm-dd-yyyy hh24:mi:ss'),
repeat_interval   => 'TRUNC(SYSDATE) + 23/24',
comments          => 'Scotts backup job',
enabled           => TRUE);      -- the default for ENABLED is FALSE --
```

The **REPEAT_INTERVAL** attribute specifies how often a Job or Schedule repeats. You can use a calendaring expression to specify it. Calendaring expressions are very flexible, so there are usually several different ways to specify the same Job or Schedule **REPEAT_INTERVAL** when using them.

Calendaring Expressions have up to 3 parts:

Component:	**Keyword:**	**Values:**
Frequency	FREQ	YEARLY, MONTHLY, WEEKLY, DAILY, HOURLY, MINUTELY, SECONDLY
Interval	INTERVAL	An integer between 1 & 999 inclusive
Specifier	BYMONTH, BYWEEKNO, BYYEARDAY, BYMONTHDAY, BYDAY, BYHOUR, BYMINUTE, BYSECOND	Allowable values depend on the Keyword you use

Frequency is the only mandatory part of a Calendaring Expression.

Examples:

Calendaring Expression:	**Meaning:**
FREQ=HOURLY; INTERVAL=8	Every 8 hours
FREQ=YEARLY;BYWEEKNO=4;BYDAY=SUN	The Sunday in the 4th week of the year
FREQ=HOURLY;BYMONTHDAY=1,-1	Hourly on the 1st and last days of the month

This example code creates a Schedule and uses a calendaring expression in its **REPEAT_INTERVAL**:

```
DBMS_SCHEDULER.CREATE_SCHEDULE(
repeat_interval          => 'FREQ=WEEKLY;BYDAY=SAT;BYHOUR=12',
start_date               => systimestamp,
comments                 => 'Saturday batch schedule',
schedule_name            => 'SAT BATCH'  ) ;
```

Just as you use **DBMS_SCHEDULER** procedure **CREATE_JOB** to create a job, you can use procedure **RUN_JOB** to run a job, **STOP_JOB** to stop a running job via an interrupt, **DROP_JOB** to permanently remove a job from the scheduler database, and **COPY_JOB** to make a copy of a job.

Job Classes

The **RESOURCE_CONSUMER_GROUP** attribute for a Job Class associates a Resource Consumer Group with the Job Class. If not defined, it defaults to **DEFAULT_CONSUMER_GROUP**. So Job Classes are how you group jobs into categories that require similar resources to run.

The **LOGGING_LEVEL** attribute for each job class can be specified as either:

- **LOGGING_OFF** - No logging
- **LOGGING_RUNS** - Logs info on all runs in the class; DEFAULT
- **LOGGING_FULL** - Logs info on all runs plus operations like Enable, Disable, Alter, etc.

Use **DBMS_SCHEDULER** procedure **CREATE_JOB_CLASS** to create a new job class, and use procedure **DROP_JOB_CLASS** to drop a job class from the Scheduler.

Windows

A Window is a time period during which a particular resource plan is active to govern resource allocation for the database. Only one Window may be active at any given time. For overlapping Windows, the one with the higher priority remains active. For Windows with the same priority, the one that started first takes precedence.

This example code uses the **DBMS_SCHEDULER** procedure **CREATE_WINDOW** to create a Window. The **DURATION** is how long the Window is open for:

```
DBMS_SCHEDULER.CREATE_WINDOW(
window_name          => 'STD_PROCESSING',
resource_plan        => "PROD_PLAN',
start_date           => systimestamp at time zone 'US/Central',
duration             => numtodsinterval(6, 'hour'),
repeat_interval      => 'FREQ=DAILY;BYHOUR=6',
window_priority      => 'HIGH',
comments             => 'evening batch time'  ) ;
```

PURGE_LOG Procedure

The **PURGE_LOG** procedure in the **DBMS_SCHEDULER** package purges the Scheduler logs. The **LOG_HISTORY** attribute tells how many days of history to keep. The **WHICH_LOG** attribute tells which log to purge:

- **JOB_LOG** - For purging Job Logs only
- **WINDOW_LOG** - For purging Window Logs only
- **JOB_AND_WINDOW_LOG** -To purge both logs (this is the default)

This example code deletes both Job and Window log entries older than 14 days:

```
DBMS_SCHEDULER.PURGE_LOG(
log_history          => 14,
which_log            => 'JOB_AND_WINDOW_LOG' );
```

Scheduler Privileges, Views, Etc.

The following table shows key privileges in using the new Scheduler:

Privilege:	**Meaning:**
CREATE JOB	For creating a Job, Program, or Schedule *in your own schema*
CREATE ANY JOB	For creating a Job, Program, or Schedule *in any user's schema*
EXECUTE ANY PROGRAM	Required to use components created by others
EXECUTE ANY CLASS	Gives user the ability to assign a Job to any Job Class

MANAGE_SCHEDULER system privilege	Required to create and manage Job Classes, Windows, and Window Groups.
SCHEDULER_ADMIN Role	Includes CREATE JOB, CREATE ANY JOB, EXECUTE ANY PROGRAM, EXECUTE ANY CLASS, and MANAGE_SCHEDULER privileges. DBA Role has this Role by default.

Windows and Window Groups have **PUBLIC** access so everyone can access them without any special privileges.

Key Data Dictionary Tables:

Table:	**Contains Information About:**
DBA_SCHEDULER_JOBS	Jobs
DBA_SCHEDULER_PROGRAMS	Programs
DBA_SCHEDULER_JOB_CLASSES	Job Classes
DBA_SCHEDULER_JOB_LOG	Log info on Scheduler Jobs
DBA_SCHEDULER_JOB_RUN_DETAILS	Log run details
DBA_SCHEDULER_RUNNING_JOBS	Jobs that are currently running
DBA_SCHEDULER_JOB_ARGUMENTS	Jobs' Arguments
DBA_SCHEDULER_SCHEDULES	The Schedules
DBA_SCHEDULER_WINDOWS	Windows
DBA_SCHEDULER_WINDOW_GROUPS	Window Groups
DBA_SCHEDULER_WINDOW_LOG	Window Logs
DBA_SCHEDULER_WINGROUP_MEMBERS	Window Group Members

Many of these tables have **USER_** or **ALL_** equivalents. For example, for view **DBA_SCHEDULER_JOBS**, there are also views called **ALL_SCHEDULER_JOBS** and **USER_SCHEDULER_JOBS**.

We've shown how to create Scheduler objects (objects the Scheduler manages), like jobs and programs. It is important to understand that these are disabled (by default) when you create them. To enable them prior to use, you can use the **DBMS_SCHEDULER** package **ENABLE** procedure. Here's an example:

```
DBMS_SCHEDULER.ENABLE('SCOTT.BACKUP') ;
```

Execute **DISABLE** to disable a schedule object:

```
DBMS_SCHEDULER.DISABLE('SCOTT.BACKUP') ;
```

You can change any attribute for a Scheduler object except its name. Use procedure **SET_ATTRIBUTE** to do this. If you need to un-set an attribute, use procedure **SET_ATTRIBUTE_NULL** for this purpose. Changes do not apply to a currently-running object, just to its future instantiations.

Automatic Database Management

The single most defining characteristic of 10g versus prior Oracle releases is that 10g automatically manages itself. This is called the Common Manageability Infrastructure, or CMI, and it encompasses system-wide statistics collection, memory management, storage management, self-diagnostics, and many other self-managing features.

Figure 2: Common Manageability Infrastructure

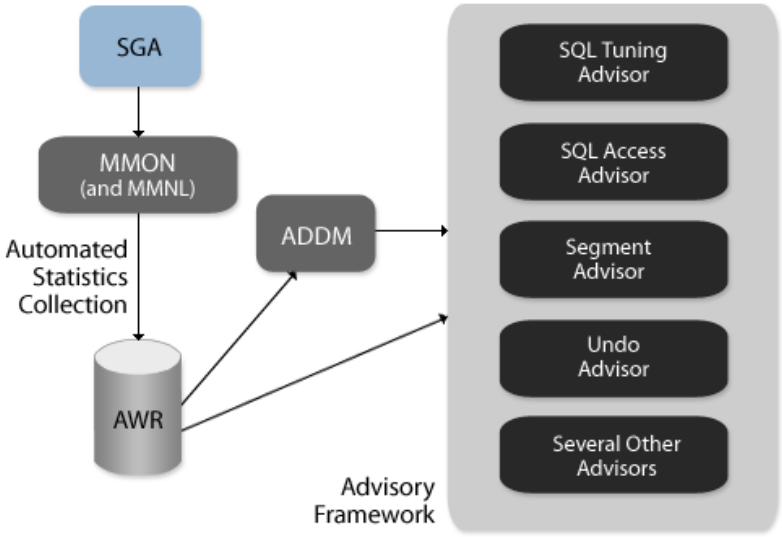

Figure 2 diagrams the CMI and its key components.

A key part of CMI is the Advisory Framework, a set of components that provides advice on how to tune Oracle. The Automatic Database Diagnostic Monitor, or ADDM, is a built-in, real-time, self-diagnostic performance monitor. Other components in the Advisory Framework automatically tune memory: the PGA Advisor, the Buffer Cache Advisor, and the Shared Pool Advisor. The Segment Advisor and the Undo Advisor help tune storage. And the SQL Tuning Advisor and the SQL Access Advisor help you tune data access through SQL statement and database design advice, respectively.

Performance Statistics

10g supersedes 9i's Statspack for database performance statistics collection and analysis. 10g's new approach is *not* compatible with Statspack and provides no migration of or integration with Statspack statistics.

10g performance statistics are collected into a memory buffer that is allocated within the Systems Global Area, or SGA, the basic memory area used by Oracle. Statistics are then written from the SGA buffer and stored onto disk in the new Automated Workload Repository, or AWR. The AWR physically resides in the new required **SYSAUX** tablespace. Data is owned by **SYS** and may be viewed through new data dictionary and V$ views. All AWR tables begin with the letters **WR**.

The new background process **MMON** (Memory Monitor) writes statistics from the SGA statistics collection buffer to the AWR by default every 1 hour. These are hourly snapshots. The new background process **MMNL** (Memory Monitor Light) flushes the statistics to the AWR whenever the SGA buffer area that holds them becomes full. (Note – you may sometimes see MMON expanded as "Memory Manageability Monitor").

Thus, snapshots of statistics are written to AWR by MMON at hourly intervals or by MMNL whenever the SGA memory buffer collection area becomes filled. The process is fast because it is fully integrated into Oracle internals. AWR performance statistics and workload information is saved by default for **7 days**.
The 3 kinds of performance statistics 10g collects are:

- Cumulative values - Accumulated performance statistics, some persistent across instance shutdowns and startups
- Metrics - These represent change rates for various parameters
- Sampled data - Samples current state of all active sessions

The new **STATISTICS_LEVEL** initialization parameter is key. It determines the level of statistics collection and analysis and may be set to any of 3 values:

- **BASIC** - Disable AWR statistics and self-tuning capabilities
- **TYPICAL** - **Default** level of statistics collection
- **ALL** - Collect exhaustive statistics

To turn off AWR statistics and self-diagnostics, set **STATISTICS_LEVEL = BASIC**.
The statistics AWR collects include:

- **Time model statistics** that indicate time spent in various activities
- **Object statistics** on database feature usage
- **V$SYSSTAT** and **V$SESSTAT** wait class statistics
- **High-load SQL** statements
- **Operating System** statistics

In addition to AWR's hourly snapshots of the above information, Active Session History or ASH samples the sessions activity in view **V$SESSION** *every second***.** ASH fills the vacuum of activity left by AWR, which operates on an hourly basis, since ASH samples **V$SESSION** every second.

Query view **V$ACTIVE_SESSION_HISTORY** for ASH information, it contains one row for each active session per sample. Or query the higher-level view **DBA_HIST_ACTIVE_SESS_HISTORY**, which contains a sampled history of **V$ACTIVE_SESSION_HISTORY** information.

The ASH resides in the System Global Area (SGA). Its size is calculated as the lesser of:

- Total number of CPUs * 2 M of memory
- 5 percent of the Shared Pool size

This means that there are only two ways to increase the size of the ASH:

- Increase the number of CPUs
- Increase the size of the Shared Pool

Do not confuse the AWR or ASH statistics with traditional optimizer statistics. Optimizer statistics are the statistics Oracle collects on tables and indexes in order to determine optimal data access strategies or access paths.

Query view **DBA_HIST_DATABASE_INSTANCE** to see which database instances the AWR tracks. If you have the **SELECT ANY DICTIONARY** privilege, you can run any of the 3 key AWR reports:

- **awrrpt.sql** - Reports detailed statistics for a range of snapshot IDs in either text or HTML formats
- **awrrpti.sql** - Reports the same but you specify the database and instance IDs as input
- **awrinfo.sql** - General information on AWR snapshots and ASH use

The **DBMS_WORKLOAD_REPOSITORY** package allows you to manage snapshots yourself. You can also manage baselines: a measurement point derived from a set of snapshots. Baselines are used for comparison purposes when a problem occurs.

DBMS_WORKLOAD_REPOSITORY procedures include:

- **CREATE_SNAPSHOT** – manually creates a snapshot at some time other than the standard 1-hour interval.
- **DROP_SNAPSHOT_RANGE** – drops a range of snapshots by specifying low and high snapshot IDs. Also drops the ASH history within that time frame from **DBA_HIST_ACTIVE_SESS_HISTORY**.
- **CREATE_BASELINE** – creates a baseline based on a range of snapshot IDs.
- **DROP_BASELINE** – drops a baseline by name. The **CASCADE** parameter drops snapshots related to the baseline you drop (the default for **CASCADE** is **FALSE**, so baseline-related snapshots are not dropped by default). The snapshots belonging to a baseline are not dropped until the baseline is dropped.
- **MODIFY_SNAPSHOT_SETTINGS** – modifies the snapshot capture and purging policy. Two key parameters to remember:
  - **RETENTION** - The new retention period in minutes
  - **INTERVAL** - The new snapshot interval in minutes

Here are a couple examples of the **DBMS_WORKLOAD_REPOSITORY** procedures. Using **MODIFY_SNAPSHOT_SETTINGS,** this example sets the **RETENTION** period for snapshots to 2,000 minutes:

```
DBMS_WORKLOAD_REPOSITORY.MODIFY_SNAPSHOT_SETTINGS -
        (RETENTION=>2000) ;
```

This example sets the snapshot **INTERVAL** to 20 minutes:

```
DBMS_WORKLOAD_REPOSITORY.MODIFY_SNAPSHOT_SETTINGS -
        (INTERVAL=>20) ;
```

This statement creates a snapshot on demand:

```
DBMS_WORKLOAD_REPOSITORY.CREATE_SNAPSHOT() ;
```

This statement drops the baseline named 'my_base':

```
DBMS_WORKLOAD_REPOSITORY.DROP_BASELINE -
        (BASELINE_NAME=>'my_base') ;
```

The key dictionary views for snapshot information are:

- **DBA_HIST_SNAPSHOT** – Snapshot info with startup time and details
- **DBA_HIST_BASELINE** – Lists snapshot IDs and baseline time range
- **DBA_HIST_WR_CONTROL** – Lists current snapshot settings for **RETENTION** and **INTERVAL**

Base statistics are raw data. Metrics describe base statistics in terms of time frames. Background process **MMON** creates and updates metric data.

Server-based Alerts

The 10g CMI includes server-generated alerts. These notify you of important database events. Alerts trigger either because:

1. A threshold is exceeded
2. An event has occurred

MMON sends Alerts to the predefined Alert Queue owned by **SYS** named **ALERT_QUE**. They also display on the EM Database Control panels and get sent to administrators via **email** or **pager** (depending on how you configured the feature).

Threshold alerts are also known as stateful alerts. They must occur a specified number of times within a time period to be raised (this avoids false alerts). Threshold alerts appear in view **DBA_OUTSTANDING_ALERTS** and when cleared are moved to view **DBA_ALERT_HISTORY** with a status of **CLEARED**. Non-threshold alerts are better known as event alerts or stateless alerts. These go directly to **DBA_ALERT_HISTORY** and never appear in **DBA_OUTSTANDING_ALERTS.**

Oracle provides 4 "out-of-the-box" alerts by default:

- Recovery area low on free space
- Snapshot too old
- Resumable session suspended
- Tablespace space is running out
    - **85% used** means a **Warning message**
    - **97% used** means a **Critical message**

The new 10g background process **MMON** checks for tablespace space usage problems every 10 minutes. To avoid false alerts, thresholds on tablespaces are computed in one of 2 ways: based on the maximum size specified when the tablespace was created, or based on the maximum operating system file size -- whichever is smaller.

While the Tablespace Space Usage alert is enabled by default for a newly created 10g database, it is disabled by default for databases upgraded to 10g from prior releases. Note that the tablespace space usage threshold only applies to locally-managed tablespaces (not dictionary-managed ones).
Important server alert views are:

View:	**Usage:**
V$ALERT_TYPES	Lists all the possible alert types and whether each is threshold or stateless
DBA_THRESHOLDS	Shows the threshold settings
DBA_ALERT_HISTORY	Alerts that have been cleared
DBA_OUTSTANDING_ALERTS	Current alerts for your resolution
V$METRIC	Metric values
V$METRIC_NAME	Identifies the metrics
V$METRIC_HISTORY	History for the metrics

Thresholds can be set either by EM Database Control panels or through the procedures in the new 10g **DBMS_SERVER_ALERT** package. For EM, from the **Database Control** panel you can choose the **Edit Thresholds** button. This presents the **Edit Thresholds** screen, where you can edit thresholds for any database metric. You can also define a response action there – a SQL command or script to automatically address the threshold issue.

The 3 **DBMS_SERVER_ALERT** procedures to know are:

- **GET_THRESHOLD** - Retrieve threshold values
- **SET_THRESHOLD** - Set threshold values
- **EXPAND_MESSAGE** - Translates numeric message to text

Parameters are the same for **SET_THRESHOLD** and **GET_THRESHOLD**, except that **WARNING_OPERATOR** and **CONSECUTIVE_OCCURRENCES** are **OUT** parameters for **GET_THRESHOLD** instead of **IN**:

GET_THRESHOLD and SET_THRESHOLD Parameters:	**Usage:**
METRICS_ID	Metric name
WARNING_OPERATOR	Comparison operator for the warning metric
WARNING_VALUE	Warning threshold value or **NULL** for none
CRITICAL_OPERATOR	Comparison operator for the critical metric
CRITICAL_VALUE	Critical threshold value or **NULL** for none
OBSERVATION_PERIOD	Time period for comparison (from 1 to 60 minutes)
CONSECUTIVE_OCCURRENCES	How many times to exceed the threshold before the alert is issued
INSTANCE_NAME	Instance name (or **NULL** for RAC instances and database-wide alerts)
OBJECT_TYPE	Object type that is observed
OBJECT_NAME	Object name

Here is an example that sets a Tablespace Space Usage alert for the **TOOLS** tablespace to greater than or equal to 90% for a Warning, and greater than or equal to 95% for Critical:

```
DBMS_SERVER_ALERT.set_threshold(
dbms_server_alert.tablespace_pct_full,
dbms_server_alert.operator_ge, 90,
dbms_server_alert.operator_ge, 95,
1, 1, null,
dbms_server_alert.object_type_tablespace, 'TOOLS') ;
```

The Alert log is Oracle's basic mechanism for logging and communicating key database events. Alert log thresholds and their corresponding server-based alerts help you keep on top of important errors that Oracle writes to the Alert log. Alerts are automatically enabled, by default, for these Alert log metrics:

Alert Log Metric:	**Meaning:**
Alert log error trace file	The system has generated a trace file
Alert log name	Oracle checks name & location of Alert log
Archiver hung error	"Archiver is hung" message in the Alert log
Data block corruption error	Alert log notes data block corruption
Generic Alert log error	Generic database error ORA-00600 occurred
Session-terminated error	Session terminated ORA-00603 error in Alert log

Automatic Database Diagnostic Monitor (ADDM)

MMON triggers ADDM analysis each time a snapshot is taken on the last two snapshots. This proactively monitors the database for any developing problems. Analysis is top-down with the goal of reducing everything to the single metric DB time (time spent on processing user requests).

Findings divide into *problem* (root cause), *symptom*, or *information*. ADDM may recommend running one of the other Advisors, such as the SQL Tuning Advisor.

Access ADDM findings through EM (off the Performance link) or through view **DBA_ADVISOR_FINDINGS**. You can also run report scripts **addmrpt.sql** or **addmrpti.sql** (takes Database and Instance IDs as input parameters).

DBA_ADVISOR_RECOMMENDATIONS shows the result of diagnostics runs with recommendations for the problems identified in each run.

You can set a few of the ADDM parameters by using **DBMS_ADVISOR.SET_DEFAULT_TASK_PARAMETER**. View current parameters through view **DBA_ADVISOR_DEF_PARAMETERS**.

Like all 10g's automation features, ADDM is enabled by default (because by default **STATISTICS_LEVEL** is **TYPICAL**). A **STATISTICS_LEVEL** setting of **ALL** also keeps ADDM on while the setting of **BASIC** turns it off.

Optimizer Statistics

So far, this Exam Manual has been discussing performance statistics. These differ from optimizer statistics – those statistics Oracle requires to develop *optimal data access paths*.

Oracle now collects 4 kinds of optimizer statistics for the optimizer to use:

- Dictionary - New to 10g, statistics on the data dictionary
- System - Hardware characteristics like CPU and disk I/O speeds
- Operating System - New to 10g, operating system level statistics
- User-defined - Statistics on user-defined functions, packages, etc.

Traditional optimizer statistics (on tables, indexes, etc) are automatically collected in a 10g database. When you create a database, Oracle creates a job called **GATHER_STATS_JOB** and places it in the Scheduler. It runs in the default Maintenance Window, defined as weeknights from 10pm to 6am, and all day long on weekends. A procedure named **DBMS_STATS.GATHER_DATABASE_STATS_JOB_PROC** actually gathers the statistics.

SQL Tuning Advisor

Among the several Advisors that comprise the Common Manageability Infrastructure, or CMI, the new SQL Tuning Advisor is one of the few covered on this Exam. The SQL Tuning Advisor automates SQL statement tuning. It operates in two modes:

- Normal
- Tuning

The former is real-time, while the latter takes minutes for its analysis rather than seconds. Tuning mode provides recommended actions (and their rationales) to produce better SQL execution. The tuning mode is also called the Automatic Tuning Optimizer (ATO). This mode is great for improving high-load SQL.

The SQL Tuning Advisor has these main inputs:

- High-load SQL (identified by ADDM)
- SQL statements in the cursor cache
- SQL statements from the AWR
- A user-provided set of SQL statements
- A SQL Tuning Set (or STS)

User-defined SQL statements are *not* executed, they are merely analyzed. An STS is a set of SQL statements along with execution information (bind variables, how much to execute the SQL, etc).

The kinds of analysis the SQL Tuning Advisor does includes:

- Statistics analysis - ATO verifies statistics, recommends gathering new ones if appropriate
- SQL profiling - A SQL Profile includes info collected during optimization useful for further optimization during future runs

- Access path analysis - On a single-statement level, may recommend Indexes; this relates to how Oracle looks up the data queried (very CBO-centric)
- SQL structure analysis- Inspects the keywords used and structure of the SQL statement, recommends improvements

You can work with the SQL Tuning Advisor through EM Database Control or through the package **DBMS_SQLTUNE.** To use this package, a typical set of steps would be:

1. Use **DBMS_SQLTUNE.CREATE_TUNING_TASK** to create a task
2. Run **EXECUTE_TUNING_TASK**
3. Accept the SQL profile generated by **ACCEPT_SQL_PROFILE**

You need the **ADVISOR** privilege to do this, as well as the **CREATE ANY SQL PROFILE** privilege to create and save SQL Profiles. SQL Profiles are uniquely identified by a category name and SQL text, so you can have identical SQL statements in different profiles as long as each profile is in a different category. Categories are determined by the new initialization parameter **SQLTUNE_CATEGORY**. The default category is called **DEFAULT**. You can change the category at the session level with a statement like:

```
SQL> ALTER SESSION SET SQLTUNE_CATEGORY = PROD ;
```

Another useful **DBMS_SQLTUNE** procedure is **QUICK_TUNE**. Use it to tune a single SQL statement. Here is an example of how to use it:

```
VARIABLE        task_id    NUMBER ;
VARIABLE        task_name       VARCHAR2(255) ;
VARIABLE        sql_stmt  VARCHAR2(4000);
EXECUTE  :sql_stmt  :=  'SELECT COUNT(*) FROM emp WHERE empno=1434';
EXECUTE         :task_name      := 'QUICKTUNE';
EXECUTE DBMS_ADVISOR.QUICK_TUNE('SQL Access Advisor', -
                :task_id, :task_name, :sql_stmt ) ;
```

Automatic Shared Memory Management (ASMM)

Another one of Oracle's self-tuning features is called Automatic Shared Memory Management, or ASMM. This feature vastly simplifies the sizing of SGA memory by automatically sizing its internal components for you, based on some overall SGA target size you specify.

SGA_TARGET is a new dynamic initialization parameter that specifies the total size of SGA memory. Setting this to a non-zero value (plus setting **STATISTICS_LEVEL** to **TYPICAL** or **ALL**) enables ASMM. ASMM then takes responsibility for dynamically sizing several parts of the SGA as appropriate. **SGA_TARGET** cannot be set larger than non-dynamic initialization parameter **SGA_MAX_SIZE**, which specifies the maximum size of the SGA.

Memory components for which ASMM automatically configures memory are:

- **DB_CACHE_SIZE** - Database buffer cache default pool
- **SHARED_POOL_SIZE** - Shared pool
- **LARGE_POOL_SIZE** - The Large pool
- **JAVA_POOL_SIZE** - The Java pool

ASMM does **not** size these areas, which you still should manually size through initialization parameters:

- **LOG_BUFFER** - The log buffer
- **STREAMS_POOL_SIZE** - Size of the Streams pool
- Fixed-size SGA components and internal allocations
- Other buffer caches:
  - **DB_nK_CACHE_SIZE**
  - **DB_KEEP_CACHE_SIZE**
  - **DB_RECYCLE_CACHE_SIZE**

Here's an example of how this works -- You set **SGA_TARGET** to 800 M, you set **LOG_BUFFER** to 10 M, you set **DB_KEEP_CACHE_SIZE** to 100 M. This means ASMM distributes 800 - (10 + 100) or 690 M to the memory areas it dynamically sizes.

When **SGA_TARGET** is set to a non-zero value, the auto-tuned SGA parameters will have default values of 0. If you specify a non-zero value for any auto-tuned parameter, ASM takes that value as the lower limit for that component. Setting **SGA_TARGET** to 0 disables ASMM. The new background process Memory Manager (MMAN) does the memory sizing for ASMM. (Note-- MMAN is distinct from MMON).

You can see ASMM and memory configuration information on the Memory Parameters page of the EM GUI, or see view **V$SGA_DYNAMIC_COMPONENTS**.

Along with ASMM, 10g also supports Automatic PGA Memory Management (APMM). To enable APMM, just set initialization parameter **PGA_AGGREGATE_TARGET** to the overall size of PGA memory:

```
PGA_AGGREGATE_TARGET = 10M ( PGA target of 2G is a little high)
```

Or set it dynamically:

```
SQL> alter system set pga_aggregate_target = 10M;
```

Oracle will manage internal PGA components to the target you specify.

APMM is enabled by default. If you do not set **PGA_AGGREGATE_TARGET**, this initialization parameter defaults to either 10 M or 20% of the SGA size, whichever is larger.

Diagnostics

Even with all 10g's new automatic self-management features (like automatic statistics collection, ADDM, and ASMM), you will still have occasion to research diagnostics.

The Alert log is the primary source of diagnostic information about a database. It contains messages for such events as:

- Redo log switches
- Oracle internal database errors (ORA-00600)
- Checkpoints
- Recovery operations

- Startups and shutdowns, noting unique initialization parameters
- Scheduler errors (from **DBMS_SCHEDULER** and **DBMS_JOB**)
- Administrative create, alter, and drop events
- **ALTER SYSTEM** commands

The Alert log is written to the background dump destination defined by initialization parameter **BACKGROUND_DUMP_DEST**. The Alert log is continually appended to; you must manage its size. You can view the Alert log through EM Database Control's **Most Recent Alert Log Entries** panel.

Trace files provide detailed information on errors. There are 2 kinds of trace files:

Trace Type:	Written to Init Parm Location:	Trace File Name Format:
Background process trace	**BACKGROUND_DUMP_DEST**	*sid_process_name_process_id*.trc Example: prod_pmon_1433.trc
User process trace	**USER_DUMP_DEST**	*sid*_ora_*process_id*.trc Example: prod_ora_8872.trc

Set init parm **MAX_DUMP_FILE_SIZE** to limit the size of trace files. By default, this parameter is set to **UNLIMITED**.

Since it is sometimes hard to identify your own trace file among the many in the trace directories, you might use the **TRACEFILE_IDENTIFIER** parameter to uniquely identify your session. Use it like this:

```
SQL> alter session set tracefile_identifier = 'BOB';
```

Storage Management

Beyond Automatic Storage Management (ASM), the exam also requires that you know about several specific storage management topics. This section covers the topics relevant to the test.

Segment Shrink

10g now allows you to make segments smaller and eliminate internal unused space. You can optionally move the High Water Mark (HWM) down, freeing up space for other segments. This new feature is called segment shrink.

To perform segment shrink, you must first enable row movement, because the process can change some ROWIDs:

```
SQL> ALTER TABLE table_name ENABLE ROW MOVEMENT;
```

Then, shrink the segment and move the HWM:

```
SQL> ALTER TABLE table_name SHRINK SPACE ;
```

While rows are moved in the shrink operation, small numbers of rows are locked during the compaction process but the data in the table is fully available to other users. However, the entire table is locked for a very brief period while the HWM is moved down.

If you want to shrink space **without** moving the HWM, issue this statement:

```
SQL> ALTER TABLE table_name SHRINK SPACE COMPACT ;
```

Later you can finish the operation by issuing:

```
SQL> ALTER TABLE table_name SHRINK SPACE ;
```

Shrinking without the **COMPACT** keyword invalidates all cursors on the segment.

If you want to shrink all dependent objects (such as indexes), be sure to specify the **CASCADE** keyword:

```
SQL> ALTER TABLE table_name SHRINK SPACE CASCADE ;
```

You cannot shrink segments with these characteristics:

- Segments lacking automatic segment space management (in other words, segments that use freelists for space management)
- Tables with:
  - LONG columns
  - On-commit or ROWID-based materialized views
  - Function-based indexes
  - Clustered tables
- LOB segments
- IOT mapping tables or overflow segments

Use the new Oracle 10g Segment Advisor to identify segments that can benefit by shrinking. Locate this under the **Advisor Central** link in the EM **Database Control** panel. The associated Growth Trend Report predicts how much space a segment will need going forward based on past growth. The Segment Resource Estimation tool gives initial size estimates for a table based on column datatypes and the number of rows the table will hold.

You can also access the Segment Advisor through its **DBMS_ADVISOR** package procedures:

- **CREATE_TASK** - Creates a new Advisor task
- **CREATE_OBJECT** - Specifies the object to analyze within a task
- **SET_TASK_PARAMETER** - Specifies task analysis parameters
- **EXECUTE_TASK** - Executes the Advisor task
- **DELETE_TASK** - Deletes a task
- **CANCEL_TASK** - Terminates a running task

Index Space

To reclaim unused internal space within an index, you can either rebuild or coalesce the index. Here are the key characteristics of each method:

Index Rebuild:	**Index Coalesce:**
Creates a new index b-tree, adjusts it as beneficial during the rebuild	Retains the existing index and coalesces index leaf blocks within each branch
Requires double the space of the original index for online, active rebuilding	No requirement for more space during the coalesce
Can optionally move the index to another tablespace	Cannot move the index to a new tablespace

To identify indexes that would benefit from a rebuild or coalesce, query column **PCT_USED** from dictionary view **INDEX_STATS**. If this value declines over time, reclaim space from the index:

```
SQL> select pct_used from index_stats where name = 'MY_INDEX';
```

To rebuild the index online, use this statement:

```
SQL> ALTER INDEX my_index REBUILD ONLINE;
```

Or coalesce the index like this:

```
SQL> ALTER INDEX my_index COALESCE;
```

Oracle 10g's new SQL Access Advisor can give you advice about when you can improve access times by building new indexes.

Index Organized Tables (IOTs)

IOTs are a data structure that permit you to store the table data in an organized way with its index. This is beneficial when table access is mainly through the primary key and the primary key constitutes most of the table's columns. This contrasts to the default table organization, heap, wherein data is just placed where it fits in the table.

To create an IOT, just specify the additional **CREATE TABLE** keywords **ORGANIZATION INDEX** (and specify the primary key!), as in this example:

```
SQL> create table my_employee_iot
        (employee_id          number,
        employee_asgn     number,
        emp_pass              varchar2(30),
        constraint access_emp primary key
              (employee_id, employee_asgn) )
        organization index
        tablespace my_emps01;
```

To improve performance, you may want to create an IOT overflow area. An IOT row is automatically placed in the overflow area if the row's data exceeds the threshold of available space in the block. You add the **PCTTHRESHOLD** and **OVERFLOW TABLESPACE** clauses to the above example to define the threshold percentage and overflow area tablespace.

You can create indexes on an IOT. If they are bit-mapped indexes, you must create a mapping table. The mapping table maps the physical rowids to their corresponding logical rowids used in the IOT. This is necessary because IOTs do not use physical rowids. You need only one mapping table per IOT, regardless of how many bit-map indexes are defined on the IOT.

Clustered Tables

An index cluster contains data rows from two or more tables in shared data blocks. The rows are stored together for faster access because they are normally retrieved together anyway. This is an alternative to placing the data within a single "pre-joined" table.

Hash clusters allow you to store the data from a single table in a cluster. Sorted hash clusters maintain a sort order for any retrieved rows by the cluster key.

This table summarizes the 3 kinds of clustered tables. It shows what they are and when to use them:

Cluster Type:	**Best Usage:**
Index cluster	• The clustered tables' data is always retrieved together. Use when this could speed up joins that match the cluster design. • Little update activity after the initial loading of the cluster.Child tables have roughly equal numbers of rows for each row in the parent table.
Hash cluster	• Use when the table(s) would benefit from hashed access. • Tables have relatively uniform distribution of indexed values. • The number of values for the indexed column is predictable. • Little update activity after the initial loading of the cluster. • Queries almost always use the equality operator to retrieve rows.
Sorted hash cluster	• Goal is to retrieve rows stored in the order of the cluster key. • The **ORDER BY** clause is therefore not even required to retrieve rows in the order of the cluster key.

For index clusters, perform these 3 steps for set-up:

1. Create the cluster definition
2. Create the index on the cluster
3. Create tables within the cluster

For hash and sorted hash clusters, perform these 2 steps for set-up:

1. Create the cluster definition
2. Create the table within the cluster

Resumable Space Allocation

Resumable space allocation is a feature that ensures that a SQL statement that causes a session to run out of space is temporarily suspended – so that you can fix the problem – rather than see it rolled back. You can enable it via a new option on the **ALTER SESSION** statement:

```
SQL> ALTER SESSION RESUMABLE [TIMEOUT timeout] [NAME name];
```

Specify the **TIMEOUT** period in seconds. If you fail to do so, the default is taken from initialization parameter **RESUMABLE_TIMEOUT**. This statement grants the **RESUMABLE** system privilege to user **SCOTT**:

```
SQL> GRANT RESUMABLE TO scott ;
```

SCOTT then turns around and enables a resumable session with a timeout period of 900 seconds:

```
SQL> alter session enable resumable timeout 900 name 'My Timeout' ;
```

Later **SCOTT** chooses to disable the feature:

```
SQL> alter session disable resumable ;
```

Control resumable space allocation through the following procedures in the package **DBMS_RESUMABLE**:

Procedure:	**Usage:**
GET_SESSION_TIMEOUT	Returns timeout value for a session
GET_TIMEOUT	Returns timeout value for the current session
SET_SESSION_TIMEOUT	Sets timeout value for a session
SET_TIMEOUT	Sets timeout value for the current session
ABORT	Cancels a suspended resumable space transaction
SPACE_ERROR_INFO	Returns error information on an object that triggered resumable space allocation

Database Resource Manager (DRM)

Oracle's Database Resource Manager (DRM) is the vehicle through which you can control and prioritize access to computer resources from within Oracle. Its major components are:

DRM Component:	Usage:
Resource Consumer Groups	These allow you to classify users based on their resource requirements. Use these also to prioritize user access.
Resource Plans	Consists of Resource Plan Directives that specify how resources should be allocated among Resource Consumer Groups or other Resource Plans.
Resource Plan Directives	Assign Resource Consumer Groups to Resource Plans and define resource allocations for each.

To create these objects, you first create an area in memory called a pending area. Here are the typical steps involved in creating DRM components:

1. Create the pending area in memory. All subsequent steps place new information or objects into the pending area.
2. Create the resource consumer groups.
3. Create the resource plans.
4. Create the resource plan directives.
5. Validate all objects in the pending area.
6. Submit the pending area, thereby updating the data dictionary with the DRM objects.
7. Enable the resource plan you want to be active.

Let's walk through a complete example. You use various procedures in the package **DBMS_RESOURCE_MANAGER** to perform each step. First, issue a statement to create the pending area:

```
SQL>  exec dbms_resource_manager.create_pending_area() ;
```

Next, you would create some Resource Consumer Groups within the pending area. Use procedure **CREATE_CONSUMER_GROUP** to do this:

```
SQL>  exec dbms_resource_manager.create_consumer_group('programmers' ,
          'application programmer staff') ;
```

Key parameters for the **CREATE_CONSUMER_GROUP** procedure are:

- **CONSUMER_GROUP** - Name for the resource consumer group
- **COMMENT** - Textual comment
- **CPU_MTH** - Method to schedule CPU resources for the group. Choose either **ROUND_ROBIN** (the default)or **RUN_TO_COMPLETION** (favor top sessions)

You can update a consumer group by procedure **UPDATE_CONSUMER_GROUP** or delete it by **DELETE_CONSUMER_GROUP**.
Next, you will typically create Resource Plans. These prioritize resource allocations by levels, with **level 1** being the highest priority and **level 8** being the lowest. Resource Plans consists of resource plan directives that

specify how resources are allocated among the resource consumer groups (or other resource plans).

Resource plans are either simple resource plans or complex resource plans. Simple resource plans have only a single level (they have no sub-plans), and the only resource they allocate is CPU. Simple plans limit the total number of resource groups to 8. They always use the **EMPHASIS** CPU resource allocation policy, which interprets each CPU allocation as a percentage of the whole.

This creates a simple resource plan by the procedure **CREATE_SIMPLE_PLAN**:

```
dbms_resource_manager.create_simple_plan(
        simple_plan          => 'ex_simple_plan',
        consumer_group1 => 'programmers',
        group1_cpu           => 50,
        consumer_group2 => 'staff',
        group2_cpu           => 25,
        consumer_group3 => 'management',
        group3_cpu           => 25 );
```

To create a complex plan, use the procedure **CREATE_PLAN**. Update plans through procedure **UPDATE_PLAN** and delete them via **DELETE_PLAN**.

Now create some Resource Plan Directives to assign resource consumer groups to resource plans and allocate resources for each. Resource plan directives provide for the following allocation methods:

Allocation Method:	**Usage:**
CPU	Specifies how CPU is allocated among resource consumer groups or sub-plans.
Active session pool queuing	Limits number of concurrent active sessions for the resource consumer group.
Degree of parallelism limit	Specifies maximum degree of parallelism for operations within the resource consumer group.
Automatic consumer group switching	Specifies that sessions exceeding their execution time limit can be automatically switched to a different consumer group.
Canceling SQL and terminating sessions	Can terminate long-running sessions or queries.
Execution time limit	Maximum execution time for an operation.
Undo pool	Maximum amount of undo for an operation.
Idle time limit	Maximum idle time for a session.

Use procedure **CREATE_PLAN_DIRECTIVE** to create the resource plan directives. Here's an example:

```
dbms_resource_manager.create_plan_directive(
        plan                        => 'dayplan',
        comment                     => 'programmers day plan',
        group_or_subplan            => 'programmers',
        parallel_degree_limit_p1    => '3') ;
```

You create multi-level plan directives and sub-plans by using the same procedure, **CREATE_PLAN_DIRECTIVE**. Use **UPDATE_PLAN_DIRECTIVE** and **DELETE_PLAN_DIRECTIVE** to update and delete plans.

Understand how automatic consumer group switching works. This allows a session in a consumer resource group that has exceeded its CPU allocation to automatically get switched to another group. Code the **SWITCH_TIME** parameter to indicate the maximum execution time in seconds before a session is automatically switched to the group defined by the **SWITCH_GROUP** parameter.

An alternative to **SWITCH_TIME** is **SWITCH_TIME_IN_CALL**. The latter parameter switches a session to a new group when it exceeds its allocation, then switches it back after the offending operation is completed. **SWITCH_TIME_IN_CALL** thus avoids permanently switching the session (unlike **SWITCH_TIME**).

Now that you've created the consumer resource groups, resource plans, and resource plan directives in the pending area, you can validate and then submit the pending area:

```
SQL> exec dbms_resource_manager.validate_pending_area ;
SQL> exec dbms_resource_manager.submit_pending_area ;
```

While this places the DRM components in the data dictionary, it does not enable the resource plan. Only one resource plan can be enabled at any one time. You can enable the plan by the static initialization parameter **RESOURCE_MANAGER_PLAN** in the initialization parameter file (**PFILE**). Better yet, if you use an **SPFILE**, you can dynamically enable the appropriate resource plan:

```
SQL> alter system set resource_manager_plan = 'dayplan' scope=both ;
```

If you want to prevent the Scheduler from automatically changing the plan you enabled, use the **FORCE:** keyword:

```
SQL> alter system set resource_manager_plan = 'FORCE:dayplan' scope=both;
```

This ends the example of how to set up and implement DRM objects. The next section discusses a few other DRM details.

Submitting the pending area clears it. If you want to manually clear it, issue:

```
SQL> exec dbms_resource_manager.clear_pending_area ;
```

Clearing the pending area effectively drops it, so you'll need to create a new pending area if you want to submit more DRM work during your session.

Another procedure to know about is **SET_CONSUMER_GROUP_MAPPING**. This allows you to easily map sessions to resource consumer groups based on either:

- Login ids
- Session runtime attributes

Consumer group mapping makes it easier to manage the assignment of logins and sessions to resource consumer groups. You can even create mapping priorities through the **SET_MAPPING_PRIORITY** procedure.

We mentioned earlier about switching sessions among resource consumer groups. You can explicitly switch resource consumer groups for active sessions by these two procedures:

- **SWITCH_CONSUMER_GROUP_FOR_SESS** - By session
- **SWITCH_CONSUMER_GROUP_FOR_USER** - By login ID

Remember that you can identify sessions uniquely by retrieving the column values **SID** and **SERIAL#** from the dynamic view **V$SESSION**.

If a user has been granted the switch privilege, they can switch their own session by executing procedure: **DBMS_SESSION.SWITCH_CURRENT_CONSUMER_GROUP**.

Grant users this right by using package **DBMS_RESOURCE_MANAGER_PRIVS** procedure **GRANT_SWITCH_CONSUMER_GROUP**. Revoke this privilege by procedure **REVOKE_SWITCH_CONSUMER_GROUP.**

What about users you do not assign to any specific resource consumer group? By default, they are members of the **DEFAULT_CONSUMER_GROUP**. Users not assigned to a group in the currently enabled resource plan are automatically assigned to the **OTHER_GROUPS** group. These are 2 of the 4 resource consumer groups Oracle creates by default when you use DRM:

Default Resource Consumer Group:	**Usage:**
DEFAULT_CONSUMER_GROUP	Default for users not assigned to an initial resource consumer group.
OTHER_GROUPS	Default for users assigned to consumer groups, but that are not included in the currently-enabled plan.
SYS_GROUP	The **SYSTEM_PLAN** plan created by Oracle for users **SYS** and **SYSTEM.**
LOW_GROUP	Also used by the **SYSTEM_PLAN** plan created by Oracle.

Globalization

Globalization is the ability of the database to support different spoken languages, as well as the cultural accouterments of those languages, such as different ways of expressing dates, times, and numbers. Oracle 10g supports globalization through these features:

- Language support - Supports all human languages
- Territory support - Supports geographical cultural conventions
- Linguistic sorting/searching - Supports language-appropriate sorting, searching
- Character sets & semantics - Supports character sets for different languages
- Calendars - Supports 7 different international calendars
- Locale customization - Oracle's Locale Builder feature customizes definitions for language, character set, territory, etc, by locale

The Oracle National Language Support Runtime Library (NLSRTL) implements globalization. NLSRTL looks for the file **lx1boot.nlb** at startup in the location defined by environmental variable **ORA_NLS10**, or if that is not defined, in the default location **$ORACLE_HOME/nls/data**. The file **lx1boot.nlb** defines a set of locales, in the form of a number of locale definition files. Implementing locale data in locale files gives Oracle modularity and flexibility in dealing with globalization and locale-specific needs.

The locale files reside in the same directory as the **lx1boot.nlb** file itself. There are 4 kinds of locale files:

1. Language
2. Territory
3. Character set
4. Linguistic sort

Locale files have this naming convention:

Code:	**Position:**	**Usage:**
lx	1 thru 2	Standard prefix for all locale files
t	3	The locale Type: 0 = Language 1 = Territory 2 = Character set 3 = Linguistic sort
nnnn	4 thru 7	Object ID (in hex)
.nlb	8 thru 11	Standard extension for all locale files

NLS Parameters

You use NLS parameter settings to direct Oracle's globalization support. NLS parameters come in two varieties:

- Server-side - Set them through initialization parameters
- Client-side - Set them through environmental variables or by the **ALTER SESSION** statement

Some NLS parameters can be used on both the server- and client- sides, while others apply only to one side of the picture. All NLS parameters group into the following categories:

- Language and territory
- Date and time
- Calendar parameters
- Numeric, list, and monetary parameters
- Length semantics

Language and Territory Parameters – Set environment variable **NLS_LANG** on the client-side to determine language, territory, and character set for the client. In most cases, it supersedes the need to individually set the more specific NLS parameters **NLS_LANGUAGE**, **NLS_TERRITORY**, and **NLS_CHARACTERSET**. The format for **NLS_LANG** is:

NLS_LANG = language_territory.characterset

Here is an example for the USA:

NLS_LANG = AMERICAN_AMERICA.US7ASCII

You can leave out individual components for **NLS_LANG** (none are required), but if you do, ensure that the territory is preceded by the underscore (_) and the character set is preceded by a period (**.**). For example, this line specifies only the territory:

NLS_LANG = _AMERICA

The language part of the specification directs Oracle conventions for messages, sorting, and the day and month names. It defaults to **AMERICAN**.

The territory part of the specification determines the conventions for default date, monetary, and numeric formats. If not defined, the default territory value for the language is used.

The character set determines the client character set. This would typically match the character set used by the client PC and its operating system.

Date and Time Parameters – Define NLS date and time functionality through 4 parms:

Parameter:	Usage:
NLS_DATE_FORMAT	The default format for dates in the current session only.
NLS_DATE_LANGUAGE	Determines day and month names & abbreviations.
NLS_TIMESTAMP_FORMAT	Sets the default date format for both **TIMESTAMP** and **TIMESTAMP WITH TIME ZONE** datatypes.
NLS_TIMESTAMP_TZ_FORMAT	Sets the default date format for both **TIMESTAMP** and **TIMESTAMP WITH TIME ZONE** datatypes, and adds the option of time zone formatting.

Here are a few example **ALTER SESSION** commands for these parameters:

```
SQL>  alter session set nls_date_format = 'MM/DD/YY' ;
SQL>  alter session set nls_date_language=Spanish ;
SQL>  alter session set nls_timestamp_format =
                        'MM/DD/YYYY HH24:MI:SS.FF';
SQL>  alter session set nls_timestamp_tz_format =
                        'MM/DD/YYYY HH:MI TZH:TZM';
```

Calendar Parameters -- 10g supports 7 international calendars (and their conventions):

Calendar:	Usage:
Gregorian	The standard 365-day calendar used in the USA and world-wide.
Japanese Imperial	Same as Gregorian but starts with the imperial era.
ROC (ROC China)	Same as Gregorian but starts with the ROC founding.
Persian	Same as Gregorian but with different numbers of days per month.
Thai Buddhist	Same as Gregorian but starts with birth of the Buddha.
Arabic Hijrah	12 months with either 354 or 355 days.
English Hijrah	12 months with either 354 or 355 days.

Numeric, List, and Monetary Parameters -- There are 4 key NLS parameters in this category:

NLS Parameter:	Usage:
NLS_NUMERIC_CHARACTERS	Defines separators for thousands and decimal place within numbers. USA example with comma and the period for the decimal place: 1,234.50.
NLS_LIST_SEPARATOR	Defines the separator character for lists of numbers (on the client-side only).
NLS_CURRENCY	Defines the currency symbol. For example, US dollars use the dollar sign symbol ($).
NLS_ISO_CURRENCY	Provides uniqueness to the **NLS_CURRENCY** setting. For example, since several countries use the dollar sign, this would append further information to that symbol, such as **USD** for US dollars or **AUD** for Australian dollars.

Length Semantics Parameters – 10g supports two kinds of length semantics (ways of counting the number of characters in a character string). They are byte semantics and character semantics. Set them through the **NLS_LENGTH_SEMANTICS** parameter:

```
SQL> alter system set nls_length_semantics = char ;
```

Prioritization of NLS Parameters – Since there are several ways to set NLS parameters, Oracle has a precedence order for them. The following table shows that precedence priority:

Priority:	Method:	Scope:
1	Set in SQL functions	Current SQL function
2	Explicit **ALTER SESSION** statement	Current user session
3	Client environmental variable	Current user session
4	Set of server initialization parameter	Instance-wide
5	Default	Instance-wide

You can get information on the NLS settings from these NLS views:

View:	Contains:
NLS_SESSION_PARAMETERS	Current NLS settings for your session
NLS_INSTANCE_PARAMETERS	Instance-wide NLS settings
NLS_DATABASE_PARAMETERS	NLS settings for the database (its default values)
V$NLS_VALID_VALUES	Lists all the valid values for **NLS_SORT**, **NLS_LANGUAGE**, **NLS_TERRITORY**, and **NLS_CHARACTERSET**.

Unicode

Unicode is a character set that supports all human languages. It assigns a guaranteed-unique value called a code point to each character so that no conflicts exist. While Unicode supports all human languages, it costs performance and storage overhead.

Oracle databases have two session-independent character sets when created:

- Database Character Set - Defines text storage (for CHAR, VARCHAR2)
- National Character Set - Defines the alternate Unicode character set for datatypes like NCHAR, NVARCHAR2, NCLOB

Oracle supports 3 Unicode encoding methods:

Unicode Method:	Usage:
UTF-8	8-bit encoding that uses from 1 to 4 bytes per character. Thus, this is a variable-length encoding method.
UCS-2	Fixed-width 16-bit encoding method that stores each character in 2 bytes.
UTF-16	A super-set of UCS-2 that also supports supplemental characters.

Datetime Datatypes

Oracle supports 4 datetime datatypes:

Datatype:	Meaning:
DATE	Date (with Time). Has 7 parts: century, year, month, day, hours, minutes, and seconds (since midnight).
TIMESTAMP	Like **DATE** but adds fractional seconds (by default to 6 digits of precision).
TIMESTAMP WITH TIME ZONE	Extends **TIMESTAMP** to include time zone, an offset in hours and minutes between local time and UTC (Coordinated Universal Time).
TIMESTAMP WITH LOCAL TIME ZONE	Synchronizes all time elements to local time. Allows an organization to sync times across the multiple time zones they operate in.

Linguistic Sorting and Searching

Binary sorting is a numeric sort based on the encoded value in the character set. It performs well and is 10g's default sorting method. Linguistic sorting imposes the rules unique to various languages and cultures to sorting and searching.

Linguistic sorts may deal only with one language (be monolingual), or deal with more than one language (be multilingual). Multilingual means Oracle can apply the rules of multiple languages in one sort operation.

Use the **NLS_SORT** parameter to dictate which sort (binary or one of the linguistic ones) to use. Just set it to the name of any valid linguistic sort definition. This example sets it to the rules of the Spanish language:

```
SQL> alter session set nls_sort = Spanish ;
```

NLS_SORT defaults to the sort method associated by default with the **NLS_LANGUAGE** setting.
For case-insensitive sorts, append the letters **_CI** to the sort name.

For accent-insensitive sorts, append the letters **_AI** to the sort name.

This example dictates a case-insensitive sort for Spanish by appending the letters **_CI** to the name of the linguistic definition:

```
SQL> alter session set nls_sort = Spanish_CI ;
```

NLS_COMP can be set to either **BINARY** (the default) or **ANSI**. It applies only to these operations:

- ORDER BY
- BETWEEN
- CASE WHEN
- HAVING

- IN/OUT
- START WITH
- WHERE

Setting **NLS_COMP** makes it easier to control linguistic sorting because its value only applies to the above operations.

Securing the Listener

With many database servers accessible through the internet, Oracle Corp. increasingly emphasizes security. This Exam covers how to secure the listener.

The listener can be managed remotely by accessing it through its port. This is convenient for administrators but means a hacker could compromise an Oracle database server through the listener. Not only could they stop the listener or change its password (locking out the legitimate administrators), they could even set up external process definitions that would allow them to execute programs through the listener's external procedure server. Let's discuss several steps in securing the listener.

Listener Utility Password

The first step in securing the listener is to assign an administrative password to the listener control utility. Recall from the "Administrator I" OCA test, you can configure the listener – and its password – through several means:

- The **lsnrctl** utility
- Net Manager GUI (**netmgr**)
- Network Creation Assistant GUI (**netca**)
- Enterprise Manager (EM) GUI

For example, in using EM, go to the initial **Database Control** screen, then select the listener to configure down in the **General** section of the screen. In the resulting **Edit Listener** screen, you can pick **Authentication** and assign a new password to the listener.

From the **lsnrctl** utility, use the **CHANGE_PASSWORD** and **SAVE_CONFIG** sub-commands to enter and save a new password for the listener. The **SET_PASSWORD** command allows you to enter the proper password to use the administrative functions when communicating with the utility.

Listener Logging

You should check the listener log for penetration attack attempts. This would typically show up as a large number of failed connection attempts (TNS-01169 errors). 3 listener configuration parameters are key:

- **LOG_DIRECTORY** - Directory where the listener log resides
- **LOG_FILE** - Name of the listener log file
- **LOG_STATUS** - Whether to enable the listener log (default is **ON**)

You can view the listener log contents either through the Enterprise Manager panels or simply by accessing the log file directly with any editor.

Valid Node Checking

Another part of listener security is valid node checking. This allows you to ensure that only communications from specified nodes are accepted by Oracle.

Set valid node checking on by manually editing the **sqlnet.ora** file. By default, this file resides in **$ORACLE_HOME/network/admin**, or it can be assigned a different directory location by setting environmental variable **TNS_ADMIN**. Here are the relevant parameters to set in this file:

SQLNET.ORA Parm:	Usage:
TCP.VALIDNODE_CHECKING	Set to **YES** to enable valid node checking.
TCP.INVITED_NODES	Addresses from which requests are accepted.
TCP.EXCLUDED_NODES	Addresses from which requests are rejected.

For **TCP.INVITED_NODES** and **TCP.EXCLUDED_NODES**, specify the addresses either as explicit IP addresses or as host names. If using host names, SQL*Net must have some way to resolve them (such as DNS, a hosts file, or whatever). **TCP.INVITED_NODES** and **TCP.EXCLUDED_NODES** are mutually exclusive; use one or the other but not both.

Remove External Procedure Services

Listeners can run binary programs residing on the server through the feature called external procedure support. These would typically be shared library files under Unix and Linux or dynamic link library (*.DLL) files under Windows. These execute under the oracle process owner (usually the user id **oracle**), so a malicious person could potentially do real damage running them.

If you do not need external procedure support, disable it by editing the **listener.ora** file. Simply remove the two sections from the file that refer to **PLSExtProc** and **extproc**. An example is shown on the next page:

Remove the italicized, boldfaced parts from this **listener.ora** file:

```
SID_LIST_LISTENER =
        (SID_LIST =
                (SID_DESCRIPTION =
                        (SID_NAME = PLSExtProc)
                        (ORACLE_HOME = /oracle/prod/apps)
                        (PROGRAM = extproc)
                )
                (SID_DESCRIPTION =
                        . . . etc . . .
                )
        )
LISTENER =
        (DESCRIPTION_LIST =
                (DESCRIPTION =
                        (ADDRESS_LIST =
                                (ADDRESS = (PROT=ICP)(KEY = EXTPROC))
                        )
                        (ADDRESS_LIST =
                                . . . etc . . .
                        )
                )
        )
```

Practice Questions

Chapter 1 Using Globalization Support Objectives

1. Place the following steps in the correct sequence for setting NLS parameters.

   1. Explicit ALTER SESSION statement
   2. Explicitly set in SQL functions
   3. Set in server initialization parameter file
   4. Default
   5. Client environment variable (implicit ALTER SESSION statement)

   Select the best answer.

   - ❍ A. 1,2,5,3,4
   - ❍ B. 2,1,5,4,3
   - ❍ C. 2,1,5,3,4
   - ❍ D. 2,5,1,3,4

Chapter 2 Securing the Oracle Listener

1. Which of the following answer choices are good ways to secure the listener?

   Select the three best answers.

   - ❑ A. Manually remove the external procedure services from the listener.ora file if they are not used.
   - ❑ B. Enable listener logging.
   - ❑ C. Implement valid node checking in the listener.ora file.
   - ❑ D. Set the SQLNET password by making the appropriate changes to the sqlnet.ora file.
   - ❑ E. Set the listener password.
   - ❑ F. Set the initialization parameter PASSWORDS_LISTENER in the SPFILE or PFILE for the database.

Chapter 3 Configuring Recovery Manager

1. Which of the following commands connect the RMAN utility to a target database but not to an RMAN catalog database?

   Select the three best answers.

   - ❑ A. c:> rman target /
   - ❑ B. c:> rman target / nocatalog
   - ❑ C. c:> rman target / catalog rman_user/rman_user@orarc
   - ❑ D. RMAN> connect target
   - ❑ E. RMAN> connect catalog rman_user/rman_user@orarc
   - ❑ F. c:> rman

2. Which of the following answer choices accurately describe how to implement parallelism for disk backups using RMAN?

   Select the two best answers.

   - ❑ A. Take no action. RMAN automatically implements parallelism for disk backups whenever possible.
   - ❑ B. Simply refer to multiple backup sets within the RMAN backup script.
   - ❑ C. Implement parallelism manually by allocating two or more channels for multiple backup sets within a single backup script.
   - ❑ D. RMAN does not support parallelism in backups. If you require parallelism you must employ user-defined backup methods.
   - ❑ E. Configure parallelism for a specific backup device type using a command such as this: RMAN> configure device type disk parallelism 3.

Chapter 4 Recovering from User Errors

1. Which of the following statements are true about Flashback Drop?

   Select the two best answers.

   - ❑ A. Flashback Drop relies on Undo tablespace records and the logs to recover a dropped table.
   - ❑ B. You use flashback drop to restore a dropped table with a statement like this: SQL> FLASHBACK DROP TABLE my_table TO BEFORE DROP.
   - ❑ C. Flashback Drop is faster than alternative recovery procedures and does not impact other users when recovering a dropped table.
   - ❑ D. You can query the Recycle Bin and find the table you accidentally dropped by its name.
   - ❑ E. Flashback Drop enables you to recover partitioned index-organized tables.
   - ❑ F. Whether you can recover a table using Flashback Drop depends on the size of the Recycle Bin and its internal space management.

Chapter 5 Dealing with Database Corruption

1. Which of the following statements are true concerning Block Media Recovery?

   Select the three best answers.

   - ❑ A. Block Media Recovery is an RMAN feature and can only be used with RMAN.
   - ❑ B. You can use either an RMAN full or incremental backup in order to perform Block Media Recovery.
   - ❑ C. You must take the datafile offline while performing Block Media Recovery.
   - ❑ D. Block Media Recovery can be used for incomplete recovery of damaged data blocks.
   - ❑ E. You issue RMAN's BLOCKRECOVER command in order to perform Block Media Recovery.
   - ❑ F. Block Media Recovery can sometimes be performed without redo logs.

2. Which of the following statements describe valid ways to detect and fix data block corruption or other data integrity problems within datafiles?

   Select the two best answers.

   - ❑ A. Use the ANALYZE command to detect and automatically repair corrupt data blocks or damaged datafiles.
   - ❑ B. Use the DBVERIFY utility to detect and automatically repair corrupt data blocks.
   - ❑ C. Use the DBMS_REPAIR package procedures to identify and fix corrupt blocks.
   - ❑ D. Identify corrupt data blocks through the Alert log or trace files, then use Block Media Recovery to recover them.
   - ❑ E. Run the ANALYZE ... VALIDATE STRUCTURE command with the FIX option.

Chapter 6 Automatic Database Management

1. Which background process flushes the ASH buffer if it fills up in less than 30 minutes?

   Select the best answer.

   - ❍ A. PMON
   - ❍ B. SMON
   - ❍ C. MMON
   - ❍ D. MMNL
   - ❍ E. SGA
   - ❍ F. DBWn

2. Which of the following statements are true concerning the DB_TIME statistic?

   Select the three best answers.

   - ❑ A. DB_TIME captures total time spent in database calls for all components (an aggregation of CPU and non-idle wait time).
   - ❑ B. DB_TIME is used by Automatic Storage Manager (ASM) to configure disk access rates.
   - ❑ C. Because DB_TIME is common across database components, it is useful in improving performance because the goal can be to reduce DB_TIME.
   - ❑ D. To improve performance, you run StatsPack, which then produces the DB_TIME statistics you use to improve database performance.
   - ❑ E. DB_TIME is collected cumulatively from instance startup time.

Chapter 7 Recovering from Non-Critical Losses

1. Which of the following answer choices is not a non-critical recovery?
   Select the best answer.

   - ❍ A. Recreating a redo log file
   - ❍ B. Recreating an index
   - ❍ C. Recreating the password file
   - ❍ D. Recovering a read-only tablespace
   - ❍ E. Recovering a datafile in the SYSTEM tablespace
   - ❍ F. Recovering a missing tempfile for the temporary tablespace

Chapter 8 Monitoring and Managing Storage

1. You run the following two statements on the large EMPLOYEES table:
   SQL> alter table prod.employees enable row movement ;
   SQL> alter table prod.employees shrink space ;
   Which of the following statements is not true?
   Select the best answer.

   - ❍ A. Space below the High Water Mark is freed up.
   - ❍ B. Full table scans on the table will now take less time.
   - ❍ C. All chained rows are fixed.
   - ❍ D. The High Water Mark is moved down.
   - ❍ E. Data blocks are now contiguous, and fragmented wasted space is reclaimed.

2. Which of the following answer choices store index and table data together?
   Select the best answer.

   - ❍ A. IOT mapping tables
   - ❍ B. Index clusters
   - ❍ C. Hash clusters
   - ❍ D. Sorted hash clusters
   - ❍ E. Index-organized tables (IOTs)
   - ❍ F. IOT overflow area

Chapter 9 Automatic Storage Management

1. Which statement about Automatic Storage Management (ASM) is not true?

   Select the best answer.

   - ❍ A. You can configure ASM storage when creating a database through the Oracle Universal Installer (OUI).
   - ❍ B. New background process RBAL coordinates the disk activity for disk groups in an ASM instance.
   - ❍ C. New background process RBAL performs the open and close of disks in the disk groups when running on the ASM instance.
   - ❍ D. ASM supports Real Application Clusters (RAC).
   - ❍ E. For ASM instances, the STARTUP command defaults to STARTUP MOUNT. There is no control file, database, or data dictionary to mount, and the ASM disk groups are mounted instead of a database.
   - ❍ F. Templates enable ASM to deal with different file types.

2. Which ordering of the steps below will successfully migrate an existing database to automatic storage management (ASM)?

   1. Delete or archive the source database files

   2. Shut down the source database

   3. Edit the PFILE to use Oracle Managed Files (OMF) for files and remove the CONTROL_FILES parameter

   4. Make a full cold backup of the source database

   5. Run an RMAN script that moves database objects from non-ASM locations into an ASM disk group

   6. Start up and test the database using ASM

   Select the best answer.

   - ❍ A. 2,4,3,5,6,1
   - ❍ B. 1,2,3,4,5,6
   - ❍ C. 4,2,3,5,1,6
   - ❍ D. 2,4,3,5,1,6

Chapter 10 Monitoring and Managing Memory

1. Which of the following statements describe ways in which you can increase the size of the Active Session History (ASH) memory buffer?

   Select the two best answers.

   - ❑ A. Increase the size of the Log Buffer.
   - ❑ B. Increase the number of CPUs.
   - ❑ C. Increase the size of the Shared Pool.
   - ❑ D. Increase the size of the Automatic Workload Repository (AWR).
   - ❑ E. You cannot increase the size of the ASH buffer.

Chapter 11 Database Recovery

1. Which of the following options are valid Oracle backup methods?

   Select the three best answers.

   - ❑ A. User-managed
   - ❑ B. Unload utility
   - ❑ C. Database COPY_FILE procedure
   - ❑ D. RMAN
   - ❑ E. EXPORT utility
   - ❑ F. Database PUT_FILE procedure

2. Put the following steps in the correct order that will enable ARCHIVELOG mode for a non-archivelog mode database.

   1. Startup Mount the database
   2. SQL> alter database open;
   3. Set log_archive_start=true and other initialization parameters
   4. Shutdown the database
   5. SQL> alter database archivelog ;

   Select the best answer.

   - ❍ A. 4,3,1,2,5
   - ❍ B. 4,3,5,2,1
   - ❍ C. 2,5,1,3,4
   - ❍ D. 4,3,1,5,2

3. What happens when a database goes down when one or more tablespaces that contain user data are in backup mode?

   Select the best answer.

   - ❍ A. The datafile(s) are not checkpointed and are not consistent with the rest of the database. When you restart the database, it will report the datafile(s) as needing recovery.
   - ❍ B. Oracle automatically checkpoints and resynchronizes the datafile(s) left in backup mode during crash recovery the next time you start up the database.
   - ❍ C. The Oracle database will not start.
   - ❍ D. The database starts up ok and automatically flips the affected datafile(s) into read-only mode. If you want to update them, you have to issue ALTER DATAFILE 'datafile' READ WRITE for each datafile.
   - ❍ E. The database will start up ok but mark the affected datafile(s) offline and needing recovery. Your only option is always to perform media recovery on the affected datafile(s) in order to get them back online.

4. When performing incomplete media recovery using RMAN, which of the following keywords are not supported?

   Select the three best answers.

   - ❑ A. UNTIL TIME
   - ❑ B. UNTIL SEQUENCE
   - ❑ C. UNTIL SCN
   - ❑ D. UNTIL CHANGE
   - ❑ E. UNTIL CANCEL
   - ❑ F. UNTIL COMPLETE

5. What does "incomplete recovery" mean?

   Select the best answer.

   - ❍ A. A recovery that restores only certain datafiles in the database.
   - ❍ B. A recovery that does not work off of a whole database backup.
   - ❍ C. A recovery that stops at some point in time prior to the failure that forced the recovery.
   - ❍ D. A recovery of only part of the database.
   - ❍ E. A recovery process that has not fully been completed, because other database activities must be completed first.

Chapter 12 Flashback Database

1. What state must the database be in to configure flashback database?

   Select the best answer.

   - ❍ A. NOMOUNT
   - ❍ B. MOUNT
   - ❍ C. OPEN
   - ❍ D. SHUTDOWN

2. Which of the following statements about the new flashback database feature are true?

   Select the two best answers.

   - ❍ A. Flashback database allows you to "flash back" selected tablespaces within the database to previous points in time that you specify.
   - ❍ B. You can inspect view V$DATABASE to see if flashback database is enabled.
   - ❍ C. When flashback database is enabled, the new background process RVWR runs and writes changed data from the flashback buffer in the SGA to the flashback logs in the flash recovery area.
   - ❍ D. The new initialization parameter DB_FLASHBACK_RETENTION_TARGET should be set to the number of seconds you want to be able to flash back the database into the past.
   - ❍ E. When you flashback a database to a prior point in time, Oracle uses the Undo tablespace data to go back in time, then the Redo logs roll forward in time to a consistent transactional state.

Chapter 13 Managing Resources

1. Place the following activities in the correct sequence for working with the Database Resource Manager (DRM).

   1. validate_pending_area

   2. create consumer resource groups, resource plans, and resource plan directives in the pending area

   3. create_pending_area

   4. submit_pending_area

   Select the best answer.

   - ❍ A. 3,2,1,4
   - ❍ B. 2,3,1,4
   - ❍ C. 1,2,3,4
   - ❍ D. 4,3,2,1

2. Which statement is not true concerning Resource Plans?

   Select the best answer.

   - ❍ A. Simple resource plans have no sub-plans.
   - ❍ B. Single-level plans may only use the CPU allocation method called RATIO.
   - ❍ C. Resource plans prioritize resource allocation through the use of levels, with level 1 being the highest priority and 8 being the lowest.
   - ❍ D. The only consumer groups assigned to a simple resource plan are those you assign to it.
   - ❍ E. You create a sub-plan in the exact same manner as a resource plan.

Chapter 14 Automatic Tasks with the Scheduler

1. Which statement about the Job Scheduler is not true?

   Select the best answer.

   - ❍ A. The Job Coordinator is a background process that ensures all jobs are run on schedule.
   - ❍ B. Job slave processes carry out the execution of tasks assigned to them by the Job Scheduler.
   - ❍ C. For Real Application Cluster (RAC) systems, there is one job coordinator for all instances in the cluster.
   - ❍ D. The Job Table is the master container for all enabled jobs in the database.
   - ❍ E. The new 10g job scheduler supersedes the old DBMS_JOB package, but DBMS_JOB is still available in 10g.

Answers and Explanations

Chapter 1

1. Answer: C

Explanation A. This is not a correct answer. Setting NLS in SQL functions (2) is of higher precedence than an explicit ALTER SESSION statement (1). So, the correct order is 2, 1, 5, 3, 4.

Explanation B. This is not a correct answer. The default setting for the instance (4) is last in order of precedence. So, the correct order is 2, 1, 5, 3, 4.

Explanation C. This is the correct answer. It shows the proper precedence order for setting the NLS parameters.

Explanation D. This is not a correct answer. In the correct precedence order, an explicit ALTER SESSION statement (1) outranks an implicit ALTER SESSION statement (5), as set by a client environmental variable.

Chapter 2

1. Answers: A, B, E

Explanation A. This is a correct answer. If you do not use external procedure services, you can manually remove them from the listener.ora file. This plugs a potential security hole.

Explanation B. This is a correct answer. By ensuring the listener activities are logged, you can inspect this file and its entries to see if any penetration attempts have been made on the listener. Turning on listener logging increases security for the listener.

Explanation C. This is not a correct answer. You implement valid node checking by making changes to the sqlnet.ora file, not to the listener.ora file.

Explanation D. This is not a correct answer. You can set a password for the listener, but you do not set passwords in the sqlnet.ora file.

Explanation E. This is a correct answer. You can set the listener password for greater security on the listener. Use listener commands such as SET PASSWORD, CHANGE_PASSWORD, and SAVE_CONFIG to do this.

Explanation F. This is not a correct answer. You set the parameter PASSWORDS_LISTENER in the listener.ora file, not in the database initialization file (the SPFILE or PFILE).

Chapter 3

1. Answers: A, B, D

Explanation A. This is a correct answer. Entering this command to the operating system's command line prompt starts up the RMAN utility and connects to the local target database. It does not connect to an RMAN catalog because if you do not explicitly specify an RMAN catalog to use, nocatalog is the default.

Explanation B. This is a correct answer. Entering this command to the operating system's prompt starts the RMAN utility and connects to a target database. Since you have specified nocatalog, the command does not connect to an RMAN catalog database.

Explanation C. This is not a correct answer. The command starts the RMAN utility and connects to a local target database, but the catalog keyword and its operands specify that the command also connects to an RMAN catalog database.

Explanation D. This is a correct answer. Here, the user is already inside the RMAN utility, as shown by the prompt RMAN>. The connect target command connects the user to the local target database. Since no RMAN catalog is specified, the command does not also connect to an RMAN target database.

Explanation E. This is incorrect because the catalog keyword specifies an RMAN catalog database to connect to. The user is already inside the RMAN utility and connects to a catalog database by issuing this command.

Explanation F. This is not a correct answer. Entering the rman command at the operating system's command prompt starts the RMAN utility but connects to neither a target database nor a catalog database.

2. Answers: C, E

Explanation A. This is not a correct answer. By default, RMAN does not configure disk backup parallelism. If you issue an RMAN show all command, you will see this line among the defaults: CONFIGURE DEVICE TYPE DISK PARALLELISM 1 BACKUP TYPE TO BACKUPSET;

Explanation B. This is not a correct answer. You need to allocate multiple channels to implement parallelism. Simply naming multiple backup sets with the RMAN script does not achieve this. By default, RMAN does not configure disk backup parallelism.

Explanation C. This is a correct answer. You can implement parallelism manually from within your backup scripts. To do this, allocate multiple channels. Typically you allocate one channel for each backup set.

Explanation D. This is not a correct answer. RMAN supports parallelism in backups. Use the CONFIGURE . . . PARALLELISM command to establish parallelism in disk backups by default. You do not have to employ user-defined backup methods to achieve parallelism in backups.

Explanation E. This is a correct answer. You can use RMAN's CONFIGURE command to set the PARALLELISM parameter to a value greater than 1. This automatically implements parallelism when backups use that device type.

Chapter 4

1. Answers: C, F

Explanation A. This is not a correct answer. Flashback Drop relies on the Recycle Bin to recover dropped tables. The Recycle Bin is a new feature of 10g that is enabled by default and specifically designed to support flashback technology.

Explanation B. This is not a correct answer. You run Flashback Drop by the command FLASHBACK TABLE (not FLASHBACK DROP TABLE). The correct statement is:
SQL> FLASHBACK TABLE my_table TO BEFORE DROP ;

Explanation C. This is a correct answer. These are two of the major advantages to Flashback Drop over traditional means of table recovery. Besides its convenience, another benefit is that it is so easy to use, the average developer or end-user can recover a dropped table with no aid from database administrators.

Explanation D. This is not a correct answer. Tables in the Recycle Bin have system-assigned names. Their names consist of a 24-character unique global ID, plus a version number assigned by the database. Your table will not appear in the Recycle Bin under its original name.

Explanation E. This is not a correct answer. One of the limitations of the new Flashback Drop feature is that it cannot recover partitioned index-organized tables.

Explanation F. This is a correct answer. The size of the Recycle Bin and how it reuses its space affects whether a particular table is still available for recovery through Flashback Drop. You can manually clear space from the Recycle Bin (to keep other objects in there longer) through such commands as PURGE TABLE, PURGE TABLESPACE, and PURGE RECYCLEBIN.

Chapter 5

1. Answers: A, E, F

Explanation A. This is not a correct answer. Block Media Recovery (BMR) is a new Recovery Manager (RMAN) feature by which you use RMAN to recover individual corrupted data blocks. Therefore, you can only use BMR if you use RMAN.

Explanation B. This is not a correct answer. You must have a full RMAN backup in order to perform Block Media Recovery (BMR). An RMAN incremental backup is not usable in this role because it backs up only changed blocks. BMR requires a full backup.

Explanation C. This is not a correct answer. One of the big advantages to Block Media Recovery is that you can have the datafile online and usable to customers while you repair individual corrupt blocks. Of course, the bad blocks are not available to users until you have completed the Block Media Recovery.

Explanation D. This is not a correct answer. You must perform full media recovery of invalid data blocks. In other words, you are required to recover to currency. You cannot recover a block merely to some prior point in time.

Explanation E. This is a correct answer. You would recover a block using a command such as this:
RMAN> blockrecover datafile 4 block 5 ;

Explanation F. This is a correct answer. If the data blocks you recover using Block Media Recovery are not affected by the redo logs, then the redo logs are not required or applied during data block recovery.

2. Answers: C, D

Explanation A. This is not a correct answer. You use the ANALYZE command to validate the integrity of the structure of the object it analyzes. The command does not correct any problems. You must fix the problem through some follow-on step. For example, you could run ANALYZE to determine when an object lacks structural integrity, then drop and re-build the object to fix it.

Explanation B. This is not a correct answer. You use the DBVERIFY utility to verify the physical structure of datafiles and detect bad data blocks. The utility does not correct the corruptions. You must fix the problem by some follow-on step. For example, you might fix problems identified by DBVERIFY by dropping and recreating objects, or performing media recovery.

Explanation C. This is a correct answer. The package DBMS_REPAIR includes about a half dozen procedures that enable you to detect and fix corrupt blocks in tables and indexes. Using these procedures is a somewhat complicated, multi-step process.

Explanation D. This is a correct answer. One way to identify and correct corrupt data blocks is to identify any problems through the Alert log and trace files. Then use Recovery Manager's Block Media Recovery (BMR) feature to recover and fix the bad blocks.

Explanation E. This is not a correct answer. The ANALYZE...VALIDATE STRUCTURE command validates the integrity of the structure of the object on which you run it. But the command cannot fix any of the corruption it finds, and there is therefore no FIX option.

Chapter 6

1. Answer: D

Explanation A. This is not a correct answer. The Process Monitor (PMON) cleans up after failed database connections. It is not involved in managing the ASH buffer.

Explanation B. This is not a correct answer. The System Monitor (SMON) is not involved in the management of the ASH buffer. SMON's three basic functions are to (1) perform instance recovery following an instance crash (2) coalesce free space in the database (3) manage space used for sorting.

Explanation C. This is not a correct answer. Memory Monitor (MMON) gathers and analyzes statistics used by the Automatic Workload Repository manager and it flushes the ASH buffer every 30 minutes. But if the ASH buffer fills before 30 minutes have elapsed, MMNL flushes that buffer.

Explanation D. This is the correct answer. Memory Monitor Light (MMNL) gathers and analyzes statistics used by the Automatic Workload Repository manager. MMON flushes the ASH buffer every 30 minutes, but if the ASH buffer fills before that time, MMNL flushes that buffer.

Explanation E. This is not a correct answer. The System Global Area or SGA is not an Oracle background process. It is the main memory block used by Oracle processes. Oracle allocates SGA memory for the Shared Pool, the Database Buffer Cache, the Redo Log Buffer, and potentially other memory areas. Since the SGA is not a background process, it does not flush the ASH buffer.

Explanation F. This is not a correct answer. The Database Writer (DBWn) writes modified database blocks from the SGA's Database Buffer Cache to the datafiles on disk. It is not involved in flushing the ASH buffer.

2. Answers: A, C, E

Explanation A. This is a correct answer. DB_TIME is the single most important ADDM statistic, representing the total time spent in database calls for all components.

Explanation B. This is not a correct answer. DB_TIME is the single most important statistic collected by the Automatic Database Diagnostic Monitor (ADDM), and is used in improving database performance. DB_TIME has nothing to do with ASM or disk management.

Explanation C. This is a correct answer. DB_TIME is a statistic that is common across database components. Improving database performance can therefore be viewed as the goal of reducing DB_TIME.

Explanation D. This is not a correct answer. DB_TIME is produced in 10g by the Automatic Database Diagnostic Monitor (ADDM) tool. Oracle 10g's ADDM statistics supersede those of StatsPack, which has been obsoleted by the new 10g tools.

Explanation E. This is a correct answer. DB_TIME is an aggregate statistic that is collected cumulatively from instance startup.

Chapter 7

1. Answer: E

Explanation A. This is not a correct answer. Recreating a redo log file is a non-critical recovery. You can usually accomplish it with the database "up." You take advantage of the multiplexing (mirroring) of the online redo log files to recreate a missing or damaged redo log file. A database can run as long as any one copy of the active redo log files is available.

Explanation B. This is not a correct answer. Recreating an index is not a critical recovery situation because the database remains "up" and available while you recreate the index. In most cases, you recover an index simply by dropping and rebuilding it.

Explanation C. This is not a correct answer. Recreating a password file is a non-critical recovery. Simply copy in the password file from a backup you made, or recreate it by issuing the Oracle line command orapwd.

Explanation D. This is not a correct answer. Recovering a read-only tablespace is a non-critical recovery scenario because only the damaged tablespace is affected. Simply restore the read-only tablespace from backup. There is no need to "recover" it (by applying logs) because the tablespace will not have changed since it was backed up, as it is read-only.

Explanation E. This is the correct answer. Recovering a datafile in the SYSTEM tablespace is a critical recovery scenario because Oracle requires that the SYSTEM tablespace be available and cannot function without it. You cannot perform this recovery with the database "up" and open; you must perform it with the database shut down.

Explanation F. This is not a correct answer. This is a non-critical recovery situation because you can simply create a new temporary tablespace and switch the system to using it.

Chapter 8

1. Answer: C

Explanation A. This is not a correct answer. The statement is true. Segment shrink frees up space below the High Water Mark (HWM) and then moves the HWM down.

Explanation B. This is not a correct answer. The statement is true. Segment shrink enables full table scans to run faster. The full scans run faster because any unused space intermingled with the table data has been removed. Data is now in fewer, contiguous blocks for faster scans.

Explanation C. This is the correct answer. The statement is false. The segment shrink feature does not fix chained rows. You will still have chained rows after running segment shrink.

Explanation D. This is not a correct answer. The statement is true. Segment shrink frees up space below the High Water Mark (HWM) and then moves the HWM down.

Explanation E. This is not a correct answer. The statement is true. Segment shrink eliminates internal unused space, and results in contiguous data blocks.

2. Answer: E

Explanation A. This is not a correct answer. IOT mapping tables are used to create bitmap indexes on Index-Organized Tables (IOTs). They are not a way to store index and table data together.

Explanation B. This is not a correct answer. Index clusters are a clustering technology that uses indexes to find rows in clusters. Clustering is not used to store index and table data together.

Explanation C. This is not a correct answer. Hash Clusters are a clustering technology that uses hash functions to determine the physical location where a given row is stored. Hash clustering is not used to store index and table data together.

Explanation D. This is not a correct answer. Sorted Hash Clusters are a special case of Hash Clusters. Sorted Hash Clusters maintain a sort order for rows that are retrieved by the same cluster key. Clustering is not used to store index and table data together.

Explanation E. This is the correct answer. The purpose of an index-organized table (IOT) is to store index and table data together. This can reduce I/O for situations where queries retrieve data mostly through the primary key (such as look-up tables, for example).

Explanation F. This is not a correct answer. The IOT overflow area is used with index-organized tables (or IOTs) when an IOT row's data exceeds the space available for it within a block. Index-organized tables are the storage technology used to store index and table data together. The IOT overflow area is an auxiliary implementation technique for IOTs.

Chapter 9

1. Answer: C

Explanation A. This is not a correct answer. The statement is true. The Oracle Universal Installer (OUI) allows you to specify and configure ASM storage for a new database you create while using OUI. This work occurs within OUI as it invokes the Database Configuration Assistant (DBCA).

Explanation B. This is not a correct answer. The statement is true. The new background process RBAL runs on both the ASM instance and database instances. On the ASM instance, its function is to coordinate disk-balancing activity. New background process ORBn on the ASM instance actually performs the re-balancing of data on disks.

Explanation C. This is the correct answer. This statement is false. The new background process RBAL runs on both the ASM instance and database instances. On the ASM instance, its function is to coordinate disk-balancing activity. On the database instance, it performs the open and close of disks in the disk groups. RBAL does not open and close disks in disk groups when running on the ASM instance.

Explanation D. This is not a correct answer. The statement is true. Real Application Clusters, or RAC, is Oracle's clustering feature. It allows multiple instances on multiple machines to run against a single database. ASM supports RAC. ASM also supports systems that do not use RAC.

Explanation E. This is not a correct answer. The statement is true. ASM instances do not have control files, databases, or data dictionaries. They do have disk groups, which the STARTUP MOUNT command mounts.

Explanation F. This is not a correct answer. The statement is true. ASM deals with multiple kinds of files through its use of templates. These templates give ASM the information it needs to deal with the different types of files.

2. Answer: A

Explanation A. This is the correct answer. Shutdown and back up the source database, then edit the PFILE to alter as required for ASM. Next, run an RMAN script to move database objects into an ASM disk group. Verify the new ASM database. Then, at the end of the process you can delete or archive the original database files.

Explanation B. This is not a correct answer. This sequence is incorrect because it starts out by deleting the original source database files (step 1). You do not want to delete them until after you have successfully copied them over to ASM and verified that the new database works.

Explanation C. This is not a correct answer. This sequence is incorrect because it tries to perform a cold backup of the source database (step 4) before shutting down that database (step 2). You have to shutdown the source database before you can make a cold backup of it.

Explanation D. This is not a correct answer. This sequence is incorrect because it deletes or archives the source database files prior to ensuring the new ASM database works. In other words, step (1) should be preceded by step (6). Even though you have backed up the source database files in step (4), you ideally should not delete the source database files until the end of the process.

Chapter 10

1. Answers: B, C

Explanation A. This is not a correct answer. The size of the ASH buffer is unrelated to the size of the log buffer. The size of the ASH buffer can only be altered by increasing the number of CPUs or the size of the Shared Pool.

Explanation B. This is a correct answer. The ASH is calculated by Oracle as the lesser of (1) the total number of CPUs times 2 MB of memory and (2) 5 percent of the Shared Pool size. Therefore, increasing the number of CPUs will increase the size of the ASH buffer.

Explanation C. This is a correct answer. The ASH is calculated by Oracle as the lesser of (1) the total number of CPUs times 2 MB of memory and (2) 5 percent of the Shared Pool size. Therefore, increasing the size of the Shared Pool will increase the size of the ASH buffer.

Explanation D. This is not a correct answer. The AWR is stored on disk, owned by the SYS user, and stored in the SYSAUX tablespace. The ASH is a memory buffer and its size is unrelated to the size of the AWR on disk.

Explanation E. This is not a correct answer. The ASH is calculated by Oracle as the lesser of (1) the total number of CPUs times 2 MB of memory and (2) 5 percent of the Shared Pool size. Therefore, increasing the number of CPUs or the size of the Shared Pool will increase the size of the ASH buffer.

Chapter 11

1. Answers: A, D, E

Explanation A. This is a correct answer. Oracle Corp refers to customized scripts that employ operating system "copy" commands as user-managed backups. Unless you have special requirements or support an older database, Oracle Corp recommends using the Recovery Manager (RMAN) instead of user-managed backups.

Explanation B. This is not a correct answer. Oracle Corp does not support backups using any "unload" utility. Although it might be possible for you to create backups using some sort of unload utility, Oracle Corp does not support this as a valid approach to database backups.

Explanation C. This is not a correct answer. Oracle 10g supports a new COPY_FILE procedure, but the purpose of this procedure is to copy datafiles on a local server. It is not intended as a backup method.

Explanation D. This is a correct answer. RMAN stands for the Recovery Manager, Oracle 10g's bundled and recommended tool for making database backups. RMAN offers many features that automate backups and reduce errors.

Explanation E. This is a correct answer. Users can make "logical backups" by using the Oracle EXPORT utility. EXPORTs have traditionally been used to backup and recover individual tables. Oracle 10g supports several kinds of "flash backup and recovery" that can replace EXPORTs for some uses.

Explanation F. This is not a correct answer. Oracle 10g supports a new PUT_FILE procedure. Its purpose is to make a copy of a local datafile on a remote server. It is not intended as a database backup method.

2. Answer: D

Explanation A. This is not a correct answer. This sequence opens the database prior to entering the alter database archivelog command. You can only enter the alter database archivelog command when the database is in the MOUNT state. Therefore, steps (2) and (5) should be exchanged, and then this series of steps would be correct.

Explanation B. This is not a correct answer. This sequence opens the database prior to entering the alter database archivelog command. Steps (2) and (5) should be exchanged. Also step (1), the STARTUP MOUNT of the database, must occur prior to step (5), the alter database archivelog command.

Explanation C. This is not a correct answer. This sequence places the steps in the reverse of the proper order. Among other problems, step (2) opens the database for general use before it has been properly placed in ARCHIVELOG mode.

Explanation D. This is the correct answer. This is a correct sequence of steps to convert a database to ARCHIVELOG mode. Shutdown the database, set initialization parms for archiving, STARTUP MOUNT the database, alter the database into archivelog mode, and end by opening the database for general use.

3. Answer: A

Explanation A. This is a correct answer. The datafile(s) that are hung in backup mode become out of sync with the rest of the database. You may be able to remedy the situation without recovery simply by flipping the datafile(s) out of backup mode by issuing the command: ALTER DATAFILE 'datafile' END BACKUP;

Explanation B. This is not a correct answer. Oracle will mark and report the datafile(s) left in backup mode as requiring recovery before they can be used. You will have to remedy this either by taking those datafile(s) out of backup mode or by recovering them.

Explanation C. This is not a correct answer. The database starts but marks the datafiles as requiring recovery and leaves them offline. The database will only not start if the datafiles are non-user and are system-critical (such as those in the SYSTEM tablespace).

Explanation D. This is not a correct answer. Since the datafile(s) are not synchronized with the rest of the database, Oracle starts up ok but leaves the affected datafile(s) offline and reports this to you. You must either issue ALTER DATAFILE 'datafile' END BACKUP on each datafile, or recover them.

Explanation E. This is not a correct answer. It is true that the database will start up ok and mark the affected datafile(s) offline and needing recovery. But you may have more options than a required media recovery. You may be able to remedy the situation without recovery simply by flipping the datafile(s) out of backup mode by issuing the command: ALTER DATAFILE 'datafile' END BACKUP;

4. Answers: D, E, F

Explanation A. This is not a correct answer. This is a valid RMAN option for incomplete media recovery. It allows you to recover to just prior to a timestamp you specify.

Explanation B. This is not a correct answer. This is a valid RMAN option for incomplete media recovery. It allows you to recover to just prior to a known redo log sequence number you specify.

Explanation C. This is not a correct option. This is a valid RMAN option for incomplete media recovery. It allows you to recover to just prior to a System Change Number (or SCN) that you specify.

Explanation D. This is a correct answer. UNTIL CHANGE is not a valid RMAN option. It is actually a valid option for user-managed incomplete media recovery. It stops the incomplete recovery at a point just prior to a specified SCN (System Change Number).

Explanation E. This is a correct answer. UNTIL CANCEL is not a valid RMAN option. It is actually a valid option for user-managed incomplete recovery. It stops the incomplete recovery at a point you specify by issuing the CANCEL command. User-managed recovery supports recovery UNTIL CANCEL while RMAN does not support this functionality.

Explanation F. This is a correct answer. UNTIL COMPLETE is not a valid RMAN option. There are no such keywords as UNTIL COMPLETE supporting incomplete media recovery.

5. Answer: C

Explanation A. This is not a correct answer. The amount of datafiles you restore has no bearing on whether the recovery is considered incomplete. Incomplete recovery refers to applying less than all available redo logs, thereby recovering to some point in time prior to the failure that forced the recovery.

Explanation B. This is not a correct answer. The kind of backup you work from (whole, full, incremental, whatever) has no bearing on whether the recovery is considered incomplete. Incomplete recovery refers to applying less than all available redo logs, thereby recovering to some point in time prior to the failure that forced the recovery.

Explanation C. This is a correct answer. Incomplete recovery refers to applying less than all available redo logs, thereby recovering to some point in time prior to the failure that forced the recovery.

Explanation D. This is not a correct answer. The "part" of the database you recover has no bearing on whether the recovery is considered incomplete. Incomplete recovery refers to applying less than all available redo logs, thereby recovering to some point in time prior to the failure that forced the recovery.

Explanation E. This is not a correct answer. "Incomplete recovery" does not refer to what stage you are at in the recovery process. Incomplete recovery refers to applying less than all available redo logs, thereby recovering to some point in time prior to the failure that forced the recovery.

Chapter 12

1. Answer: B

Explanation A. This is not a correct answer. You do not configure flashback database in the NOMOUNT state. The main activity you perform in NOMOUNT state (in terms of this exam) is to create or recreate a control file or a database.

Explanation B. This is the correct answer. The steps to configure flashback database are: STARTUP MOUNT the database; ALTER DB_FLASHBACK_RETENTION_TARGET=minutes; ALTER DATABASE FLASHBACK ON; and then ALTER DATABASE OPEN.

Explanation C. This is not a correct answer. You must be in a MOUNT state to configure flashback database. You also use the MOUNT state for such activities as altering the archivelog mode of the database.

Explanation D. This is not a correct answer. The database must be in MOUNT state to configure flashback database. You cannot perform many activities that change the database when it is shutdown completely, other than change PFILE parameters. To perform most activities that change a database you must be either in the NOMOUNT, MOUNT, or OPEN state.

2. Answers: B, C

Explanation A. This is not a correct answer. The flashback database feature allows you to "flash back" the entire database to a specified prior point in time. You cannot select and flash back just certain individual tablespaces.

Explanation B. This is a correct answer. View V$DATABASE has a new column FLASHBACK_ON you can inspect to see if the flashback database feature is enabled.

Explanation C. This is a correct answer. RVWR is the new background process that the flashback database feature relies on to keep track of data changes. RVWR writes changed data from the flashback buffer in the SGA to the flashback logs. The flashback logs reside in the flash recovery area.

Explanation D. This is a correct answer. You set parameter DB_FLASHBACK_RETENTION_TARGET to tell Oracle how far into the past you want the flashback logs to go. But, its value is set in minutes, not seconds.

Explanation E. This is not a correct answer. Flashback database uses the flashback logs to go back to the specified point in time. It does not use the Undo data for this purpose. Oracle keeps the flashback logs to go back to some point in time. Oracle may then apply Redo logs to go forward to a consistent transactional state.

Chapter 13

1. Answer: A

Explanation A. This is the correct answer. First you create a pending area, then you build DRM objects within it. After this, you would validate the pending area, and submit it.

Explanation B. This is not a correct answer. You always need to create the pending area first (step 3), before you can create DRM objects within it (step 2), or perform the other steps. The correct ordering is 3, 2, 1, 4.

Explanation C. This is not a correct answer. You must create the pending area first (step 3), then create objects for DRM within it (step 2). After this, you validate the pending area (step 1), and if it validates, you submit it (step 4). The correct ordering is 3, 2, 1, 4.

Explanation D. This is not a correct answer. You cannot submit the pending area (step 4) prior to creating it, building objects within it, and verifying those objects. The correct ordering is 3, 2, 1, 4.

2. Answer: D

Explanation A. This is not a correct answer. The statement is true. Simple resource plans are not allowed to have sub-plans, which is why they are called single-level resource plans.

Explanation B. This is not a correct answer. The statement is true. Simple or single-level resource plans can only use the RATIO CPU allocation method.

Explanation C. This is not a correct answer. The statement is true. There are 8 levels of prioritization for resource plans. 1 is the highest level and 8 is the lowest.

Explanation D. This is the correct answer. The statement is false. Oracle automatically adds consumer groups SYS_GROUP (representing the user ids SYS and SYSTEM) and OTHER_GROUPS (which ensures that users who are not assigned to any group active in the resource plan will still have some resources allocated).

Explanation E. This is not a correct answer. The statement is true. A sub-plan and a plan are the same. A plan becomes a sub-plan if a higher-level plan allocates resources to it (through a resource plan directive).

Chapter 14

1. Answer: C

Explanation A. This is not a correct answer. The statement is true. It is the function of the Job Coordinator to ensure that all jobs are run on schedule.

Explanation B. This is not a correct answer. The statement is true. Job slave processes carry out the execution of tasks assigned to them by the Job Scheduler.

Explanation C. This is the correct answer. The statement is false. For RAC systems, each instance has its own job coordinator. The job coordinators communicate with each other as necessary to share information.

Explanation D. This is not a correct answer. The statement is true. The job table stores information about all database jobs, including owner of the job, the objects the job references, and its next run date. The job table is the master container for all enabled jobs in the database.

Explanation E. This is not a correct answer. The statement is true. While Oracle Corp. recommends that you use the new job scheduler, the older DBMS_JOB package is still present and usable in Oracle 10g.