Exam
Manual

LearnSmart

Oracle 10g
(1Z0-042) DBA I

This LearnSmart exam manual breaks down the most important concepts you needed to know in order to pass the Oracle 10g DBA I exam (1Z0-042). By studying this manual, you will become familiar with an array of exam-related topics, including:

- Creating an Oracle Database
- Managing Schema Objects
- Oracle Net Services
- Proactive Maintenance
- 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 1 (1Z0-042) LearnSmart Exam Manual

Product ID: 010295
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 5
What to Know 5
Tips 6
Concepts 8
Metadata 9
Interfaces 10
Oracle Architecture 14
Instance Memory 14
Oracle Background Processes 15
Oracle Database Files 17
Oracle Initialization Parameters 21
Installing Oracle and Creating Databases 24
Installing Oracle 24
Creating Databases 28
Controlling Databases 30
User Connections and Oracle Shared Server 32
Dedicated Versus Shared Connections 32
How Shared Server Works 34
Memory for Shared Server 35
Configuring Shared Server 35
Tuning Shared Server 39
10g Shared Server Improvements 40
Oracle Data Objects 42
Views 42
Tables 43
Constraints 48
Tablespaces 49
Datafiles 53
Segments 54
Extents and Blocks 54
Indexes 55
Sequences 56

Managing Data 57
SELECT Statements 57
INSERT Statements 58
UPDATE and DELETE 58
PL/SQL 58
Data Pump 61
SQL*Loader 64
Managing Users and Security 65
User Accounts 65
Profiles 65
Privileges 66
Roles 67
Implementing Security 68
Auditing 68
Database Consistency and Concurrency 71
Oracle Net Services 72
Oracle Net on the Server 72
Oracle Net on the Client 75
Performance Monitoring and Maintenance 77
Other Features 79
Backups and Recoveries 80
Archive Logging 80
The Flash Recovery Area 80
Kinds of Backups and RMAN's Role 81
Recoveries 82
Recovering from Logical Errors 83
Recovering from Media Errors 84
Closing Remarks 86
Practice Questions 87
Answers and Explanations 95

Abstract

This Exam Manual prepares you for the Oracle 10g DBA certification exam #1Z0-042, "Oracle Database 10g: Administration I." This test is popularly referred to as the Oracle 10g "OCA Test." Passing it awards you the certification title Oracle Certified Associate, or OCA.

Topics on this exam include: installing Oracle 10g software with the Oracle Universal Installer and Optimal Flexible Architecture; creating an Oracle instance and database; controlling the database (starting it, stopping it, using common database interfaces, defining configuration files); the basics of storage (including tablespaces, datafiles, and the physical storage structures on disk); logical database objects (such as tables and views); manipulating data through SQL statements and utilities like Data Pump and SQL*Loader; creating and managing database users; creating tables including datatypes and constraints; Oracle's procedural language superset of SQL, called PL/SQL; database security; and backup and recovery basics.

What to Know

Before we begin, it is very important to note that you as a candidate need to know the basic architecture of Oracle, its components, and their functions. This includes both in-memory components (the instance) and components stored on disk (the database). Understand the basic operations of Oracle in terms of data access, updating, logging, and backup and recovery. Furthermore, it's necessary that you know and/or understand the following:

- Understand how to plan for and install Oracle software. Be able to create a database after installing the software. Be familiar with both command line interfaces into Oracle (such as SQL*Plus and *i*SQL*Plus), as well as graphical user interfaces (or GUIs) such as the Oracle Universal Installer (OUI), the Database Configuration Assistant (DBCA), the Database Upgrade Assistant (DBUA), the Enterprise Manager (EM), and EM Database Control.
- Understand how Oracle manages metadata necessary to its operations through the data dictionary and the dynamic performance views.
- Be able to describe how Oracle manages storage with segments, tables, indexes, tablespaces, datafiles, extents, and blocks.
- Understand schema objects like tables, views, indexes, and sequences. Know how to create, maintain, and drop them.
- Understand Oracle Net Services, including the different architectures it supports, its basic features, how to configure it, the role of the listener, and the various client name resolution methods.
- Know what Oracle Shared Server is, how it works, and how to configure it. Know its advantages and disadvantages versus dedicated connections, and be able to describe and contrast how dedicated versus shared connections operate. Be able to configure and tune the Shared Server feature.
- Understand Oracle's security system. This includes how to grant and revoke privileges, the kinds of privileges available, managing user accounts, quotas, and database auditing in its various manifestations.
- Be able to manage data with the Structured Query Language data manipulation statements. Regarding PL/SQL and its components, understand how functions, procedures, packages, triggers, and PL/SQL programs work and fit together.
- Describe Oracle's approach to locking data, what it accomplishes, and why it is necessary. Know what locks and transactions are and how to resolve locking conflicts. Be able to configure and manage undo data.

- Understand the basics of database performance monitoring, including the new automatic proactive tools Oracle 10g offers. Be capable of using both the command line and the EM Database Control GUI for performance monitoring and management.
- Understand how the components of Oracle interact to support database recovery. Be familiar with the ways in which databases are backed up and recovered. Understand Oracle 10g's new flash backup and recovery features.
- Understand the different kinds of failure to which all databases are subject. Describe their differences and the various kinds of recovery procedures that apply to each. Be able to recover a database from key failures including the loss of a control file, a datafile, or an online redo log.

Tips

- You must know how to accomplish tasks both through the command line and through 10g's new Graphical User Interface (called the Enterprise Manager or EM). You'll have to know about commands, even for performing tasks in cases where most DBAs would use the EM because it's easier.
- 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 the process of elimination. Eliminate answers you know are wrong, and you'll often then be able to make your best guess from the remaining alternatives.
- You can always mark a question you're not sure about for later review. It's not unusual to find information in other questions that will later help you answer the questions you've marked for review.

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

1. Verify Oracle Corp's exam requirements
2. Get hands-on experience with the database
3. Study a book 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=41&p_org_id=1001&lang=US&p_exam_id=1Z0_042.

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.

Concepts

Oracle Corporation offers a wide variety of products. The Oracle Database 10g, the product this exam focuses on, is the centerpiece of the Oracle product set. The "g" in "10g" stands for the Grid computing model -- the idea that an organization's computer and communications network contains a variety of resources, which Oracle dynamically applies to problems as needed. The emphasis is on flexibility and the application of a large number of commodity computers or servers.

10g database comes in 5 standard packagings or editions. From most comprehensive to least, they are:

Database 10g Edition:	**Contains:**
Enterprise	All possible 10g features, either bundled or available at additional cost.
Standard	Like Enterprise Edition, but only runs on up to 4 processors.
Standard Edition One	Runs on up to 2 processors.
Personal	For an individual user, and tailored to single-user needs.
Lite	Includes only features relevant for mobile databases.

A relational database like Oracle 10g presents data to users as tables. A table consists of data presented in terms of rows and columns. Any particular data item or field exists at the intersection of a specific row and column. Every set of fields described by a particular column have a common datatype (such as numeric or character) that describes the kind of data the fields in that column contain.

A table is an un-ordered set of rows. From the end-user standpoint, a view is just like a table. A view contains no data itself but provides a "logical table" or a view into a table. Tables and views are *logical concepts only* that do not imply any particular way of physically storing the data on disk.

Data is manipulated and managed in relational databases like Oracle 10g via Structured Query Language or SQL. SQL statements classify into 4 categories:

SQL Category:	**Use:**
Queries	Uses the **SELECT** statement to retrieve data.
Data Manipulation Language (DML)	Uses **INSERT**, **UPDATE**, and **DELETE** statements to update databases.
Data Definition Language (DDL)	Statements that **CREATE**, **ALTER**, and **DROP** objects like tables, views, and indexes.
Data Control Language (DCL)	Statements that **GRANT** or **REVOKE** privileges (the ability to do something in the database or with its data).

To keep data accurate (i.e., maintain data integrity), relational systems like 10g process data updates in units called transactions. Either all the updates in a transaction are applied to the database as an atomic unit, or none of them are applied. A transaction begins with the first DML statement that a user issues and ends by one of the following actions:

- **COMMIT** - This command applies all updates in the transaction permanently to the database
- **ROLLBACK** - This command cancels all the updates in the transaction
- DDL or DCL - Issuing any DDL or DCL SQL statement implicitly **COMMIT**s the prior transaction
- Abnormal End - Abnormal termination of the database connection or session implicitly issues a **ROLLBACK** (Handled by Oracle processes, not by a user session)

Grouping SQL into atomic units or transactions is what permits the database to guarantee data integrity, and it underlies data recovery. This is because the database can then establish logical, database-wide points of consistency – you would recover the database to such a point of consistency if a database recovery were needed. Transactions are essentially individual "update units" the Oracle database uses to track its state at all times.

Metadata

Key to the Oracle database's operations is metadata – "data about data." The database requires such metadata to operate. Example metadata includes information about the structure and definition of tables and views, and lists of valid users and their access rights and privileges. 10g keeps metadata in two places: in its data dictionary and its dynamic performance views. The data dictionary contains more than 1,300 views. Data dictionary views start with any of 3 character strings that identify their scope or contents:

View starts with letters:	Meaning:
DBA_	All info and objects in the data dictionary. Only accessible by login ids having DBA privileges.
ALL_	All info and objects a particular user *can access.*
USER_	All info and objects a particular user *owns.*

Here's an example. You could look for information about table structure in 3 similar tables:

DBA_TABLES -- lists all tables in the database
ALL_TABLES -- lists all tables a particular user can access
USER_TABLES -- lists all tables a particular user owns

In addition to the data dictionary, an Oracle database maintains metadata in its dynamic performance views (sometimes called the **V$** views because each view name begins with those two letters). The 350-odd dynamic performance views contain dynamic data mainly of a statistical or operational nature that is lost each time the database is shut down.

Here are the key differences between the data dictionary and V$ views:

Data Dictionary views:	**V$ views:**
Contains persistent data that is *not* lost when the database is shut down	Contains data that is lost when the database instance is shut down
Only available for viewing when the database is open and running	Some V$ views are available when the database is in "lesser states" than open and running
The view name is usually plural (example: **DBA_TABLESPACES**)	The view name is usually singular (example: **V$TABLESPACE**)
Data contents are usually all uppercase	Data contents are usually lowercase

Both data dictionary and dynamic performance views are just that -- views -- logical tables that provide views into underlying internal structures that may be more complex and of various kinds of physical implementation.

Interfaces

So that users and administrators can interact with the database, 10g offers a variety of interfaces to retrieve and manipulate data, and to manage and administer the database objects that support the data.
SQL*Plus -- provides for command-line entry of SQL statements.

It requires:

- SQL*Plus client software installed on your computer
- Username and password to access a database
- Oracle Net connection string to access a database

SQL statements entered through SQL*Plus must end with either:

- A semi-colon (**;**)
- A slash (**/**) entered on its own line

Command-line Oracle tools like SQL*Plus also allow you to enter PL/SQL, Oracle Corporation's proprietary superset of the SQL language. PL/SQL extends and enhances SQL by adding procedural logic extensions such as looping, conditional control of execution, and the like. SQL itself is not a full programming language, but PL/SQL is.

Statements you enter via SQL*Plus are **not** case sensitive, with the exception of data entered between quoted strings (e.g., 'Oracle' does not equal 'ORACLE'). These statements are all the same to SQL*Plus:

```
select  * from table_name ;
Select * from table_name ;
SELECT  *  FROM   table_name ;
```

***i*SQL*Plus --** this product is the browser-based, "web version" of SQL*Plus.

It requires:

- A web browser
- The web address (or URL) of the host server running the *i*SQL*Plus web site software
- User name and password to access a database
- Oracle Net connection string to access a database
- The *i*SQL*Plus listener process must be running on the host server

*i*SQL*Plus divides the web browser screen into upper- and lower- halves. You enter SQL into the upper half and see results displayed in the lower half.

You can enter SQL statements to *i*SQL*Plus terminated by a semi-colon, the lone forward slash (/), or without any terminator at all. Click the **Execute** button to run statement(s).

To start the required *i*SQL*Plus listener on a host server, enter this command:

isqlplusctl start

To stop it, enter:

isqlplusctl stop

The default port for *i*SQL*Plus on the host is **5560**.

Once the *i*SQL*Plus listener is started, you can enter this web address or URL into your browser to access the database via *i*SQL*Plus:

http://machine_name.domain_name:port/iSQLplus

(where **machine_name** is the host name, **domain_name** is its domain, and **iSQLplus** is a constant)

Here is an example that accesses *i*SQL*Plus from your personal computer:

http://myhost.mydomain:5560/isqlplus

Now you'll see the login screen where you can enter the username, password, and connect string to access the database. Then you're in the *i*SQL*Plus interface.

Enterprise Manager (EM) -- EM is the new 10g GUI replacement for the old graphical user interface database management tool, the Oracle Enterprise Manager (OEM). EM can be installed in either of two forms: Grid Control or Database Control.

Grid Control provides central management of more than one database through the web browser-based EM GUI. It also manages additional resources beyond databases (such as application servers, web servers, etc). These are all called managed targets.

While Grid Control centrally manages many Oracle databases, Database Control manages only one database through EM. It is installed by default and uses port **5500** on the database server. The database it manages may or may not be clustered (using Oracle' Real Application Clusters or RAC product). Database Control can be viewed as a subset of the functionality of Grid Control -- it controls only a single database and does not support other kinds of managed targets. Database Control requires a Database Control Agent running on the managed database server.

You can access Database Control through your web browser by opening the browser and accessing this web address (URL):

http://host_name:port_number/em

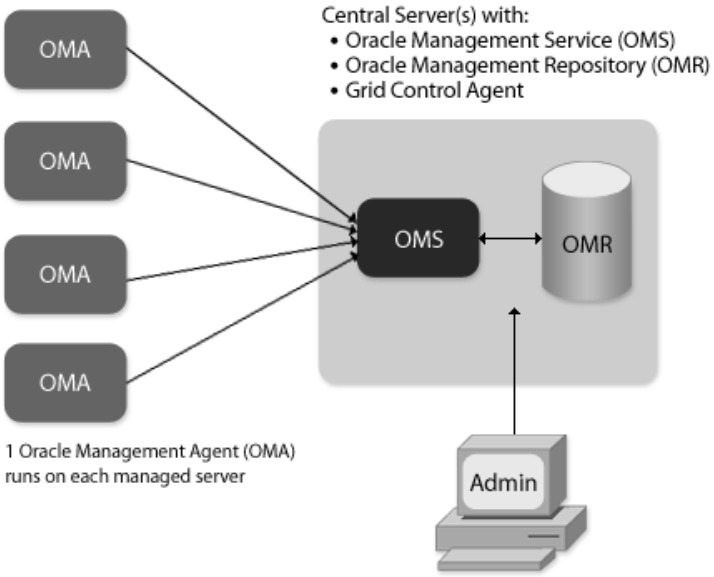
(where **host_name** is the host name, **port_number** is the port to access the host, and **em** is a constant)

The default port for Database Control is **5500**. Here's an example of entering the URL:

http://myhost:5500/em

To login, enter the user id **SYS** and its password and login as **SYSDBA. You can also login with any privileged account.**

Figure 1: Grid Control Architecture



To better understand Grid Control, **Figure 1** conceptually sketches the architecture through which Grid Control collects information on the resources it manages. Grid Control requires one Oracle Management Agent (OMA) running on each server you will manage. On the central management server(s), it requires one Oracle Management Service (OMS), one Oracle Management Repository (OMR) for OMS to store information in, and the Grid Control Agent.

OMA sends database configuration information to OMS every 12 hours and host configuration information to OMS every 24 hours**.** The OMR consists of two tablespaces (logical storage areas) within an Oracle database.

Here are line commands for starting and stopping OMA and the Database Control Agent:

Function:	**Command:**
Start the Oracle Management Agent	**emctl start agent**
Stop the Oracle Management Agent	**emctl stop agent**
Start Database Control Agent	**emctl start dbconsole**
Stop Database Control Agent	**emctl stop dbconsole**
Get status for the Database Control Agent	**emctl status dbconsole**

Whether through Grid Control or Database Control, the EM presents a graphical user interface (GUI) alternative for issuing commands and working on the command line in products like SQL*Plus and *i*SQL*Plus. The Exam asks questions both about the command-line commands and EM Database Control.

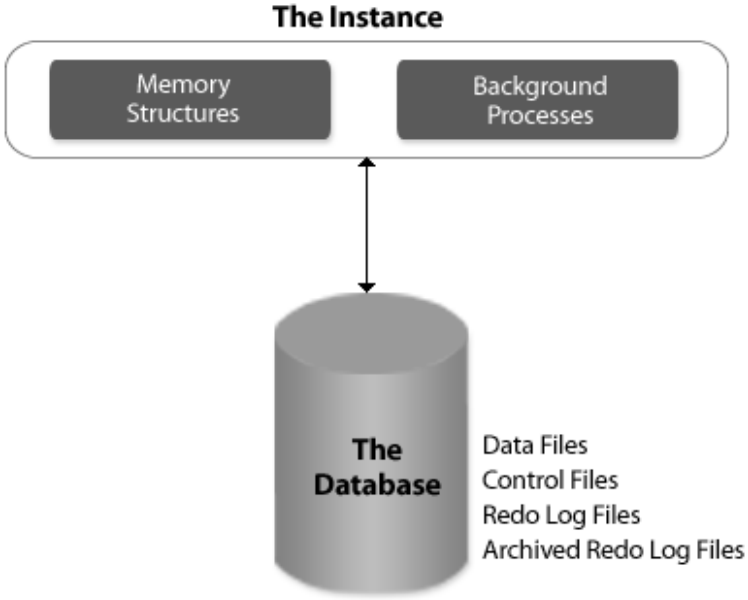
The EM panels have a button labeled **Show SQL** that will display any SQL command the GUI generates on your behalf. Use this valuable feature for learning the SQL commands to control the database.

Oracle Architecture

Let's explore Oracle's architecture at a high level. **Figure 2** shows that an Oracle database system contains 2 key components:

1. The Instance
2. The Database

Figure 2: Oracle Components



Instance Memory

The Instance consists of 2 components:

1. Memory structures
2. Background processes

The main memory block for the instance is called the System Global Area or SGA. It can be decomposed into further detail, but at the highest level it consists of 3 required components and 3 optional components. The 3 required SGA components are listed on the next page in **boldface**, while the 3 optional components are in regular typeface:

SGA Component:	Use:
Database Buffer Cache	Keeps the most-recently referenced data in memory
Shared Pool	Keeps the most-recently used SQL statements in memory
Redo Log Buffer	Keeps transaction log information in memory (also known as redo records)
Large Pool	Caches data for intensive operations like Recovery Manager (RMAN) and Shared Server
Java Pool	Caches most-recently used Java objects and code
Streams Pool	Caches queued message requests for the Advanced Queuing Option

When we refer to caching, we mean that Oracle retains a subset of the data in memory for faster access than storing it on disk would provide. Oracle uses a Least-Recently Used (LRU) algorithm to determine what information to keep in memory areas like the Database Buffer Cache and the Shared Pool. This minimizes data access from disk and SQL parsing (respectively).

Oracle can automatically manage the sizing of the SGA for you (now available in version 10g and recommended), or you can do it yourself manually (the pre-10g way). Oracle allocates and de-allocates SGA space in units called granules. Granules may be 4 M, 8 M, or 16 M, depending on your operating system.

Oracle Background Processes

We said before that an Instance consists of memory and background processes. Each Oracle background process handles one or more functions for the oracle database system. 5 are required, while many others are optional and may be present depending on what features you use.

The 5 required background processes are:

Name:	Required Background Process:	Use:
SMON	System Monitor	1. Performs crash recovery (the instance recovery Oracle automatically performs upon Startup, if necessary) 2. Manages sort space 3. Coalesces free space
PMON	Process Monitor	Cleans up after failed user connections to the database
LGWR	Log Writer	Writes redo records from the Log Buffer in SGA memory to the Online Redo Logs on disk
DBWn	Database Writer	Writes modified (dirty) data blocks from the Database Buffer Cache in SGA memory to the datafiles on disk
CKPT	Checkpoint	Manages Checkpoints by updating database files

For any background process that has one or more letter **n**'s, replace those letters with digits since there is more than one background process to name. For example, for Database Writers you might have several named **DBW0**, **DBW1**, and so on.

The optional background processes include the following:

Name:	**Optional Background Process:**	**Use:**
MMAN	Memory Manager	Automatically sizes key SGA components when 10g's Automatic Shared Memory Management (ASMM) feature is enabled.
MMON	Memory Monitor	Gathers/analyzes performance statistics used by the Automatic Workload Repository (AWR).
MMNL	Memory Monitor Light	Same functions as MMON, but activates when buffer space is filled rather than at periodic intervals.
RVWR	Recovery Writer	Writes recovery info to the Flash Recovery Area when the Flashback Database Recovery feature is enabled.
CTWR	Change Tracking Writer	Keeps track of updated database blocks for the new Fast Incremental Backup feature.
Snnn	Shared Server	These are the Shared Server processes that are shared among users when the Shared Server feature is enabled (Shared Server is explained in its own section of this Exam Manual).
Dnnn	Dispatcher	Puts user requests in a queue where the Shared Server processes receive them when the Shared Server feature is enabled.
ARCn	Archiver	Copies Online Redo Log file records to Archived Redo Log files. This is part of the operation called a log switch. This saves all redo log information. Archiver only runs when the database is in Archivelog mode. It is not run for databases in Noarchivelog mode.
RECO	Recoverer	Recovers **distributed** failed transactions.
CJQn	Job Queue Monitor	Assigns jobs to the Job Queues for the 10g job Scheduler.
Jnnn	Job Queue	Executes jobs in the Job Queue.
QMNn	Queue Monitor	Monitors message queues for the Advanced Queuing feature.
Qnnn	Parallel Query Slave	Parallel worker processes for the Parallel Query feature.

The background processes fall into these basic categories:

- Essential database operations (the 5 required processes)
- Oracle self-management and self-tuning (MMON, MMAN, MMNL, etc)
- Oracle's job Scheduler and queue processing (Jnnn, CJQn, QMNn)
- Other

The Windows operating system uses threads instead of background processes like Unix and Linux. The Windows Service **OracleService*Instance_Name*** is associated with each instance. This Service must be used to start up the instance. Each listener running under Windows is also a separate Windows Service, as explained later in the section on "Oracle Net Services." In Windows environments, it is possible to have the database Windows Service up and running, but the database itself is down or has not been started.

Oracle Database Files

While the instance consists of memory and background processes, the database consists of persistent files on disk.

The 3 key files that comprise the database are:

- *Datafiles* – hold the data.
- *Control Files* – contain information about the physical structure of the database; used by Oracle during its ongoing operations. Contains the database name, creation timestamp, and names, locations, and sizes of all datafiles and redo log files as well as the current SCN.
- *Redo Log Files* – contain change records ("redo" information) used to ensure against data loss (for example, they can be applied during database recovery).

Datafiles -- Datafiles are logically grouped into a higher-level logical construct called the tablespace. Tablespaces may either be permanent (permanently holding data) or temporary (used for such temporary or transient operations as sorting data). Each datafile belongs to one and only one tablespace.
These data dictionary views for datafiles and tablespaces are key:

- **DBA_DATA_FILES** - Tracks datafile information
- **DBA_TABLESPACES** - Keeps permanent tablespace information
- **DBA_TEMP_FILES** - Holds info on temporary tablespaces

The most frequently used pages or blocks of data are held in the SGA memory for quickest access. This is in the database buffer cache area of the SGA. The Database Writer (DBWn) background process writes updated (dirty) pages from the database buffer cache to the datafiles to keep those datafiles accurate and externalize data changes to disk storage.

Database Writer writes dirty pages to datafiles when:

- The number of modified and **COMMIT**'ed buffer blocks in the database buffer cache is too large.
- At a database Checkpoint Event (A checkpoint event means that dirty blocks are written from the database buffer cache to datafiles on disk and that the headers of all datafiles and control files are updated.)

- Oracle has to search for too long to find a free buffer
- The instance is shut down (except for by a **SHUTDOWN ABORT**)
- A tablespace is placed into Backup Mode, taken Offline, or changed to **READ ONLY** status
- A segment is dropped (A segment is the object that stores data in datafiles outside of the data dictionary)

Control Files -- Control Files are used by Oracle to direct its operations. They are multiplexed, meaning that Oracle keeps identical copies of them to ensure reliability and availability. (Outside of Oracle databases, multiplexing is usually referred to as *mirroring*). These multiplexed copies should be stored on different disks (using different disk controllers) for greatest reliability.

The Checkpoint background process, CKPT, updates the control files to keep them synchronized with datafile contents and ensures a system-wide point of data consistency. It does this using System Change Numbers, or SCNs, which are the unique, ascending numbers assigned to transactions. A common SCN between the control files and the datafile headers indicates a database-wide point of consistency. View **V$CONTROLFILE** contains information on the control files.

Contents of the control files include: database info (name and creation timestamp), datafiles info (names, locations, offline/online status), redo log info (names, locations), redo log archive info, whether the database is in Archivelog or Noarchivelog mode, tablespace names, most-recent checkpoint info, the current log sequence number, and Recovery Manager (RMAN) info for backups. You should always back up the control file after making any structural changes to the database (for example, you run DDL that creates or changes tablespaces or datafiles). You can set up 10g to ensure that all database backups include the controlfile through a feature called control file autobackup.

Control files include both reusable and non-reusable portions. RMAN, for example, uses the reusable part to store backup datafile names. Initialization parameter **CONTROL_FILE_RECORD_KEEP_TIME** specifies the number of days to keep such reusable data, and serves to limit the size of the control file. The default value for this parameter is 7 days and its maximum permissible value is 365 days.

Several background processes update the control files. Log Writer (LGWR) updates the control file's current log sequence number. Checkpoint (CKPT) updates its recent checkpoint information. For a database in Archivelog mode, the Archiver (ARCn) updates its archive log name and log sequence number.

To add a control file, the procedure varies slightly depending on whether you use a **PFILE** or **SPFILE** to configure Oracle upon startup (a **PFILE** is a textual initialization parameter file, while an **SPFILE** is its binary equivalent). For a **PFILE**, **SHUTDOWN NORMAL** the database. Copy one of the valid control files to where you need the extra one. Change the **CONTROL_FILES** initialization parameter to reflect the new control file, then **STARTUP** the database.

If you use an **SPFILE**, alter the **SPFILE** to reflect the new control file by this command while the database is open:

```
ALTER SYSTEM SET CONTROL_FILES =
        … [ your list of control name files goes here ] …  SCOPE=SPFILE ;
```

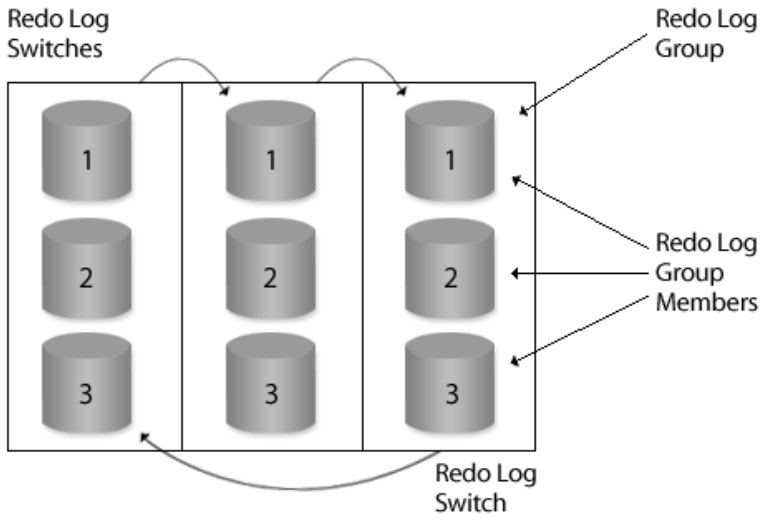
Then **SHUTDOWN NORMAL** the database. Copy a valid control file into the new location, then **STARTUP** the database.

Be sure to specify **SCOPE=SPFILE** in this process, because this only applies the new **CONTROL_FILES** specification after instance startup.

Redo Log Files -- Online Redo Log Files contain the information necessary to re-create any transaction applied to the database. They contain Redo Records (or Redo Entries). Redo records are collected in the SGA memory Redo Log Buffer and written to the online redo log files by the Log Writer background process (LGWR).

Like the control files, the redo log files are multiplexed (mirrored) for reliability. A set of individual redo log files is called a Redo Log Group, while each member of the Group is called a Redo Log Group Member. Each member is typically in a separate disk location to guard against negative impact from disk failure. A redo log group must have a minimum of one member each, and a database must have a minimum of two redo log groups.

Figure 3: Redo Log Files



As **Figure 3** shows, LGWR writes to all the members in a redo log group simultaneously. So LGWR concurrently updates all members of a redo log group. The members should all be the same size. When the members become filled, LGWR then switches to write to another redo log group, and continues to do so, in a circular fashion, one after another. Whenever LGWR points to a new redo log group to write to, it is called a log switch.

If the database is in Noarchivelog mode, when LGWR circles around as the online redo log files fill, the contents of the older online redo log files are overwritten. If the database is in Archivelog mode, log archiving occurs before reusing any online redo log file. During this procedure the Archiver background process(es) ARCn writes the redo records from the online redo log file to Archived Redo Log files. This preserves the redo information before the online redo log file data is overwritten. Should there be no disk space available to create the archived redo log file, the database will hang until space becomes available.

Online redo log files that are currently being used (written to) are termed the current log files. They are active. The online redo log files that are not presently being written to are inactive. You can force a log switch by the command:

```
ALTER SYSTEM SWITCH LOGFILE ;
```

Redo records are essential to recover all database transactions in the event of a database or media failure, so databases that require complete recoverability should be run in Archivelog mode.

When you create a database, the maximum number of allowed online redo log file groups is specified by the parameter **MAXLOGFILES**, while the maximum number of members in each group is specified by parameter **MAXLOGMEMBERS**. Within these limits, you can add log file groups and members. To add a new redo log file group, run the command **ALTER DATABASE ADD LOGFILE GROUP**. To add a member, execute the command **ALTER DATABASE ADD LOGFILE MEMBER**.

You can drop a group via **ALTER DATABASE DROP LOGFILE GROUP**, and you can drop a member by **ALTER DATABASE DROP LOGFILE MEMBER**. Of course, you would never drop the active (current) log file group.

If an online redo log file becomes corrupted, you can reset or clear it. Do this by:

```
ALTER DATABASE CLEAR LOGFILE GROUP n ;
```

where **n** is the integer that represents the log file group. View **V$LOGFILE** lists the online redo log file groups and their members, while view **V$LOG** displays the online redo log group that is active and being written to by LGWR.

To rename an online redo log file member:

1. **SHUTDOWN NORMAL** the database.
2. Copy or rename the redo log file member using the appropriate operating system command (such as **copy** or **cp** or Windows' **Copy**/**Paste**).
3. **STARTUP MOUNT** the database.
4. Rename the log file member in the control files by:

   ```
   ALTER DATABASE RENAME FILE 'old_name' TO 'new_name' ;
   ```
5. **ALTER DATABASE OPEN** to open the database.
6. Whenever you make a "structural change" to the database, you should backup the control file immediately afterwards. Here's one way to do this:

   ```
   ALTER DATABASE BACKUP CONTROLFILE TO 'file_name' ;
   ```

It is important to understand exactly when the Log Writer background process LGWR writes redo entries from the Redo Log Buffer in the SGA to the current online redo log group.

It does so at any of these times:

- When a user **COMMIT**'s a transaction
- Every 3 seconds
- The Redo Log Buffer in the SGA becomes one-third full
- The Redo Log Buffer in the SGA contains 1 M of redo information
- Before the Database Writer process DBWn writes when a database checkpoint occurs. (LGWR always writes its log buffer records to the online redo log files *before* the Database Writer, DBWn, writes database buffer cache blocks to the datafiles. This is called write-ahead logging and is fundamental to how oracle databases ensure data integrity.)

Oracle Initialization Parameters

To control how an Oracle database is configured, Oracle provides a group of initialization parameters or "init parms." There are about 25 basic parameters, parms that you will usually set, while there are over 100 advanced parameters (parms that you will rarely need to set). There are also over 1,000 undocumented parameters, parms that Oracle Corp. does not document and recommends that you not use. These all start with the underscore " _ " character. You do not need to know any of them for the exam.

The 10g initialization parameters are different from those of previous Oracle releases (such as 9i), so be sure your knowledge of the parms is up-to-date. 10g obsoletes many of the 9i parms.

You need to know the 25 basic parameters shown in the table for the exam:

Basic Initialization Parm:	**Use:**
DB_CREATE_FILE_DEST	If using Oracle-Managed Files (OMF), provides the directory location for OMF to use
DB_CREATE_ONLINE_LOG_DEST_n	If using OMF, provides directory location for the online Redo Log Files
DB_DOMAIN	Domain the database resides in on the network
DB_NAME	Name of the database mounted by the instance
DB_RECOVERY_FILE_DEST	If using the Flash Recovery feature, provides the location where recovery files reside
DB_RECOVER_FILE_DEST_SIZE	If using the Flash Recovery feature, specifies the size of the recovery file area
DB_UNIQUE_NAME	Globally-unique database name in the organization
INSTANCE_NUMBER	If using Real Application Clusters (RAC or clustering), identifies the instance in the cluster
JOB_QUEUE_PROCESSES	Number of background jobs to start for processing Scheduler jobs
LOG_ARCHIVE_DEST_n	Where to write Archived Redo Logs
LOG_ARCHIVE_DEST_STATE_n	How the Archived Redo Log locations should be used
NLS_LANGUAGE	Default database language
NLS_TERRITORY	Default database region
OPEN_CURSORS	Maximum number of allowable open cursors for an individual session
PGA_AGGREGATE_TARGET	Total amount of memory for the Program Global Area (PGA)

PROCESSES	Maximum number of operating system processes that can connect to the instance
REMOTE_LISTENER	Network name for the address list of remote Oracle Net listeners
REMOTE_LOGIN_PASSWORDFILE	Specifies if the instance uses a Password File to authenticate users during login
ROLLBACK_SEGMENTS	Use only if not using Automatic Undo Management to indicate the number of Rollback Segments (this parm is typically used only in pre-10g configurations)
SESSIONS	Maximum number of sessions that can connect to thedatabase
SGA_TARGET	Maximum System Global Area (SGA) size, from which Oracle typically allocates internal components
SHARED_SERVERS	If using the Shared Server feature, tells how many Shared Servers to start when the instance starts
STAR_TRANSFORMATION_ENABLED	Tells the optimizer to consider star transformations when optimizing queries
UNDO_MANAGEMENT	Tells whether to use Automatic Undo Management
UNDO_TABLESPACE	Specifies the tablespace to contain Undo records

The parameters can be set in one of either of these 2 kinds of initialization parm files:

- **PFILE** - Called "Parameter File" or "textual parameter file"
- **SPFILE** - Called "Server Parameter File" or "binary parameter file"

To get the full benefit of the ability to dynamically change the initialization parameters, and to take advantage of all 10g's self-management features, you need to use an **SPFILE** rather than a **PFILE**. Oracle Corp recommends using an **SPFILE**. The differences between the PFILE and SPFILE are shown in the table:

PFILE:	**SPFILE:**
A text file you can edit	A binary file you should never edit
Named: init*Instance_Name*.ora (example: initProd.ora)	Named: spfile*Instance_Name*.ora (example: spfileProd.ora)
After changing any parameters, you must shut down and re-start the instance for changes to take effect	Most changes can be dynamically applied to the instance without shutting it down and re-starting it
Create it from an **SPFILE** by the command: **CREATE PFILE FROM SPFILE**	Create it from a **PFILE** by the command: **CREATE SPFILE FROM PFILE**
The original way of setting instance parms	The more modern way of setting instance parms that supports many new features

You can remotely shutdown and start the db with an SPFILE without having access to it (this avoices the pfile= or spfile= parameter during startup).

When you start up an Oracle database, Oracle determines which **PFILE** or **SPFILE** file to use by this search order:

1. Looks for file **spfile$ORACLE_SID.ora**
2. Looks for file **spfile.ora**
3. Looks for file **init$ORACLE_SID.ora**

The default location for these files is **$ORACLE_HOME/dbs** on Unix and Linux systems and **%ORACLE_HOME%\database** on Windows systems. You can always override Oracle's default search order and file location by specifying the exact initialization file you want to use on the Oracle **STARTUP** command.

There are several ways to alter the initialization parameters. Many can be dynamically altered (changed while the database is up) by the **ALTER SYSTEM** command -- if you use an **SPFILE** file rather than a **PFILE**. You can choose to apply changes to only the running instance or permanently to the database whenever it runs. You can also alter init parms through the EM interface. A GUI tool called the Database Configuration Assistant (DBCA) can also be used to alter parameters. (We describe DBCA in detail later).

When making parameter changes with an **SPFILE**, the **SCOPE** parameter on the **ALTER SYSTEM** statement determines whether your change affects only the currently running instance, the **SPFILE** only**,** or both**.** Here are the possible values you can use on the **SCOPE** parameter of the **ALTER SYSTEM** statement:

ALTER SYSTEM parameter:	**Use:**
SCOPE=MEMORY	Changes ONLY apply as long as the instance is still up, but they do apply immediately. This is the DEFAULT.
SCOPE=SPFILE	Changes take effect ONLY after the instance is shut down and restarted.
SCOPE=BOTH	Changes apply immediately, plus it will also apply to all future use of the instance.

Installing Oracle and Creating Databases

Installing Oracle

You perform a number of steps to install Oracle:

1. Collect and review the install documentation
2. Review system requirements
3. Plan the install and perform any pre-install steps
4. Run the Oracle Universal Installer (OUI)
5. Perform any post-install steps

All Oracle documentation (including installation guides and release notes) is bundled with the Oracle database (on a separate CD-ROM). Or it can be freely downloaded from the Oracle Technical Network, Oracle Technet, at http://technet.oracle.com.

Oracle system requirements are:

Resource:	**Oracle Install Requirement:**
Memory	256 M minimum or 512 M with EM Database Control (recommended).
Swap Space	1 G or two times the amount of memory (RAM).
Temp Space	400 M on **/tmp** directory for Unix systems.
Disk Space	1.5 G for Oracle. 1 G more for creating a database using the Database Configuration Assistant (DBCA).

Pre-Install Steps -- On Unix and Linux systems you will need to configure kernel parameters prior to installing Oracle for:

- Maximum number of concurrent processes
- Maximum number of concurrently-open files
- Maximum size for a shared memory segment (this is key because you don't want the SGA to be paged or not fit into memory)

Create an Oracle user account — the login id that "owns" the Oracle directories. Most sites choose to use **oracle** for this account. Create an operating system group for the Oracle user, usually named **dba**.

You may need to create various mount points — filesystems used for storing Oracle binaries and various Oracle-generated files (like the dump files and the Alert log). You may also want to create various directories under the mount points for files. On Windows systems, instead of mount points and directories, use drive letters, and folders underneath those drive letters.

Create 2 key environmental variables prior to installation:

- **ORACLE_HOME** - Directory where you install the 10g binaries
- **ORACLE_BASE** - Top-level directory for Oracle software

In Unix systems, these are typically preceded by a dollar sign (**$ORACLE_HOME** and **$ORACLE_BASE)**.

On Windows, these environmental variables are preceded and followed by a percent sign (**%ORACLE_HOME%** and **%ORACLE_BASE%).**

The directory paths to which they are assigned typically look similar to these:

Environmental Variable:	**Unix:**	**Windows:**
ORACLE_BASE	/u01/app/oracle	D:\ORACLE
ORACLE_HOME	/u01/app/oracle/product/10.1.0	D:\ORACLE\ORA101

A key part of pre-install planning is figuring out the naming for directories and files. Oracle Corp. recommends Optimal Flexible Architecture (or OFA), which simplifies this process and offers consistency across systems by specifying naming conventions for:

- File systems and mount points (or Windows' drive letters and folders)
- Directory paths
- Database and Oracle file names
- Standardized locations for database files and components

OFA uses the standard **ORACLE_HOME** and **ORACLE_BASE** directories as the higher-level qualifiers for a wide variety of standardized directory locations. OFA specifies these file names:

- Control files - control***n***.ctl
- Datafiles - filename***n***.dbf
- Redo log files - redo***gm***.log (where **g** is the group and **m** is the member number)

If you use Oracle's automated file management feature, Oracle-Managed Files or OMF, OMF generates file names automatically.

OFA recommends a directory structure for datafiles that contains the database name in its path and uses at least 5 directory mount points. For a database named **TEST** you would have something like this:

- /u01/oradata/TEST
- /u02/oradata/TEST
- /u03/oradata/TEST
- /u04/oradata/TEST
- /u05/oradata/TEST

OFA recommends a number of administrative sub-directories located under the directory **$ORACLE_BASE/admin/TEST**:

Administrative Directory:	**Use:**
adhoc	Stores adhoc SQL scripts
arch	Where the Archiver process writes the Archived Redo Logs
adump	Where audit trail information is written if Auditing feature is used
bdump	Alert log -- Oracle's main message file -- and trace files written by Oracle background processes are written here
cdump	Core dump files are written here
create	SQL scripts to create the database are stored here
exp	Export utility files are written here
logbook	Stores files that document your database activity
pfile	The parameter initialization file or **PFILE** resides here
udump	Trace files written by user processes go here

Common directories based on the **ORACLE_HOME** and **ORACLE_BASE** environmental variables include:

Directory:	**Use:**
$ORACLE_HOME/dbs	Default location for **PFILE** and **SPFILE** on Unix systems
%ORACLE_HOME%\database	Default location for **PFILE** and **SPFILE** on Windows systems
$ORACLE_HOME/network/admin	Default location for Oracle Net configuration files on Unix systems
%ORACLE_HOME%\network\admin	Default location for Oracle Net configuration files on Windows systems
$ORACLE_HOME/rdbms/admin	Default location for Oracle-provided database setup and configuration scripts on Unix systems
%ORACLE_HOME%\rdbms\admin	Default location for Oracle-provided database setup and configuration scripts on Windows systems

Non-OFA environmental variables to know are:

Environmental Variable:	**Use:**
$LD_LIBRARY_PATH	Unix -- Location of Oracle shared object libraries
%LOCAL%	Windows -- Default Oracle Net connection string
$TWO_TASK	Unix -- Default Oracle Net connection string
ORACLE_SID	Default instance to connect to
PATH	Where the operating system should look for executables
TNS_ADMIN	Where Oracle Net configuration files are located

The Oracle Universal Installer (OUI) -- On systems with AutoRun, insert the database CD-ROM and the OUI automatically starts. On certain Unix systems, you may need to insert the CD-ROM, then mount it via a Unix **mount** command. Then start the Installer by the command **runInstaller.sh**. Specify the **-ignoreSysPrereqs** option if you need to ignore the system prerequisites check the Installer otherwise automatically performs.

One key to a successful install is to ensure that all relevant environmental variables have been set correctly. These include: **ORACLE_BASE**, **ORACLE_HOME**, and **PATH**. On Unix systems, add **LD_LIBRARY_PATH** and **DISPLAY** to this list.

The first install panels ask for the operating system group of the user doing the install (normally **dba**). It also suggests an inventory location, based on the value of **ORACLE_BASE**. The inventory directory is where Oracle keeps its inventory file, a list of what is installed on that system. The system will also require running script **orainstRoot.sh** under the system super-user id (Unix **root** or Windows **Administrator**) to set up some directory structures. Eventually the Installer asks what products you want to install and it copies them to disk and compiles them. Towards the end of the Install you must run another script under the super-user id, named **root.sh.** This script copies files to directories outside of **ORACLE_HOME** and sets permission bits. The install concludes while Oracle completes some post-install tasks.

Creating Databases

The Database Configuration Assistant (DBCA) is the basic Oracle 10g GUI tool for creating, configuring, cloning (copying), and dropping a database.

If you pick the option labeled **Create a Starter Database** while running the Installer, then the OUI will automatically invoke DBCA to create your database. Or you can enter command **dbca** to start the product stand-alone at a command-line prompt (or select it from the Windows **Program** menu).

DBCA operates off of database templates, XML-based documents that describe databases. Its bundled templates include generic databases for: **Data Warehouse**, **Transaction Processing**, and **General Purpose** databases. The **Custom** option allows you to create your own customized templates. As you create your own templates, DBCA offers options to manage them (modify and delete them). Template characteristics include these attributes: Common options, Character sets, Initialization parameters, Control files, Datafiles, Tablespaces, and Redo log groups.

Beyond creating new databases and templates, DBCA also aids in configuring existing databases. It's an easy GUI for changing many database characteristics.

Using DBCA -- Starting DBCA presents you with the first panel asking whether you want to:

- Create a new database
- Configure a database
- Delete a database
- Manage database templates

Creating a new database requires an Oracle System Identification name or SID. This is the name of the instance associated with the database. It can have a maximum of 8 characters and must be unique on the server. You also need a global database name, which consists of:

database_name . domain_name

For example, if your database name is **prod** and your domain is **my_company.com**:

prod.my_company.com

The **domain_name** is typically the same as the network domain in the organization.

Creating a new database also asks which of the templates you will use: **Custom database**, **Data Warehouse**, **Transaction Processing**, or **General Purpose**.

You will be asked whether to use the **Grid Control** or **Database Control** feature of **Enterprise Management**. Recall that the latter only manages a single database. You can set up email or pager notifications for problems. The **Enable Daily Backup** option automatically sets up and schedules an RMAN backup job for the database.

You can set passwords for 4 default database user ids:

Use:	
SYS	Owns the data dictionary tables
SYSTEM	General administrative user id
DBSNMP	Used by Enterprise Manager
SYSMAN	Administrator for the Enterprise Manager

DBCA supports three database storage options:

Storage Option:	**Use:**
File System	Use operating system files for storage. DBCA uses OFA naming for the files and directories.
Raw Devices	Oracle uses raw disk partitions for storage.
Automatic Storage Management (ASM)	ASM is a new storage management feature in 10g. It uses groups of disk (disk groups) to manage storage and provides such features as mirroring and striping. ASM can replace traditional operating system Volume Managers for Oracle disks. It uses OMF. The ASM feature requires a special Oracle instance, called the ASM instance, which DBCA automatically creates and configures for you.

DBCA lets you configure a flash recovery area for the database. The Recovery Manager (RMAN) uses this area to store backups on disk (for faster recoveries). The default location for the flash recovery area is **ORACLE_BASE\flash_recovery_area** and its default size is 2048 M. Both defaults can be modified. 10g includes a range of Flash Recovery features that use the flash recovery area.

DBCA panels also allow you to specify:

- Whether to enable archive logging (which permits *point-in-time recovery*)
- Which character sets to use
    - National character set - How to represent Unicode characters
    - Default language - Locale-specific information
    - Default date - How to specify dates

- How much memory to use and whether Oracle should automatically manage it (for the System-Global Area or SGA and the Program Global Area or PGA).
- The default data block size (page size) for the database (defaults to 8 K).
- The maximum number of processes that can connect to the database (these minimally must include Oracle's required background processes).

You **cannot** change the default block size of the database once it has been chosen. 10g does, however, permit creating tablespaces that use block sizes other than the default.

DBCA displays storage options through a tree structure. This allows you to specify storage and file characteristics for Control Files, Datafiles, Redo Log Groups, and Tablespaces.

To clone a database is to make a copy of it. DBCA clones database through two kinds of templates:

- Seed -- This includes datafiles
- Non-seed -- This is structure-only (no data)

DBCA puts the source database in the **MOUNT** state to create the template. After the template has been created, the database can be re-opened (the database is **OPEN** and available while the clone database is being created). One template can be used to create many new databases, if desired.

Controlling Databases

To startup and shutdown the database, you need an administration account with either the **SYSDBA** or **SYSOPER** authorization. The accounts are equivalent except that **SYSDBA** has additional authorities to work with database objects. These authorizations are managed either by:

- An Oracle password file
- Operating system control (an operating system group lists the allowable user ids having these authorities)

In a new database, only the **SYS** user login initially has **SYSDBA** authorization.

Use the **STARTUP** and **SHUTDOWN** commands to start and stop Oracle databases.

Starting a database fully passes it through these 3 stages in this order:

1. **NOMOUNT**	Starts the instance only (allocates *memory structures* and starts the *background processes*). Accesses the **PFILE** or **SPFILE**.
2. **MOUNT**	Reads the control file (and verifies it to the **PFILE** or **SPFILE**).
3. **OPEN**	Accesses all datafiles and online redo log files and makes the database available to all users.

The **STARTUP** command parameters you can use are:

Command:	Use:
STARTUP NOMOUNT	Puts the database into the **NOMOUNT** state. Useful for running the **CREATE DATABASE** command.
STARTUP MOUNT	Puts the database into the **MOUNT** state. Useful for administrative tasks including recovery, changing file locations, or changing the archival mode of the database.
STARTUP OPEN	Makes the database available to all users. This is the default mode if you just specify **STARTUP** without parameters.
STARTUP FORCE	Usually only used if you have trouble starting the database (for example after a power loss). Internally does a **SHUTDOWN ABORT** followed by a **STARTUP**. Unique in that it can be issued regardless of what mode the database is in.
STARTUP RESTRICT	Starts the database in **OPEN** mode but only users with **RESTRICTED SESSION** privilege can access it. Useful for performing database maintenance or stand-alone DBA work. Transit to a fully **OPEN** and available database by issuing: **ATLER SYSTEM DISABLE RESTRICTED SESSION;**

Here are the **SHUTDOWN** commands, listed in order from least to most aggressive:

Command:	Use:
SHUTDOWN NORMAL	Allows no new database connections and waits until all users disconnect before shutting down the database.
SHUTDOWN TRANSACTIONAL	Allows no new database connections or transactions and waits until all running transactions complete, then disconnects users and shuts down the database.
SHUTDOWN IMMEDIATE	Allows no new database connections and rolls back in-flight uncommitted transactions, then disconnects users and shuts down the database. While **SHUTDOWN TRANSACTIONAL** preserves any user work in progress, **SHUTDOWN IMMEDIATE** loses this work by forcing rollbacks of current transactions.
SHUTDOWN ABORT	Allows no new database connections. Terminates any SQL statements in progress *without* rolling them back and disconnects all users and shuts down the database. This is the only **SHUTDOWN** command that is *not clean* and requires recovery the next time the database is started. Oracle automatically performs instance recovery (also known as crash recovery) upon startup to handle this situation.

As well as line commands, you can start and stop databases through the EM GUI panels. You can also issue these *i*SQL*Plus commands:

STARTUP [NOMOUNT | MOUNT | OPEN] [PFILE/SPFILE=] [RESTRICT]

and

SHUTDOWN [NORMAL | IMMEDIATE | ABORT | TRANSACTIONAL]

You can review database starts and shutdowns through the database's Alert Log. Oracle continually appends to this log with information like:

- Database starts and stops, including init parms that differ from the defaults
- Important database errors, such as ORA-600 internal errors
- Database-wide administrative actions, such as **ALTER SYSTEM** and **ALTER DATABASE** commands

The Alert Log is written to the directory specified by the initialization parameter **BACKGROUND_DUMP_DEST.**

Windows Systems -- Starting the **Oracle Service** automatically starts the database. If you shutdown the database using EM Database Control, SQL*Plus, or *i*SQL*Plus, the **Oracle Service** continues running, but the database is down until you issue a **STARTUP**.

User Connections and Oracle Shared Server

Dedicated Versus Shared Connections

User connections come into the Oracle database server in one of two ways:

- Dedicated
- Shared

In the dedicated model, there are two processes involved per user connection:

- The user process
- The server process

The user process exists either on the user's personal computer or on an application server in a multi-tiered architecture. The user process gains a connection into the database server. The connection then establishes a session in the instance. A dedicated server process performs SQL and manages the user's database interactions. This dedicated server process is sometimes called a shadow process. There is a one-to-one relationship between the user processes and the dedicated server or shadow processes.

The alternative to a dedicated connection is a shared connection. Oracle's Shared Server is an optional feature of the product. In this configuration, each server process serves multiple users. The goal is a more efficient use of resources and a more scalable system.

Applications that are best for **dedicated connections** include:

- Sessions that involve lengthy queries with much I/O and large result sets
- Sessions that generate lots of network traffic
- Many data warehouse (DW) and decision support systems (DSS)
- Sessions that involve certain commands not allowed in shared connections (such as starting up or shutting down the database, and RMAN recoveries)
- Utility functions poorly suited to shared servers (e.g.: bulk loads, table rebuilds, index builds, gathering statistics)

Applications that are best for **shared server connections** are:

- Short transactions
- Online Transaction Processing systems (OLTP)
- Interactions involving lengthy human "think time"
- Many web-based applications with significant time gaps between actions
- Applications that handle small amounts of data, generating little network traffic

Oracle instances can simultaneously support both dedicated and shared server connections when the shared server feature is enabled.

Once you've enabled the shared server feature, if you want a dedicated connection, you must specifically request it. In other words, once shared server is on, any incoming client gets a shared connection by default.

To get a dedicated connection, the client must send information that requests it in the connection descriptor. Two different ways to do this are:

1. Change the **tnsnames.ora** file to include the parameter **SERVER=DEDICATED**. Here's an example:

```
ORCL =
        (DESCRIPTION =
                (ADDRESS = …
                (CONNECT_DATA =
                        (SERVICE_NAME = orcl)
                        (SERVER = DEDICATED)
                )
        )
```

2. Go into the **Oracle Net Manager** GUI tool on the client and change the **Connection Type** to **Dedicated Server**.

How Shared Server Works

Oracle manages shared server connections entirely differently than dedicated connections. Among shared server's advantages are:

- Higher scalability for suitable applications.
- Lesser server resource requirements for large numbers of users.
- Connection Pooling - more efficient use of connections, as idle connections can be multiplexed out and then reconnected when ready. Connection pooling shares connections and loads multiple users per connection.
- Supports greater number of connections without adding more hardware.

The shared server feature is a *scalability benefit* -- but **not** a *performance benefit*. It does not add more resources to your system, it simply uses those you have in a more efficient way for applications having certain characteristics.

These are the steps involved when Oracle database manages users through the shared server feature:

1. The client sends in a request. This is picked up by a Dispatcher background process on the server.
2. The Dispatcher places the request on a Request Queue in the SGA.
3. Any one of the Shared Server processes may handle the request (depending on which is available or least busy).
4. The Shared Server process that handled the request places the results on the Dispatcher's Response Queue in the SGA.
5. The completed request is passed back to the client.

There is one Request Queue that all requests go into. All dispatchers pick up the requests from this single request queue. Requests in the request queue are picked up and serviced in First-In, First-Out (FIFO) order. Requests cannot be prioritized within the request queue for 10g.

Each dispatcher has its own Response Queue. So, the number of response queues equals the number of dispatchers. For example, if you have 5 dispatchers running on your system, you will have 1 request queue and 5 response queues. Both the request queue and the response queues reside in SGA memory.

Each connection serviced by a dispatcher is associated with a shared memory segment and forms a virtual circuit. The memory area supports the virtual circuit for communication that the dispatcher uses to manage communication between the client and shared server process.

The Listener process has a new role in a shared server environment. It supplies the client with the address of which dispatcher to connect to. The Oracle Process Monitor (PMON) background process tells the listener which dispatcher is responsible for servicing each virtual circuit. Thus, listener knows which connections each dispatcher services. This enables the listener to manage load balancing, a process by which the listener assigns incoming requests to the least-busy dispatchers. It also enables the listener to send requests to the least-busy instance in a clustering environment (Oracle's clustering feature is called Real Application Clusters or RAC). By clustering, we mean that multiple Oracle instances simultaneously attach to and service one Oracle database.

Here are the steps in this process from the standpoint of the listener:

1. The client gets the Oracle service name resolved and contacts the database server and its listener.
2. The listener hands off or redirects the client connection to the least-busy dispatcher.
3. The dispatcher process manages the client server request.
4. **PMON** registers connection information with the listener. This keeps the listener informed as to the load on the dispatchers and allows the listener to implement load balancing across instances in a cluster and among multiple dispatchers.

Memory for Shared Server

For dedicated connections, Oracle maintains several data items per client in a dedicated memory area for that client session called the Program Global Area, or PGA. The specific items kept in the PGA for the client are:

- User session data (including bind variables)
- Cursor state
- Stack space (or sort space)

For the shared server, all this data except for the stack space is moved to a part of the Large Pool within the SGA called the User Global Area (or UGA). Configure the large pool and the SGA with enough memory to absorb this extra session information that would otherwise be held within the PGA. So using shared server, the UGA within the large pool holds:

- User session data (including bind variables)
- Cursor state

Using shared server, the PGA now just holds:

- Stack space (or sort space)

Remember that the other big memory change for shared server is that there is now 1 request queue, plus 1 response queue per each dispatcher process. Both request and response queues are allocated from SGA memory.

Configuring Shared Server

Shared server can be configured from the command line in tools like SQL*Plus or by the GUI of the Enterprise Manager (EM). Though you'd probably use the easier EM, in practice you should also know the command-line approach for the Exam. Note that all panels of the EM (not just those that manage Shared Server) have a **Show SQL** button. Pressing this displays the EM-generated SQL statement(s) that EM will execute on your behalf. This is a superb tool for learning SQL administration commands.

Another way to configure shared server is when you create a database through the Database Configuration Assistant (DBCA). This GUI tool has several panels that allow you to set appropriate initialization parameters to run shared server.

Initialization parameters for shared server include **SHARED_SERVERS**, the number of shared servers to start when the database starts. Setting **SHARED_SERVERS** to 0 or not setting a value at all disables the shared server feature – it will not be used. If you have configured any dispatchers through the **DISPATCHERS** parameter, the default setting for **SHARED_SERVERS** is 1.

The rule of thumb for **SHARED_SERVERS** is to start 1 shared server for every 25 concurrent connections. Start more if users will have large result sets or do intensive processing.

Here is an example: To start the shared server feature with 5 shared servers upon instance startup, issue:

```
SHARED_SERVERS = 5
```

You can dynamically start more shared servers up to the maximum specified by the initialization parameter **MAX_SHARED_SERVERS** by issuing this command:

```
ALTER  SYSTEM  SET  SHARED_SERVERS = 20 ;
```

*The number you specify in this command is the total number of shared servers you want -- **not** the number of shared servers to add.*

The initialization parameter **MAX_SHARED_SERVERS** specifies the maximum number of shared servers you can have running concurrently. You set it typically to reflect your highest expected workload. This parameter is also dynamic and so it can be altered by the **ALTER SYSTEM** command. If you do not set a value for **MAX_SHARED_SERVERS**, its effective upper limit is unlimited. (This is the default.)

Here is an example that initially sets **MAX_SHARED_SERVERS** to 3 as an initialization parameter, then dynamically alters it up to 15:

```
MAX_SHARED_SERVERS = 3
ALTER  SYSTEM  SET  MAX_SHARED_SERVERS = 15 ;
```

Whereas the initialization parameter **MAX_SHARED_SERVERS** specifies the maximum number of shared servers you can have running concurrently, the **SHARED_SERVER_SESSIONS** parameter specifies the total number of shared server sessions that are allowed for the instance. This parameter limits the total number of shared servers sessions. If the number of shared server client connections reaches this limit, the subsequent clients connection attempts receive this error message:

```
ORA-00018 maximum number of sessions exceeded
```

You can specify this parameter and dynamically alter it, as in this example that initially sets it to 10 by an init parm, and then dynamically alters it up to 20:

```
SHARED_SERVER_SESSIONS = 10
ALTER  SYSTEM  SET  SHARED_SERVER_SESSIONS = 20 ;
```

Another configuration issue for the shared server feature is how to configure the dispatchers. For a 10g instance that has the default network setting of a **TCP/IP** listener listening on default **port 1521**, Oracle now starts a dispatcher to listen for TCP/IP connection attempts on the default port. So in this case, you would not be required to configure the **DISPATCHERS** initialization parameter.

In most cases, you will want to configure the number of dispatchers for shared server. For each dispatcher, you configure a number of attributes, as shown by the chart below. The only required attribute is **PROTOCOL**.

Attribute:	Abbreviations:	Use:
ADDRESS	ADD or ADDR	The network address on which the dispatcher listens.
CONNECTIONS	CON or CONN	Maximum number of network connections per dispatcher.
DESCRIPTION	DES or DESC	The network description of the end point for the dispatcher.
DISPATCHERS	DIS or DISP	Number of dispatchers of this kind to start when the instance starts (default is 1).
LISTENER	LIS or LIST	Address of the listener (not needed for a local listener on the default port **1521**).
POOL	POO	Turns on Connection Pooling.
PROTOCOL	PRO or PROT	Network protocol.
SESSIONS	SES or SESS	Maximum number of sessions for this dispatcher.
SERVICE	SER or SERV	Oracle Net Service Name registered with the listener by the dispatcher (defaults to the **SERVICE_NAMES** parameter if not specified).
TICK	TICK	Use with **POOL** for Connection Pooling. Sets the number of 10-minute increments after which a connection will be considered idle.

Here is an example that starts 4 dispatchers for the TCP/IP protocol and 2 for the IPC protocol:

DISPATCHERS = "(PROTOCOL=TCP)(DISPATCHERS=4)(PROTOCOL=IPC)(DISPATCHERS=2)"

With abbreviations you might see this same setting as:

DISPATCHERS = "(PRO=TCP)(DIS=4)(PRO=IPC)(DIS=2)"

Using the alternate abbreviations, this same line could be coded as:

DISPATCHERS = "(PROT=TCP)(DISP=4)(PROT=IPC)(DISP=2)"

You would code this line in the initialization parameter file, or dynamically alter the number of dispatchers by using the **ALTER SYSTEM** command. This allows you to dynamically add or remove dispatchers. Set it to the total number of dispatchers you want on the system for the protocol(s) you specify, **not** to the number of dispatchers to add or remove.

This example adds 2 more dispatchers to a system already running 8. Therefore it specifies the total number of dispatchers for the protocol specified (which will be 10):

```
ALTER SYSTEM SET DISPATCHERS= "(PRO=TCP)(DIS=10)" ;
```

Note that **DISPATCHERS** always takes a quoted character string for its assignment.

How many dispatchers should you start? The rule of thumb is 50 concurrent sessions for each dispatcher. For situations involving data intensive operations or a large number of sessions, each dispatcher should handle fewer concurrent sessions. Oracle Corp uses this formula to determine how many dispatchers to initially configure:

```
Number of Dispatchers =
CEIL (maximum number of concurrent sessions / connections per dispatcher)
```

For example, for 100 concurrent TCP/IP connections and the standard rule of 50 concurrent sessions per dispatchers, you would configure 2 TCP/IP dispatchers. Set this up like this:

```
DISPATCHERS = "(PROT=TCP)(DISP=2)"
```

As the **SHARED_SERVERS** init parm has its **MAX_SHARED_SERVERS**, so does **DISPATCHERS** have an analogous **MAX_DISPATCHERS** parameter**. MAX_DISPATCHERS** sets the maximum number of dispatchers allowed. Here is an example that sets the maximum number of dispatchers at 10 in the initialization parm file:

```
MAX_DISPATCHERS = 10
```

As with all the parameters we've discussed so far, **MAX_DISPATCHERS** is dynamic. You can change it via the **ALTER SYSTEM** statement, as in this example:

```
ALTER SYSTEM SET MAX_DISPATCHERS = 25 ;
```

Recall that Connection Pooling allows Oracle shared server to handle a larger number of simultaneous connections. It does this by disconnecting idle connections in favor of new requests, then re-connecting the idle connection later when it becomes active again. You implement connection pooling by specifying the **POOL=ON** keywords on the **DISPATCHER** initialization parameter. You also need to code **TICK**. This parm defines the number of 10-minute increments after which a connection is considered idle.

This example turns on connection pooling. Any connection that is inactive for 10 minutes or more is considered idle and could temporarily be disconnected as connection pooling manages the connections:

```
DISPATCHERS =
"(PRO=TCP)(DIS=5)(POOL=ON)(TICK=1)(CON=500)(SES=1000)"
```

On rare occasions, you may want to configure the total number of virtual circuits for all incoming and outgoing network sessions. Set the **CIRCUITS** initialization parameter to do this, as in the following example:

```
CIRCUITS = 100
```

You can also use the **ALTER SYSTEM** parameter to dynamically modify this value:

```
ALTER SYSTEM SET CIRCUITS = 200 ;
```

Tuning Shared Server

There are several views to query for details of the shared server system:

View:	Contains:
V$SESSION	Info on clients currently connected and how many concurrent connections you have.
V$LICENSE	Current number of sessions and the maximum number of sessions since the instance started.
V$SHARED_SERVER	Shared server process information including how many requests and bytes of data they are processing.
V$SHARED_SERVER_MONITOR	Maximum number of shared servers started since the instance was started, other good info for tuning the number of shared servers.
V$DISPATCHER	Dispatcher activity, number of connections they are handling, total number of connections each dispatcher handled since startup. The columns **WAIT**, **IDLE**, and **BUSY** tell you whether dispatchers are waiting or whether they are busy.
V$DISPATCHER_CONFIG	Configuration information about dispatchers (this view is new in 10g).
V$DISPATCHER_RATE	Dispatcher statistics including maximum number of connections, bytes processed, etc.
V$QUEUE	Information on the request and response queues.
V$CIRCUIT	Virtual circuit information.
V$SGASTAT	Information on the SGA's use of memory and the size of the Large Pool.
V$SESSTAT	How much memory shared server sessions use.

Another good source of information is this line command:**lsnrctl services**
This command shows you:

- Process IDs for the dispatchers
- How many connections each dispatcher manages
- Listening address for the dispatchers
- How many client connections have been established
- How many client connections have been rejected (connections are rejected when users supply invalid usernames or passwords, and when the **MAX_SHARED_SERVERS** limit is reached)

To tune the shared server option:

1. Ensure you have the right number of shared servers
2. Ensure you have the right number of dispatchers
3. Size the Large Pool correctly

For shared servers, you should have enough to service user requests without having any client requests waiting. But you do not want too many shared servers, as this wastes resources as some sit idle or under-utilized.

The **V$SHARED_SERVER** and **V$QUEUE** views are key. **V$QUEUE** tells you how long client requests are waiting in hundredths of seconds.

For dispatchers, you should have enough so that users are not waiting on dispatchers, but not so many that some are under-utilized. The views **V$QUEUE** and **V$DISPATCHER** allow you to see how long users are waiting (on average) for dispatchers.

For the Large Pool, remember that this memory area contains data for user sessions that used to be in the PGA (for dedicated connections). For the shared server feature, the large pool includes the User Global Areas or UGAs. If you do not configure a large pool, Oracle takes memory from the SGA for this purpose. Therefore, configure a big enough large pool to handle the increased memory requirements for the shared server feature.

Check view **V$SGASTAT** to see the sizes of the SGA and large pool. The **free memory** row shows how much memory is available in the large pool, while the **session heap** row shows how much memory is used in the large pool. Ensure there is enough free memory.

Configure the large pool through the **LARGE_POOL_SIZE** initialization parameter. It is dynamic, so you can change it via the **ALTER SYSTEM** command.

You can initially size the large pool by multiplying the number of shared server connections by the maximum UGA memory per session. For example, if **V$SESSTAT** shows clients are using an average of 250 K of UGA memory, and you expect 100 concurrent connections, the initial large pool should be at least 25 M (250 K * 100).

10g Shared Server Improvements

10g is a big advance over previous releases of Oracle database and its shared server feature in that all these shared server parameters are now dynamic, making 10g's shared server configuration much more flexible than in the past:

- SHARED_SERVERS
- MAX_SHARED_SERVERS
- DISPATCHERS
- MAX_DISPATCHERS
- CIRCUITS

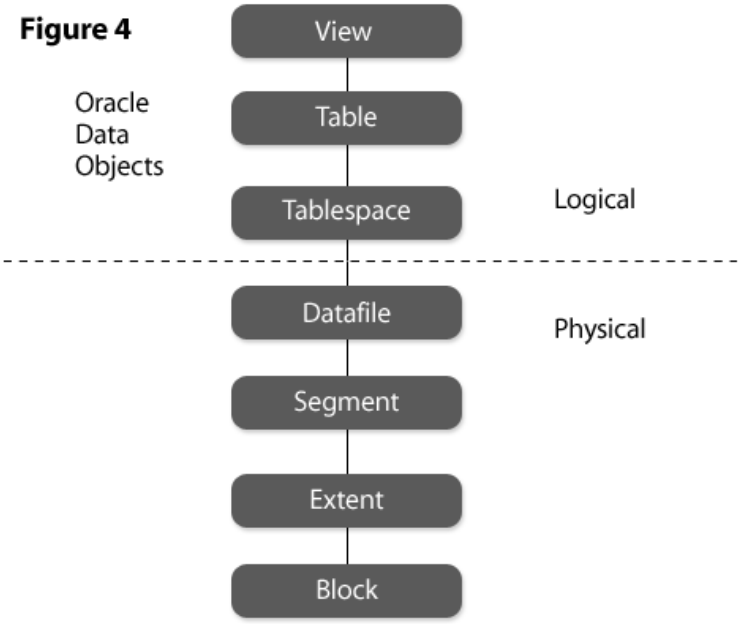
With 10g, you rarely have to configure **CIRCUITS** and virtual circuits like you did in previous releases. Also, if you're using the default network configuration (a local listener using TCP/IP protocol on the default port 1521), Oracle automatically starts one dispatcher for you.

10g obsoletes many older shared server configuration parameters including these:

Obsolete Initialization Parameter:	**New 10g Equivalent:**
MTS_SERVERS	SHARED_SERVERS
MTS_MAX_SERVERS	MAX_SHARED_SERVERS
MTS_DISPATCHERS	DISPATCHERS
MTS_MAX_DISPATCHERS	MAX_DISPATCHERS
MTS_SERVICE	SERVICE_NAMES
MTS_SESSIONS	SHARED_SERVER_SESSIONS
MTS_LISTENER_ADDRESS	LOCAL_LISTENER
MTS_MULTIPLE_LISTENERS	LOCAL_LISTENER
MTS_CIRCUITS	CIRCUITS

Oracle Data Objects

Figure 4 shows Oracle's data objects and the relationships between them. You need to know the basics about these objects, how to create and alter them, and how they are used.



Views

Views are "virtual tables." They display data in terms of rows and columns, just like tables. But views are based on one or more underlying tables, called base tables. Views do not store data themselves. They instead consist of a query that represents data from their one or more underlying base tables.
Views are used for two primary reasons:

1. To simplify how data is presented to the user. For example, a view can contain a complicated join of two or more tables, or include computed or derived column values. The view simplifies the query the user creates to retrieve this information.
2. To enforce security. Views may present only certain columns or rows to the user. Giving out access to views instead of the underlying base tables therefore enforces security.

Create a view with the **CREATE VIEW** statement. Typically you'll use **CREATE OR REPLACE VIEW**, since if you use **CREATE VIEW** and the view already exists, you'll get an error. **CREATE OR REPLACE VIEW** creates a view if it is new, or creates a view and replaces its previous incarnation if a view with the given name already exists.

Here's an example of creating a view based on a single base table:

```
CREATE OR REPLACE VIEW my_view
        (emp, dept, mgr)
AS SELECT employee, department, manager FROM emp_table
        WHERE mgr_level > 2 ;
```

Drop a view with the **DROP VIEW** statement:

```
DROP VIEW my_view ;
```

You can refer to multiple base tables in creating a view (via a *join predicate*). Some views can be used to insert, update, and delete data. You can insert into a view if its definition does *not* contain:

- The **DISTINCT** operator
- A set operator (such as **UNION** or **MINUS**)
- A **GROUP BY**, **ORDER BY**, START WITH, or **CONNECT BY**
- An aggregate function (such as **SUM** or **AVG**)
- A subquery in the **SELECT** list
- A collection expression in a SELECT list
- Joins (with some exceptions)

Tables

Tables and views are both schema objects; they belong to specific schemas, the collection of objects associated with a specific user id. Certain objects -- like tablespaces, roles, and directories -- are not associated with specific user ids and so are not considered schema objects.

Tables are the primary data storage objects in Oracle. Users create tables as objects with a specific list of associated columns. Columns have datatypes (which require that they hold certain kinds of data, for example, numeric or character). Columns may have constraints (further rules about the characteristics for the data in the column) and defaults (values inserted in the absence of any other).

Table and column names must:

- Begin with a letter
- Be from 1 to 30 characters in length
- Can include letters, numbers, underscores (_), dollar signs ($), and pound signs (#).
- Cannot be an Oracle reserved word (for example, the keyword **TABLE**)

Enclose a name within double quotes ("), it must be from 1 to 30 characters long and not contain an embedded double quote mark. Table names must be unique within the namespace (schema) and column names must be unique within the table.

Create a table by the **CREATE TABLE** statement:

```
CREATE TABLE my_table
        (first_column                NUMBER,
        second_column    VARCHAR2(221)  ) ;
```

View table structure and definitional attributes by the **DESCRIBE** command:

```
DESCRIBE my_table ;
```

View table contents by a **SELECT** statement. For example, view all the contents by an unqualified **SELECT** statement (a **SELECT** without a **WHERE** clause):

```
SELECT * FROM my_table ;
```

Alternatively, create a table by the **CREATE TABLE AS SELECT** statement. This specifies a **SELECT** statement to choose the rows and columns that comprise the new table. Here's an example:

```
CREATE TABLE my_table
        NOLOGGING COMPRESS
        TABLESPACE my_tablespace
AS SELECT * FROM my_older_table
        WHERE first_column > 100 ;
```

NOLOGGING turns off the logging of redo records for the table, **COMPRESS** automatically compresses the data as stored on disk, and **TABLESPACE** specifies in which tablespace the table will reside.

Add a comment to either a table or column of up to 4,000 characters to describe it. Comments can include embedded white space and punctuation and span physical lines. Here are examples:

```
COMMENT ON TABLE my_table IS
'This table is just an example, delete it later in the job stream, we do not need it for very long' ;
COMMENT ON COLUMN first_column IS
        'This column contains the id' ;
```

View table comments in **DBA_TAB_COMMENTS** and column comments in **DBA_COL_COMMENTS**. Remember you can also use the high-level qualifiers **USER_** and **ALL_** . For example, **ALL_TAB_COMMENTS** and **USER_TAB_COMMENTS**.

Rename a table (or a view, sequence, or private synonym) through the **RENAME** statement. Renaming a table invalidates all objects that depend on it, such as views, synonyms, and procedures. Oracle automatically updates any associated indexes, grants, and constraints to refer to the new name. Here's an example **RENAME**:

```
RENAME my_table TO my_new_table ;
```

You can alternatively use **ALTER TABLE** for renaming tables:

```
ALTER TABLE my_table RENAME TO my_new_table ;
```

Each column in a table definition may have several attributes including **datatypes**, **default values**, and **constraints**.

Oracle 10g recognizes 6 major categories of datatypes:

- Character - Alphanumeric or Unicode data
- Numeric - Positive and negative fixed and floating point numbers, plus the special value **Not A Number**
- Datetime - Stores dates, times, and time periods
- LOB - Stores LOBs or Large Objects, such as video, text, image, audio, and spatial data
- ROWID - Uniquely identifies rows within tables
- Binary - Stores binary data (unstructured, unconverted data)

Here are 10g's datatypes for column definitions:

==CHARACTER=	
CHAR(n)	Fixed-length character with length specification.
VARCHAR2(n)	Variable-length character string. You **must** specify n.
VARCHAR(n)	Same as VARCHAR2 but Oracle recommends using VARCHAR2 instead of VARCHAR.
NCHAR(n)	Unicode version of CHAR.
NVARCHAR2(n)	Unicode version of VARCHAR2.
LONG	Obsolete, use CLOB. Variable-length alphanumeric string to 2G.
==NUMERIC==	
NUMBER(p,s)	Numbers (fixed or floating point). P is Precision (the total number of digits, which defaults to 38). S is Scale (the number of digits to the right of the decimal place which defaults to 0).
BINARY_FLOAT	Single-precision floating point (new in 10g).
BINARY_DOUBLE	Double-precision floating point (new in 10g).
==DATETIME==	
DATE	Date with century down to seconds.
TIMESTAMP(p)	Date with time down to fractional seconds.
TIMESTAMP(p) WITH TIME ZONE	**TIMESTAMP** with time zone displacement.
TIMESTAMP(p) WITH LOCAL TIME ZONE	**TIMESTAMP** with normalized local time.

INTERVAL YEAR TO MONTH	An interval time period with years down to months.
INTERVAL DAY TO SECOND	An interval time period with days down to seconds.
==LOB==	
BLOB	Binary variable-length data.
CLOB	Character variable-length data.
NCLOB	Unicode character variable-length data.
BFILE	Points to binary data stored in a flat file outside of Oracle, to 4 G.
==ROWID==	
ROWID	Uniquely identifies a row within a table. Refer to it by the **ROWID** pseudo-column in SQL. ROWIDs include: Object ID (OID), relative file number, block number, and slot in the block.
UROWID	Universal ROWID stores a logical row address.
==BINARY==	
RAW(s)	Unstructured data up to 2,000 bytes.
LONG RAW	Unstructured data up to 2 G. Obsolete, use **BLOB** instead.

Add one or more columns to an existing table, or drop columns, via the **ALTER TABLE** statement. Here's how to add a column:

```
ALTER TABLE my_table
        ADD  new_column  VARCHAR2(221) ;
```

You name the column, specify its datatype, and also optionally its default value. Here's how to add multiple columns to an existing table:

```
ALTER  TABLE  my_table
        ADD (new_col_one  VARCHAR2(221),
             new_col_two  NUMERIC ) ;
```

This example drops one column from a table:

```
ALTER TABLE my_table  DROP COLUMN new_column ;
```

This example drops multiple columns from an existing table. Note that it excludes the **COLUMN** keyword:

```
ALTER TABLE  my_table DROP (new_col_one, new_col_two) ;
```

You can modify one or more columns using the **ALTER TABLE … MODIFY** statement. The attributes you can modify are:

- Rename a column
- Increase or decrease column size
- Assign a default value to a column

This example modifies one column to have a new length specification:

```
ALTER TABLE my_table MODIFY new_col_one VARCHAR2(255) ;
```

This example modifies two columns, including the default value on one of them:

```
ALTER TABLE my_table_2  MODIFY
            (col_one  VARCHAR2(255) ,
             col_two  VARCHAR2(64)  DEFAULT  USER  ) ;
```

Default values are values Oracle assigns to a column when an **INSERT** statement does not provide a value to populate that column. Default values can be constants or derived from SQL expressions. You can explicitly override a default value assignment in an **INSERT** or **UPDATE** statement by setting the column using keyword **NULL** (if allowed by the table definition). An alternative way to assign default values to columns is by using database triggers (triggers will be discussed later).

Tables can be permanent (the default) or temporary. Permanent tables are regular tables whose data is permanently stored on disk, while temporary tables are transient and last only for the duration of the *transaction* or the *duration of the session*. Temporary tables can only be viewed by the session creating and updating them. This example creates a temporary table:

```
CREATE  GLOBAL  TEMPORARY  TABLE mine_for_the_txn
            (which_one                  NUMBER,
             contents             VARCHAR2(221)  )
ON  COMMIT  DELETE  ROWS ;
```

This example temporary table exists for the duration of the transaction (**ON COMMIT DELETE ROWS**). The alternative is a "temp table" that lasts for the duration of the session **(ON COMMIT PRESERVE ROWS).**

Constraints

Constraints are rules or restrictions that apply to tables and their columns. They encode business rules in the database (instead of in program logic) for simplicity, accuracy, universal application, and data integrity. Constraints define the values that can acceptably be placed into columns.

The 5 constraint types are:

Constraint Keyword:	Use:
NOT NULL	Requires every row to contain some value for the column. By default, all columns are allowed to contain nulls (defined as the absence of any data value. Note that nulls are different from values like 0 or blanks).
UNIQUE	Requires that every value in the column be unique. **NULL** values do not count as unique, so a **UNIQUE** column could have multiple rows with **NULL** values. Oracle automatically creates an index to enforce uniqueness if necessary.
PRIMARY KEY	1 allowed per table to uniquely identify rows. Oracle automatically creates the index necessary to support this. This implies the **UNIQUE** and **NOT NULL** attributes.
CHECK	Verifies that a column meets the condition described by some expression that evaluates to a boolean condition (**TRUE** or **FALSE**). Only rows with **CHECK** columns that evaluate to **TRUE** are inserted, others are rejected with error.
FOREIGN KEY	Used to enforce Referential Integrity (RI) back to the unique or primary key referenced in the parent table.

Referential integrity (or **FOREIGN KEY**) constraints "link" two tables together to ensure that any data values entered into one table are already defined in another table. They can be used, for example, to link together tables into a parent-child relationship, based on the referential column values. The table with the primary or unique key is the parent table, and the table with the **FOREIGN KEY** constraint is the child or dependent table. **FOREIGN KEY** constraints cannot be defined on columns of datatype **CLOB**, **NCLOB**, **BLOB**, **LONG**, **LONG RAW**, or **TIMESTAMP WITH TIMEZONE** (this also applies to columns with **PRIMARY KEY** and **UNIQUE** constraints).

By default, the database does not allow deleting rows from a parent table that still has dependent rows in a child table. Alternatives are to *cascade deletes*, that is, to automatically delete all dependent child rows when deleting a parent row. The other alternative is to set relevant columns in the dependent child table to nulls.

To cascade deletes, use the clause **ON DELETE CASCADE** when defining the foreign key column. To set deleted dependents to nulls, use the clause **ON DELETE SET NULL** for the foreign key column.

Self-referencing foreign keys or self-referencing tables occur when the parent and dependent rows exist in the same table. A classic example is an EMPLOYEES table, where a hierarchical relationship exists among the employees in that table. The MANAGER relationship could trace a hierarchical relationship among employees and their managers by means of a self-referential relationship within the EMPLOYEES table.

You can use **ALTER TABLE** to add or drop constraints on existing tables. Examples:

```
ALTER TABLE tab ADD (CONSTRAINT cname PRIMARY KEY (col1));
ALTER TABLE tab ADD (CONSTRAINT cname UNIQUE (col1));
ALTER TABLE tab ADD (CONSTRAINT cname CHECK (col3=col4));
ALTER TABLE tab MODIFY (col1 NOT NULL);
ALTER TABLE tab DROP CONSTRAINT cname ;
ALTER TABLE tab DROP PRIMARY KEY ;
```

Constraints may also be enabled (activated) or disabled (turned off) by use of the **ALTER TABLE** statement with the **ENABLE** or **DISABLE** keywords. Example:

```
ALTER TABLE  tab  DISABLE CONSTRAINT  cname ;
```

Constraints may also be deferred. This means that instead of being satisfied at the end of each statement (as per the default), the constraint can be satisfied later (such as by the end of the transaction).

The dictionary views **DBA_CONSTRAINTS** and **DBA_CONS_COLUMNS** contain information about constraints and the constraint columns. Look in **DBA_TABLES** and **DBA_TAB_COLUMNS** for table information and **DBA_VIEWS** for view information.

Tablespaces

Data is stored *logically* in tablespaces and *physically* in datafiles. Each tablespace consists of 1 or more datafiles; each datafile belongs to 1 tablespace only. Tablespaces can be used as the unit for backup/recovery, space quotas, read-only data, online/offline status, and to store data in particular locations on disk to enhance I/O performance.

When you create a table it is associated with a tablespace. In the **CREATE TABLE** statement, the **TABLESPACE** keyword dictates in which tablespace a table resides. If that keyword is omitted, the table will reside in the default tablespace for that user.

By default, 10g databases have at least 3 tablespaces. Most have many more. The mandatory 3 are in **boldface** below, plus we list 3 other tablespaces you'll typically see:

Tablespace:	**Use:**
SYSTEM	Stores the data dictionary (owned by user **SYS**), and PL/SQL code.
SYSAUX	Stores data used by various database options.
TEMP	For sorting. Required only if **SYSTEM** is a locally managed tablespace.
UNDOTBS1 or UNDO	Stores undo records, permitting read consistency and recovery.
USERS	The default tablespaces for users.
TOOLS	Stores data for various management tools.

10g includes Oracle Managed Files (OMF) to make working with tablespaces and their underlying datafiles simple. With OMF you only manage tablespaces; Oracle manages the underlying datafiles and temp files for you.

To enable OMF, set initialization parameter **DB_CREATE_FILE_DEST** to the directory where Oracle should create and manage your files:

```
ALTER SYSTEM SET
        DB_CREATE_FILE_DEST = 'C:\oracle\oradata\omf'  SCOPE=BOTH ;
```

When creating an OMF tablespace, you just specify the tablespace name:

```
CREATE  TABLESPACE  my_ts ;
```

OMF creates and names the underlying datafile. By default it will be 100 M in size and have autoextend on (this permits Oracle to automatically extend the datafile if it needs space).

Just as OMF automatically creates the underlying OS datafile for you, it also deletes it for you if you later drop the tablespace:

```
DROP  TABLESPACE  my_ts ;
```

10g supports two kinds of tablespaces: **SMALLFILE** and **BIGFILE**. Smallfiles are the default and were the only kind of tablespaces prior to 10g. You create smallfile tablespaces by default unless you specify the keyword **BIGFILE**.

Bigfile tablespaces consist of 1 -- and only 1 -- underlying datafile. They are intended for large databases with lots of read/write activity in environments that do not have an underlying volume manager tool to do dynamic volume resizing, striping, and mirroring.

This example creates a bigfile tablespace:

```
CREATE  BIGFILE TABLESPACE  my_ts
DATAFILE '/oradata/prod/bigds03.dbf'  SIZE 2 G ;
```

Omit the **BIGFILE** keyword and you've created a smallfile tablespace. Bigfiles can get up to 2 to the 32nd power blocks in size, so their maximum size depends on the block size you use. Bigfiles could be up to 32 T with an 8 K block size, for example. Smallfiles get up to 2 to the 22nd power per datafile, and may have up to 1,022 datafiles per tablespace, resulting in a maximum of almost 32 T for an 8 K block size.

An extent is a group of contiguous blocks on the storage media (discussed in detail later). Oracle manages tablespace extents either by local extent management or dictionary extent management. Local extent management is more efficient because Oracle does not have to access the data dictionary to manage the tablespace's extents.

Extent Management:	Use:	Keywords:
Local Extent Management	**Recommended**. Manages extents through bitmaps. **Default.**	EXTENT MANAGEMENT LOCAL
Dictionary Extent Management	Older approach, less efficient. Uses the data dictionary to manage extents.	EXTENT MANAGEMENT DICTIONARY

Segments are a higher-level physical storage object than extents. Segments are made up of extents. Oracle manages tablespace segments either by automatic segment space management or manual segment space management. Automatic segment space management is newer and recommended. It uses bitmaps to keep track of usable data blocks, while the older manual segment space management uses free block lists. Unfortunately, the default is manual segment space management, so when creating a tablespace be sure to use keywords **SEGMENT SPACE MANAGEMENT AUTO** to get the recommended setting."

Segment Management:	Use:	Keywords:
Automatic Segment Space Management	**Recommended**. Manages free blocks through bitmaps.	SEGMENT SPACE MANAGEMENT AUTO
Manual Segment Space Management	Older approach, less efficient. Manages free blocks through free lists. **Default**.	SEGMENT SPACE MANAGEMENT MANUAL

Tablespace Examples -- Create a tablespace with multiple datafiles, specifying their sizes and locations. Defaults mean this will be a locally-managed tablespace (**EXTENT MANAGEMENT LOCAL**) but with manual free-list space management within the segments (**SEGMENT SPACE MANAGEMENT MANUAL**) rather than bit-mapped:

```
CREATE TABLESPACE my_ts DATAFILE '/ora/my_df01.dbf' SIZE 500M,
'/ora/my_df02.dbf' SIZE 300M ;
```

A similar tablespace but with bit-mapped space management for the segments:

```
CREATE TABLESPACE my_ts DATAFILE '/ora/my_df01.dbf' SIZE 500M EXTENT MANAGEMENT LO-
CAL    SEGMENT SPACE MANAGEMENT AUTO ;
```

Change tablespace attributes by **ALTER TABLESPACE**. Clauses in this statement are mutually exclusive, you can only change 1 parm at a time.

Every tablespace can be either **ONLINE** or **OFFLINE**. Change status by **ALTER TABLESPACE**. Examples:

```
ALTER TABLESPACE my_tablespace OFFLINE ;
ALTER TABLESPACE my_tablespace ONLINE ;
```

You can also make tablespaces read-write (the default) or read-only. To make a tablespace read-only:

```
ALTER TABLESPACE  my_ts  READ  ONLY;
```

Flip it back to read-write:

```
ALTER TABLESPACE my_ts READ WRITE;
```

Remember to change your backup strategy for tablespaces when you change their read/write status!

Coalesce adjacent free extents for efficiency by:

```
ALTER TABLESPACE my_ts COALESCE ;
```

You can increase or decrease datafile size:

```
ALTER DATABASE DATAFILE '/ora/my_df01.dbf' RESIZE 200M ;
```

Drop a tablespace by:

```
DROP TABLESPACE my_ts ;
```

If it is not empty, use the keywords **INCLUDING CONTENTS**:

```
DROP TABLESPACE my_ts INCLUDING CONTENTS;
```

If it has constraints:

```
DROP TABLESPACE my_ts INCLUDING CONTENTS CASCADE CONSTRAINTS ;
```

Non-OMF files can be deleted automatically by Oracle by specifying the INCLUDE CONTENTS AND DATA-FILES clause. If it is OMF, Oracle automatically deletes the underlying OS files for you.

To rename a datafile, take its tablespace OFFLINE first, rename the datafile, then put the tablespace back ONLINE:

```
ALTER TABLESPACE my_ts OFFLINE;
ALTER DATABASE RENAME FILE '/ora/mdf01.dbf' TO '/ora/new_name.dbf' ;
    ALTER TABLESPACE my_ts ONLINE;
```

To rename or relocate datafiles for multiple tablespaces, it may be easier to shutdown the database prior to the activity (instead of taking all the files offline). You **must** shutdown the database when tampering with any datafiles underlying the **SYSTEM** tablespace. Oracle requires a **SYSTEM** tablespace and its Data Dictionary to operate, and you cannot rename either the **SYSTEM** or **SYSAUX** tablespaces.

Dictionary views **DBA_TABLESPACES** and **V$TABLESPACE** show all the basic information on tablespaces. Temporary tablespace info is in **DBA_TEMP_FILES**.

Datafiles

Tablespaces can further be managed at the datafile level. Every tablespace consists of 1 or more datafiles and every datafile belongs solely to 1 tablespace. Operations you can perform on datafiles include:

- Taking them offline or back online
- Resizing them
- Moving or renaming them
- Recovering them

Here are the steps in the procedure to move or rename a datafile when working on the datafile level. It illustrates several of the above operations:

1. Take the datafile offline:

   ```
   ALTER DATABASE DATAFILE 'C:\ORADATA\DATA05.DBF' OFFLINE ;
   ```

2. Copy the datafile using the operating system's "copy" or "cp" command:

   ```
   COPY C:\ORADATA\DATA05.DBF C:\ORA\DATA10.DBF
   ```

3. Change the name of the datafile in the control file:

   ```
   ALTER DATABASE RENAME FILE 'C:\ORADATA\DATA05.DBF' TO
       'C:\ORA\DATA10.DBF' ;
   ```

4. Ensure the datafile header is synchronized with that in the database:

   ```
   RECOVER DATAFILE 'C:\ORA\DATA10.DBF' ;
   ```

5. Bring the datafile back online for use:

   ```
   ALTER DATABASE DATAFILE 'C:\ORA\DATA10.DBF' ONLINE ;
   ```

Dictionary view **DBA_DATA_FILES** shows datafile information and which tablespace each is associated with. **DBA_TEMP_FILES** contains information on temp datafiles.

Segments

A segment is a unit of physical storage in the database. Each segment consists of chunks of storage called extents. A segment must have 1 extent and may have up to 2 billion.

Every segment is of a specific segment type defined by its use in the database and the kind of data it holds. There are over a dozen segment types in Oracle, here are a few common ones:

Segment Type:	Use:
Table	Stores table data.
Index	Stores an index: a data structure for more rapid access to table data and for better query performance.
Table Partition	Partitioned tables can be stored in separate tablespaces (this technique better manages very large tables) and use their own segment type.
Index Partition	For partitioned indexes on partitioned tables.
Undo (or Rollback)	Undo records are used to rollback a data change and are stored in their own segment type.

Extents and Blocks

Segments consist of between 1 and 2 billion extents. An extent is a group of contiguous blocks on the storage media. A block is the smallest unit of storage. Sometimes a block is called a data block or a page. An extent consists of a minimum of 5 blocks. When you create a database, you specify its default block size by the initialization parameter **DB_BLOCK_SIZE**. But you can also define tablespaces with alternate block sizes by specifying the block size on the **CREATE TABLESPACE** statement. Allowable block sizes are 2 K, 4 K, 8 K, and 16 K. Some operating systems further support block sizes of 32 K and 64 K. A database allows up to 5 different block sizes.

The required tablespaces **SYSTEM** and **SYSAUX** will always be of the default block size for the database (sometimes called the database's standard block size).

If you created the database to use operating system **filesystems**, then the Oracle block size should be a multiple of the operating system block size (sometimes called its page size). If you created the database using **raw devices**, you've assigned a chunk of storage (either a disk or disk partition) over to Oracle's use. Oracle does I/O to these raw devices without going through the operating system's filesystem or disk access method.

Indexes

Indexes provide faster data access, avoid sorts, and enforce uniqueness and primary key constraints. You build indexes on base tables. There are two basic kinds of indexes:

Index Type:	Use:
==Bitmap==	Uses a bitmap index for low cardinality data (data with few distinct values, e.g. "Gender").
==B-tree Indexes==	The default. Use for high cardinality data (data with many distinct values).
Unique	Each entry in the index must be unique.
Non-unique	Allows duplicate entries in the index.
Reverse Key	Reverses the bytes of the indexed column. For example: **ABC** will be stored as **CBA**.
Function-based	The index is defined on results of a function. For example, on this: SUBSTR(column_a, 1, 4).

Bitmap indexes are best for low-cardinality data, columns that have few different values (for example: gender). They update inefficiently, so use them for data warehouses and retrieval-only systems.

B-tree indexes are the default type of index. Each entry in the index tree contains the key (consisting of one or more columns from the table) and a rowid or pointer into the database where the associated data row exists. B-tree indexes work well for high-cardinality columns and transactional systems where concurrent updating is required. The chart above lists the 4 kinds of B-tree indexes.

Index Examples -- Create a Unique index:

```
CREATE UNIQUE INDEX ix_name ON my_tab (col_name);
```

Create a Bitmap index:

```
CREATE BITMAP INDEX ix_bitmap ON my_tab (col_name);
```

Create a Reverse Key index:

```
CREATE UNIQUE INDEX ixr ON my_tab(col) TABLESPACE tsx REVERSE;
```

Create a Function-based index:

```
CREATE INDEX ixf ON my_tab (SUBSTR(cola, 1, 4)) TABLESPACE ts_ix;
```

Decrease index fragmentation while others still use the underlying table data:

```
ALTER INDEX ix_name COALESCE ;
```

Or alternatively, you can rebuild the index, with the **ONLINE** clause so others can use it while you do (Oracle accomplishes this by creating the new index and only dropping the original after it is completed):

```
ALTER INDEX ix_1 REBUILD TABLESPACE new_ix_ts ONLINE NOLOGGING ;
```

While you alter indexes via the **ALTER INDEX** statement, drop them like this:

```
DROP INDEX ix_name ;
```

You can only drop primary key or uniqueness constraint indexes *after* disabling the constraint they support.

Sequences

Sequences are schema objects that automatically generate unique integers. They are typically applied to certain columns in tables to generate unique identifiers, for example, for primary keys. An example would be when you need to generate a unique ID for each new row you insert into a table.

The **CREATE SEQUENCE** statement creates a new sequence. Its major keywords:

CREATE SEQUENCE keyword:	**Use:**
START WITH	Starting value for the sequence.
INCREMENT BY	How much to increment each sequence value. Default is 1. A negative value results in negative increments.
MAXVALUE	Highest value the sequence can generate. Default is **NOMAXVALUE**.
MINVALUE	Lowest value the sequence can generate. Default is **NOMINVALUE**.
CACHE	For performance, tells how many values to hold in memory. Default is 20.
CYCLE	Specifies that the sequence "starts over" after reaching its maximum value for an ascending sequence (or minimum value for a descending sequence).

To reset the next value a sequence generates, you must drop and re-create it. Then re-grant any privileges on it. Here's an example drop:

```
DROP  SEQUENCE  my_sequence ;
```

This example creates a sequence that creates unique integers. The values it generates start with 1, increment by 1, and will go up to the highest possible maximum value (which is 10 to the 27th power):

```
CREATE  SEQUENCE  my_sequence  START WITH 1  NOMAXVALUE ;
```

To get the next value in the sequence from within a program, you might issue this statement with the **NEXTVAL** keyword (or pseudo-column):

```
SELECT  my_sequence.nextval  FROM dual ;
```

The **ALTER SEQUENCE** statement allows you to change:

- INCREMENT BY
- MAXVALUE
- MINVALUE
- CYCLE
- CACHE

You cannot change the **START WITH** value -- instead, drop and re-create the sequence.

Managing Data

You need to know the basics about Data Manipulation Language (DML), including **SELECT**, **INSERT**, **UPDATE**, and **DELETE** statements. Plus know a bit about PL/SQL, triggers, Data Pump, and SQL*Loader.

SELECT Statements

SELECT returns **unordered** rows called the result set. A minimal **SELECT** statement has the keyword **SELECT**, an expression, and a **FROM** clause. Some **SELECT** statement keywords:

DISTINCT -- returns only non-duplicate rows, versus the alternative **ALL** (default), which will return duplicates.

WHERE -- restricts result set based on selection criteria. A **SELECT** without a **WHERE** clause is an unrestricted **SELECT** and returns all rows in the table (or view). A join (combination of rows from two or more tables) without a **WHERE** clause results in a Cartesian product: all combinations of all rows (a very expensive query, usually only issued in error). Use a **WHERE** clause for Oracle to use an index (with some minor exceptions).

ORDER BY -- required if you want ordering of the returned rows. **ORDER BY** defaults to ascending (**ASC**) order, or specify descending (**DESC**). If more than 1 column (or alias or column positional indicator) is listed, rows are sorted by the sort specifications in left to right order. **ORDER BY** must be coded *after* **WHERE** and **GROUP BY**.

GROUP BY -- groups result set rows by the specified column's values. Only groups rows (does not restrict them). **GROUP BY CUBE** gives cross-tabulation and **GROUP BY ROLLUP** gives automatic subtotals.

HAVING -- limits **GROUP BY** to groups for which the condition is true (like a **WHERE** clause for the groups of **GROUP BY** queries). **HAVING** is only used with **GROUP BY**.

When you have several of these clauses in the same **SELECT** statement, the order in which they occur is: **WHERE**, **GROUP BY**, **HAVING**, **ORDER BY**.

INSERT Statements

Use an **INSERT** statement with a **VALUES** clause to insert a row into a table. You only need to specify a column list (field names) if you do not supply a value for every column.

You may use an **INSERT** with a subquery to insert multiple rows. Subqueries must return rows that match the number of columns in the target table's column list and have compatible (i.e., convertible) datatypes:

```
INSERT INTO emp SELECT * FROM human_tab ;
```

When inserting into a view, the view should have been defined with the **WITH CHECK OPTION** to ensure inserted rows must meet the criteria of the view's defining query.

Use the **RETURNING** clause to return inserted values to variables inside of programs. Use **PARTITION** to insert data into a specific table partition. Both clauses also apply to the **UPDATE** statement.

You can insert data rows into multiple tables in one pass (one read) of the data by using **WHEN** conditions. When a **WHEN** condition is met, the row is inserted into the table named in the **WHEN** clause. Use keyword **FIRST** to insert only into the first table for which the **WHEN** condition is true, or use keyword **ALL** to check every **WHEN** condition.

UPDATE and DELETE

UPDATEs or **DELETE**s without a **WHERE** clause update or delete all rows in the table (called "unqualified" **UPDATE**s or **DELETE**s). Unqualified **UPDATE**s and **DELETE**s are very powerful and should not be issued in error!

You can use subqueries in **UPDATE**s and **DELETE**s just like you do with **SELECT**s and **INSERT**s. Here's an example:

```
DELETE FROM employee_table
        WHERE empno IN (SELECT empno FROM bad_employee_table) ;
```

Update multiple columns in a table in one **UPDATE** statement using this syntax:

```
UPDATE employee_table
        SET (badge_no, stat) = (SELECT badge_no, stat
                                FROM bad_employee_table
                                WHERE manager = 'Fillbert')
        WHERE manager = 'Fillbert' ;
```

PL/SQL

PL/SQL is a super-set of SQL. PL/SQL's additions to SQL include:

- Conditional execution
- Looping
- Exception-handling (error handling)

The 5 kinds of PL/SQL objects are:

PL/SQL Object:	Use:
Anonymous Block	PL/SQL code that is not stored in the database but is instead embedded in a script, form or web page.
Procedure	A named block of PL/SQL code stored in the database.
Function	A named block of PL/SQL code stored in the database that returns a single value via its **RETURN** statement.
Package	A unit or set of related procedures, functions, and data structures (like records, cursors, constants, and variables). The package spec uses keyword **PACKAGE** to identify and name the package, while the package body contains the procedures, functions, etc., and is identified by keywords **PACKAGE BODY**. Packages are useful for collecting related objects together and for issuing one **EXECUTE** grant for use of all items in the package.
Trigger	PL/SQL code that runs whenever certain DML or DDL executes, or database events occur.

Use the **CREATE OR REPLACE** clause when creating procedures, functions, packages, and triggers. This replaces the object if it already exists with the updated code and avoids having to re-create the privileges that would be lost if you instead did a **DROP** and then a **CREATE**.

You can manage PL/SQL by these 4 initialization parameters (all are dynamic and can be set through the **ALTER SYSTEM** statement):

Parameter:	Use:
PLSQL_WARNING	Enables/disables PL/SQL compiler warnings.
PLSQL_DEBUG	Directs the PL/SQL library units to be compiled for debugging if **TRUE**. Default is **FALSE**.
PLSQL_OPTIMIZE_LEVEL	Tells how much time the optimizer should spend optimizing. Valid settings are 1 and 2, with 2 being the highest optimization. Default is 2.
PLSQL_CODE_TYPE	Specifies if code is **NATIVE** or **INTERPRETED**. **NATIVE** runs faster, **INTERPRETED** is the default.

Triggers -- Triggers execute when certain events occur in the database. There are 3 kinds of triggers:

1. DML triggers
2. DDL triggers
3. Database triggers

DML triggers fire when **INSERT**, **UPDATE**, or **DELETE** statements execute against the specified table. They run either:

- Once per SQL statement - The default
- Once per row - Use keywords: **FOR EACH ROW**

Multiple triggers on a table will fire in this order:

1. Before-statement triggers
2. Before-row triggers
3. After-row triggers
4. After-statement triggers

DDL triggers fire when any of these 15 or so different DDL commands execute: **ALTER**, **ANALYZE**, **ASSOCIATE STATISTICS**, **AUDIT**, **COMMENT**, **CREATE**, **DISASSOCIATE STATISTICS**, **DROP**, **GRANT**, **NOAUDIT**, **RENAME**, **REVOKE**, **TRUNCATE**.

Like DML triggers, DDL triggers can execute **BEFORE** or **AFTER** the event.

Database triggers fire when a specified database event occurs. You denote a database trigger by the keywords **ON DATABASE**, as in this partial example:

```
CREATE OR REPLACE TRIGGER my_db_trigger
        AFTER SERVERERROR ON DATABASE
… trigger code goes here …
```

Allowable database trigger events are:

Database Trigger Event:	Fires When…	Restrictions:
LOGON	Database session is established	**AFTER** trigger only
LOGOFF	Database session ends normally	**BEFORE** trigger only
STARTUP	Database is opened	**AFTER** trigger only
SHUTDOWN	Database is closed	**BEFORE** trigger only
SERVERERROR	Database exception is raised	**AFTER** trigger only
SUSPEND	Transaction is suspended by server error	none

Triggers are automatically enabled when created. You can disable them like this:

```
ALTER TRIGGER my_trigger DISABLE ;
```

Or disable all the DML triggers on a table at once:

```
ALTER TABLE employee DISABLE ALL TRIGGERS ;
```

To enable a trigger:

```
ALTER TABLE my_trigger ENABLE ;
```

To enable all DML triggers on a table:

```
ALTER TABLE employee ENABLE ALL TRIGGERS ;
```

Data Pump

Data Pump is new with 10g and supersedes the older Export and Import utilities of 9i (which are still present in 10g). Data Pump's Export and Import utilities are called **expdp** and **impdp**, respectively. They run on the *server* (the older Export and Import utilities ran on the *client*).

Use Data Pump to copy or extract (1) Data and/or (2) Data Definition Language (DDL). Use it within a single database or across different databases. Run it through the command line (as the **expdp** and **impdp** utilities), through the PL/SQL package **DBMS_DATAPUMP** and its procedures, or through the EM GUI.

Data Pump exports and imports can be performed on 5 levels. Here are the levels with the command-line keywords you would use to specify each:

- Database **FULL = Y**
- Tablespace **TABLESPACES =** list
- Schema **SCHEMAS =** list
- Table **TABLES =** list
- Transportable Tablespace **TRANSPORT_TABLESPACES =** list

Data Pump writes to 3 types of files:

- Dump files - Contains data and metadata
- Log files - Tracks job activities
- SQL files - Contains extracted SQL statements

There are many parameters to the **expdp** and **impdp** utilities. These lists focus on key ones for the test.

Here are the key **expdp** parameters:

Parameter:	**Use:**
ATTACH	Connects or re-connects a client to a Data Pump session or job.
CONTENT	**ALL** (default), **DATA_ONLY** or **METADATA_ONLY**. ***BOTH** is **not** an allowable parameter!*
DIRECTORY	Names a database directory object for output.
DUMPFILE	File to store export data in. Use **%U** to specify more than one file with the same name, **expdp** expands this to a unique number. Cannot be specified with the **ESTIMATE_ONLY** parameter.
ESTIMATE	Specifies either **BLOCKS** or **STATISTICS** as the way to estimate export disk space.
ESTIMATE_ONLY	Estimates the disk space needed for export -- **but does *not* perform the export. Can*not* be specified with the DUMPFILE parameter.**
EXCLUDE	Lists metadata objects excluded from export. This helps you to specify the objects for export.
FILESIZE	Maximum file size for each dump file.
FLASHBACK_SCN	Export the data as of this consistent System Change Number (SCN).
FLASHBACK_TIME	Export the data as of this consistent Time.
FULL	= Y means export the full Database.
HELP	Display Help info.
INCLUDE	Lists metadata objects included in the export. This helps you specify the objects for export.
JOB_NAME	The Job name.
KEEP_MASTER	= Y means keep the Master Table that controls the Data Pump job after the job completes.
LOGFILE	Name of the log file.
NETWORK_LINK	Use this Database Link to perform the export. *This allows you to export directly to another database server.*
PARALLEL	Maximum number of parallel threads for export.
PARFILE	Parameter file (may not be nested).
QUERY	Adds a **WHERE** clause for export selectivity.

SCHEMAS	Names of the users/schemas to export.
TABLES	Lists Tables to export.
TABLESPACES	Lists the Tablespaces to export.
TRANSPORT_FULL_CHECK	= Y means check dependencies for transportable tablespaces.
TRANSPORT_TABLESPACES	= Y for transportable tablespace mode.
VERSION	**Only** exports objects compatible with the specified database version.

Many of the above parameters are also common to the Import (**impdp**) utility. Here are some other parameters for **impdp**:

Parameter:	**Use:**
FLASHBACK_SCN	Import data consistent with this Systems Change Number. **Valid only when NETWORK_LINK is specified.**
FLASHBACK_TIME	Import data consistent with the Time specified. **Valid only when NETWORK_LINK is specified.**
NETWORK_LINK	Imports from the source database using this Database Link.
REUSE_DATAFILES	Overwrites existing datafiles.
SKIP_UNUSABLE_INDEXES	= Y means skip bad indexes and continue the job.
SQLFILE	Metadata SQL statements are written to this file.
STATUS	Operational status.
TABLE_EXISTS_ACTION	Options are expanded in 10g and now include: **APPEND, REPLACE, SKIP, TRUNCATE**
TRANSPORT_DATAFILES	Datafiles to import via transportable tablespaces method.
REMAP_DATAFILE	Changes source Datafile name.
REMAP_SCHEMA	Changes source Schema name.
REMAP_TABLESPACE	Changes source Tablespace name.
TRANSFORM	Changes object creation attributes. *Relies on a boolean value for the transform.*

Data Pump dictionary views include:

- **V$SESSION_LONGOPS** - Preferred view for info on job progress
- **DBA_DATAPUMP_JOBS** - Active Data Pump job information
- **DBA_DATAPUMP_SESSIONS** - Session information
- **DATAPUMP_PATHS** - Valid object types for Export/Import

SQL*Loader

SQL*Loader allows you to load data from operating system flat files into Oracle databases. It loads a variety of data input formats including delimited text files (files in which data elements are separated by some *delimiter*, such as the comma), and also fixed-field files (files in which data items start in the same position in each input record).

SQL*Loader can load via the direct path or by conventional path. Direct path is faster. It directly writes data blocks to Oracle datafiles while bypassing the buffer cache and the redo and undo logging mechanisms of the database. Direct path restrictions include:

- Writes to tables above the high-water mark (HWM), increasing table size
- Cannot run concurrently with transactions against the table
- Does not support clustered tables
- Foreign key constraints are disabled during loading, and later enabled
- Triggers do not fire
- Indexes must be built after loading
- Uniqueness constraint violations leave indexes unusable

Conventional path loads data into a *bind array* and writes data via **INSERT** statements.

Invoke SQL*Loader by the **sqlldr** command, followed by various options. SQL*Loader uses several file types:

- Control - The mandatory control file directs SQL*Loader operations.
- Log - This mandatory file logs progress and outcomes.
- Data - Placing your load data in-line here is optional.
- Bad - Holds records that failed validation. If you do not specify a Bad file, SQL*Loader creates one for you.
- Discard - Holds records that did not satisfy control file selection criteria.

Managing Users and Security

User Accounts

Create, alter and drop users through the **CREATE**/**ALTER**/**DROP USER** commands. Here's an example of creating a user account:

```
CREATE  USER my_id   IDENTIFIED  BY my_password
            DEFAULT  TABLESPACE  userspace
            TEMPORARY  TABLESPACE  temp
            PASSWORD  EXPIRE
            QUOTA 1M  ON  userspace ;
```

There are 3 ways a user can be authenticated to Oracle (verified as a legitimate user).

The **IDENTIFIED BY** clause above specifies the user id's initial password. This is password authentication performed by Oracle database.

The alternative keywords **IDENTIFIED EXTERNALLY** means an operating system login suffices for Oracle access. When Oracle relies on the operating system to authenticate users it is called external authentication.

Keywords **IDENTIFIED GLOBALLY AS 'external_name'** means a central product like Oracle Security Server handles security. The global authentication parameters may be coded slightly differently depending on what product does the authentication.

The **DEFAULT TABLESPACE** specified in the **CREATE USER** statement will contain any objects the user creates but does not specify a tablespace for. You must grant the user some space via the **QUOTA** clause to create objects in the tablespace. Grant unlimited space (as available) by the **QUOTA UNLIMITED** clause. The **TEMPORARY TABLESPACE** is used for operations requiring temporary workspace, like sorting. By default, the quota on it is unlimited.

You can alter any of the attributes of a user account (except the user name) by an **ALTER USER** statement. If you alter the password, it takes effect next time the person logs in. You must have the **CREATE USER** privilege to create a new user, and the **ALTER USER** privilege to alter user accounts.

To remove a user and all objects he owns from the database:

```
DROP USER  my_id  CASCADE ;
```

Keyword **CASCADE** drops all objects owned by that user. If you try to drop the user but exclude **CASCADE**, you get an error message if he owns objects. You can only drop a user when he is not logged on.

Profiles

A profile is a named set of resource limits. Use profiles to set:

- Resource limits per session
- Resource limits per SQL call
- Various restrictions or requirements on passwords

Create a new profile by the **CREATE PROFILE** statement, then assign it to a user by **ALTER USER**. By default, all users are created with the default profile (which provides unlimited resource usage). Change a profile by issuing **ALTER PROFILE**.

When a user exceeds a per SQL call resource limit, his last SQL command is rolled back, and no further work is allowed until a commit, rollback, or exit.

Here are allowable resource and password parameters for **CREATE**/**ALTER PROFILE.** All can be set to a value, or to **UNLIMITED** or **DEFAULT**:

Profile Resource Parms:	**Profile Password Parms:**
SESSIONS_PER_USER	FAILED_LOGIN_ATTEMPTS
CPU_PER_SESSION	PASSWORD_LIFE_TIME
CPU_PER_CALL	PASSWORD_REUSE_TIME
CONNECT_TIME	PASSWORD_REUSE_MAX
IDLE_TIME	PASSWORD_LOCK_TIME
LOGICAL_READS_PER_SESSION	PASSWORD_GRACE_TIME
LOGICAL_READS_PER_CALL	PASSWORD_VERIFY_FUNCTION
COMPOSITE_LIMIT	
PRIVATE_SGA	

This example creates a profile that limits connect time to 20 minutes and requires that the password be changed every 30 days. It then assigns this profile to a user:

```
CREATE PROFILE my_profile LIMIT
        CONNECT_TIME = 20
        PASSWORD_LIFE_TIME = 30 ;
ALTER USER  scott  PROFILE  my_profile ;
```

Privileges

There are 2 kinds of privileges: system and object. System privileges give system-wide abilities or the ability to perform an action on an object type. Object privileges are the ability to perform an action on a particular object (table, view, package, etc.)

GRANT system privileges to a user, role, or **PUBLIC** (everyone):

```
GRANT { system_privilege | ALL PRIVILEGES | role }  TO
        { user | role | PUBLIC }  {WITH ADMIN OPTION} ;
```

The optional clause **WITH ADMIN OPTION** allows the recipient to grant his system privileges to others. Dictionary views **DBA_SYS_PRIVS** and **USER_SYS_PRIVS** track system privileges granted. There are over 90 different system privileges -- be generally familiar with them. They apply to all kinds of database objects including indexes, the job scheduler, procedures, profiles, roles, sequences, sessions, synonyms, tables, tablespaces, triggers, users, views, and other objects.

Object privileges are rights on specific objects. You can grant them if you're the object owner (or have the system privilege **GRANT ANY PRIVILEGE**, or have been given an object privilege **WITH GRANT OPTION** so you can pass it on). Dictionary tables **DBA_TAB_PRIVS**, **USER_TAB_PRIVS**, and **ALL_TAB_PRIVS** track object privileges. The object privileges the object owner can grant include **ALTER**, **DELETE**, **INDEX**, **INSERT**, **REFERENCES**, **SELECT**, **UPDATE**, **EXECUTE**, and **READ**. These apply to objects like tables, views, sequences, packages, functions, etc.

There are also privileges that can be granted only at the column level (for tables, views or sequences). These are **INSERT**, **UPDATE**, and **REFERENCES** only. Column privileges are tracked in **DBA_COL_PRIVS**, **USER_COL_PRIVS_RECD** and **USER_COL_PRIVS_MADE**.

Revoke (rescind) grants through the **REVOKE** statement. Object privileges automatically cascade when revoked, while system privileges do not! So if you granted a system privilege **WITH ADMIN OPTION**, you still must individually **REVOKE** whatever system privileges were subsequently granted with that authority. Example: I grant some table and system privileges to John (**WITH GRANT OPTION** and **WITH ADMIN OPTION**, respectively). I revoke both grants from John. All table privileges he has given out are rescinded, but any system privileges he gave out remain in force.

Roles

Roles are groups or collections of system and/or object privileges (and other roles). Granting and revoking privileges through roles simplifies security administration, reduces the number of commands you issue, and reduces errors.

You can grant almost any privilege through roles. You can grant create object privileges to a role (i.e., 'create role testrole;' 'grant create procedure to testrole' 'grant create any table to testrole'). Issue direct grants for those privileges. Manage roles by the commands **CREATE**/**ALTER**/**DROP ROLE**.
By default, all roles assigned to a user id are active (or *enabled*). The user can change this by issuing various **SET ROLE** commands:

- **SET ROLE** - To switch to a specific role
- **SET ROLE ALL** - Activate all possible roles
- **SET ROLE NONE** - Disable all roles
- **SET ROLE ALL EXCEPT** - Activate some roles and not others

The **MAX_ENABLED_ROLES** initialization parameter determines the maximum number of roles that can be active (enabled) for any user. Data dictionary view **SESSION_ROLES** tells what roles are active for your session.

Implementing Security

When you create a new Oracle database, it will likely have certain security vulnerabilities you'll want to eliminate.

Creating an Oracle database always creates the **SYS** and **SYSTEM** accounts for the data dictionary owner and administrative account, respectively. Creating a database using the Database Configuration Assistant (DBCA) also creates the **SYSMAN** and **DBSNMP** user ids for use by the Enterprise Manager (EM). Depending on the features and options you install, Oracle may also create other user accounts automatically.

DBCA creates all accounts as *locked* and *expired*, except for **SYS**, **SYSTEM**, **SYSMAN**, and **DBSNMP**. If you use other means to create the database, many other accounts may be left open and immediately usable.

To enforce good security, lock and expire any unnecessary user ids like this:

```
ALTER USER  mdsys  PASSWORD EXPIRE  ACCOUNT LOCK ;
```

Or use EM to do this. You can always unlock and un-expire them later if you need them.

Revoke any unnecessary privileges from **PUBLIC** (every user). **PUBLIC** can **EXECUTE** various packages including **UTL_TCP**, **UTL_SMTP**, **UTL_HTTP**, **UTL_FILE** and others. Revoke these privileges by statements like this:

```
REVOKE  EXECUTE  ON utl_tcp  FROM PUBLIC ;
```

Ensure users cannot access the tables that underlie the data dictionary by setting the initialization parameter **O7_DICTIONARY_ACCESSIBILITY** to **FALSE**. This is the default. Turn off external authentication for user ids by setting initialization parameter **REMOTE_OS_AUTHENT** to **FALSE**. This is the default.

Carefully review who has powerful privileges like **SYSDBA** and **DBA**. Only users who must have them need these roles.

Auditing

Oracle supports 4 kinds of auditing:

- **Statement** - Audits the execution of specified SQL statements.
- **Privilege**- Audits system privileges and execution of SQL statements that require specific system privileges.
- **Object** - Audits execution of SQL statements requiring specific object privileges.
- **Fine-grained** - Audits based on access to data and data content.

The setting of initialization parameter **AUDIT_TRAIL** tells where audit records are written:

AUDIT_TRAIL Setting:	**Use:**
NONE	Default setting.
DB	Write audit records in the database.
DB_EXTENDED	Write audit records to the database along with bind variables (**SQLBIND**) and the text of the SQL statement that caused the entry (**SQLTEXT**).
OS	Write audit records to operating system files.

Statement Auditing – this audits execution of specified SQL statements. You specify the **AUDIT** command with any of about 30 options or keywords, each of which audits execution of an associated group of SQL statements.

For example, use option **TABLE** on the **AUDIT** statement to audit the execution of any **CREATE TABLE**, **DROP TABLE**, and **TRUNCATE TABLE** statements:

```
AUDIT  table ;
```

You can narrow auditing down to a specific user by:

```
AUDIT  table  BY  johnny ;
```

Narrow auditing down to successfully executed statements by the **WHENEVER SUCCESSFUL** keywords, or to statements that fail by the **WHENEVER NOT SUCCESSFUL** clauses:

```
AUDIT  table  BY johnny  WHENEVER SUCCESSFUL ;
```

Set the level of auditing to once per triggering session or once per triggering action by the **BY SESSION** and **BY ACCESS** keywords (respectively):

```
AUDIT  table  BY johnny  BY SESSION ;
```

Disable statement auditing (reverse the effects of an **AUDIT** command), by the **NOAUDIT** statement:

```
NOAUDIT  table ;
```

Privilege Auditing – this records the execution of SQL statements that require the rights to any system privilege you specify. Examples of such system privileges would be SELECT ANY TABLE or GRANT ANY PRIVILEGE.

Privilege auditing is enabled and disabled just like Statement Auditing, with different options on the **AUDIT** statement. Here are examples:

```
AUDIT  create any table ;
AUDIT  create any table BY johnny ;
AUDIT  create any table BY johnny BY ACCESS ;
NOAUDIT  create any table ;
```

Object Auditing -- this records execution of SQL statements that require specific object privileges, such as SELECT, INSERT, UPDATE, DELETE, and EXECUTE. Use of the AUDIT and NOAUDIT statements and their keywords are similar to Statement and Privilege Auditing. The difference is that you specify the object to audit, as in these examples:

```
AUDIT  select  ON  prod.employee ;
AUDIT  insert ON  prod.employee BY SESSION WHENEVER SUCCESSFUL ;
NOAUDIT  select  ON  prod.employee ;
```

Unlike Statement and Privilege Auditing, Object Auditing is enabled for all users or no users. You cannot apply it selectively to specific users.

Fine-Grained Auditing (FGA) -- FGA audits based on data content and user access to that data. FGA can apply to the table and optionally the column access level. You use the DBMS_FGA package to create, enable, disable, and drop FGA auditing policies for the database.

To create a FGA audit policy, use the **DBMS_FGA** package's **ADD_POLICY** procedure. In this procedure, the **AUDIT_CONDITION** parameter is a SQL expression that evaluates to a boolean **TRUE** or **FALSE**. If the **AUDIT_CONDITION** is fulfilled by an action, it results in an audit record. The **AUDIT_COLUMN_OPS** parameter is either set to **DBMS_FGA.ALL_COLUMNS** or **DBMS_FGA.ANY_COLUMNS**. The former requires access to *all* the columns listed in the parameter **AUDIT_COLUMN** to write the audit record, while the latter only requires that *any* of the columns in **AUDIT_COLUMN** be accessed to write the audit record. **DBMS_FGA.ANY_COLUMNS** is the default -- access to any of the columns listed writes an audit record. Parameter **STATEMENT_TYPES** tells which SQL statement(s) result in an audit record. The options are **SELECT**, **INSERT**, **UPDATE**, **DELETE**; the default is **SELECT**.

Useful **DBMS_FGA** procedures include:

- **ADD_POLICY -** Creates an FGA policy
- **ENABLE_POLICY -** Enables an FGA policy
- **DISABLE_POLICY -** Disables an FGA policy
- **DROP_POLIC** - Drops an FGA policy

Key data dictionary views for auditing:

Dictionary View:	Contents:
DBA_AUDIT_TRAIL	Displays database audit entries for Statement, Privilege, and Object Auditing (like **SYS.AUD$).**
SYS.AUD$	Records database audit entries for Statement, Privilege, and Object Auditing. You must prune this table yourself. For example, to purge entries older than 30 days: DELETE FROM sys.aud$ WHERE timestamp# < SYSDATE – 30 ;
DBA_FGA_AUDIT_TRAIL	Displays database audit trail entries for Fine-Grained Access (FGA) Auditing.
DBA_STMT_AUDIT_OPTS	Identifies the Statement Auditing options turned on.
DBA_PRIV_AUDIT_OPTS	Identifies the Privilege Auditing options turned on.
DBA_OBJ_AUDIT_OPTS	Identifies the Object Auditing options turned on.
DBA_AUDIT_POLICIES	Lists the FGA policies in the database.

Database Consistency and Concurrency

Locking prevents two users from invalidly updating the same data at the same time. It is key to preserving the accuracy of data (data integrity). Locking is automatically performed within the database, but it's useful to know how it works to optimize performance and resolve any locking issues that may pop up. Oracle automatically locks the least amount of data possible for maximum concurrency and performance. Oracle locks data on any of 3 levels: **row**, **multiple rows**, **table**.

Read-only queries never require a lock in Oracle databases because they can be directed against the pre-lock image of data that resides in the Undo tablespace.

Oracle allocates locks on a First-In, First-Out (FIFO) basis. Sessions release locks upon either *implicit* or *explicit* **COMMIT** or **ROLLBACK**. Implicit means the **COMMIT** or **ROLLBACK** is automatically issued on your behalf (for example, when your program or session ends with a transaction still open or active). Explicit means you directly issue the **COMMIT** or **ROLLBACK** statement yourself through your program or session.

You can explicitly lock an entire table to favor a particular session (assuming you have the required privilege):

```
LOCK TABLE hr.employee IN EXCLUSIVE MODE ;
```

This may be advantageous to speed up a large batch update program, for example, at the expense of other users.

The default behavior of a session or program that cannot get a lock is to wait. To avoid waiting and instead get an error message if a table is locked by someone else, specify **NOWAIT**:

```
LOCK TABLE hr.employee  IN SHARE ROW EXCLUSIVE MODE NOWAIT;
```

Valid modes for the **LOCK TABLE** statement are: **ROW SHARE**, **ROW EXCLUSIVE**, **SHARE**, **SHARE ROW EXCLUSIVE**, and **EXCLUSIVE**.

Lock conflicts occur when some transactions are waiting on rows being used by other transactions. Use the EM Database Control panels to easily identify lock conflicts and the waits they cause. Or query **V$SESSION**, **V$TRANSACTION**, **V$LOCK**, and **V$LOCKED_OBJECT** for this same information (also DBA_LOCKS)

Deadlocks occur when two transactions each have a resource the other needs but refuses to give up. Oracle automatically detects and resolves deadlocks. The session that causes the deadlock to be detected will cause the statement causing the deadlock to be rolled back with the message: **ORA-00060 Deadlock detected while waiting for resource.**

Oracle Net Services

Oracle Net is the communications and networking component for Oracle databases. It supports **single-tier**, **two-tier**, and **n-tier** architectures. N-tier architecture is the most scalable. Oracle Net supplies connectivity across multiple operating systems, platforms, and protocols. Among other features, it supports Java with JDBC, web applications, and location transparency.

Oracle Internet Directory (OID) provides a central repository for network names resolution. It supports the Lightweight Directory Access Protocol (LDAP). In 10g, OID runs as an application that uses an Oracle database. Security is tailored through policies and integration of the Secure Sockets Layer (SSL).

Oracle's Connection Manager is middleware that supports connection multiplexing and cross-protocol interconnectivity. It runs as the background processes **CMGW** and **CMADMIN** for its administration. Oracle Advanced Security supports secure transmission, authentication, and single sign-on. It provides industry-standard algorithms for encryption and checksumming.

Heterogeneous Services includes Oracle Transparent Gateway and Generic Connectivity. The former is a ready-to-go gateway to various data sources and databases, while the latter is a set of agents with which you can build custom connectivity solutions based on Microsoft's OLE Database standard database interface.

Oracle Net on the Server

Configure Oracle Net on the server through the Installer's Oracle Net Configuration Assistant GUI for basic configurations. Or use Oracle Net Manager, the Enterprise Manager screens, or the **lsnrctl** command-line utility.

The major files these methods configure or edit are:

- **listener.ora** - Configures Listeners
- **sqlnet.ora** - Profile for network connections
- **tnsnames.ora** - Optional file for remote access and name resolution

The **listener.ora** file configures one or more listeners. Here are some of its key parameters:

listener.ora Parameter:	**Use:**
LISTENER	The name of the listener (default is **LISTENER**).
DESCRIPTION	Describes the listening locations.
ADDRESS_LIST	Address info for the listening locations.
PROTOCOL	The protocol for a listening location.
HOST	Name of the host where the listener resides.
PORT	Port on which the listener listens.
SID_LIST_LISTENER	Lists Oracle services the listener handles.
SID_DESC	Describes the Oracle SIDs.
GLOBAL_DBNAME	The Global Database Name, which must match the **SERVICE_ NAMES** initialization parameter value.
ORACLE_HOME	Oracle home directory for its binaries.
SID_NAME	Name of the Oracle SID for the instance.
INBOUND_CONNECT_TIMEOUT	How long the listener waits for a client response.
SAVE_CONFIG_ON_ STOP	Whether **lsncrtl** session changes should be saved.
LOG_FILE	Overrides the default location to write the listener log file to.
TRACE_LEVEL	If missing or set to **OFF**, tracing is disabled. If set to **USER**, **ADMIN**, or **SUPPORT**, tracing is enabled at that level (each level provides more information in the order listed).
TRACE_FILE	Where to write the listener trace file.
PASSWORD	Password to use the **lsnrctl** utility for administration.
TRACE_DIRECTORY	Directory to write the trace file to.
LOG_DIRECTORY	Directory to write the log file to.

As discussed in the section above "User Connections and Oracle Shared Server," listeners may work with dedicated or shared connections. For both kinds of connections, listeners may hand off client connections by either direct or redirect methods. Direct hand-offs are also know as bequeath connections. In direct hand-offs, the listener gives its client connection directly over to the dedicated server process or dispatcher with which the client will communicate to handle its requests. With the redirect methods, the listener sends connection information back to the client, which enables the client to then form a new con-

nection with its dedicated server or dispatcher process.

A single listener can listen to an unlimited number of service names. *Default listener* characteristics are:

- Name **LISTENER**
- Port **1521**
- Protocols **TCP** and **IPC**
- Host Name The default host name
- SID Name The default instance

Listener registration can be static or dynamic. Static service registration occurs when you update the **listener.ora** file. Dynamic service registration means that database automatically registers its presence with listeners. It is required to use features like load balancing and automatic fail-over. The PMON process registers the database information with the listener.

For dynamic service registration, use either of these methods:

- Use the default listener configuration
- Specify the 3 required initialization parameters:
  - **LOCAL_LISTENER** - Location of the listener to register with
  - **INSTANCE_NAME** - Name of the instance to register
  - **SERVICE_NAMES** - The instance name plus the domain name

Advanced features available when using dynamic service registration include:

Feature:	**What is it:**	**How to get it:**
Connect-Time Failover	Allows clients to connect to another listener if the first one is unavailable.	Code multiple entries under the **ADDRESS_LIST** section in **tnsnames.ora** file
Transparent Application Failover (TAF)	Allows clients to automatically reconnect to the database if necessary.	Code **FAILOVER_MODE** parameter in **tnsnames.ora** file.
Client Load Balancing	Allows clients to randomly connect to any of several listeners.	Code **LOAD_BALANCE** parameter in **tnsnames.ora** file and multiple listeners.
Connection Load Balancing	Distributes connections among dispatchers for Shared Server.	Use Shared Server with multiple dispatchers.

To start the lsnrctl utility, enter the **lsnrctl** command. Here are its major sub-commands:

lsnrctl Sub-command:	**Use:**
start [listener_name]	Start a listener.
stop [listener_name]	Stop a listener.
help	Display help information.
reload	Re-read **listener.ora** and pick up any changes without stopping and restarting the listener.
status	Get listener status.
services	Lists listener and connection details and history.
exit --or-- quit	Exit the utility.
change_password	Change the utility's password.
save_config	Saves changes.
trace	Turns on listener tracing.
version	Reports Oracle Net software and components versions.
set	Sets any of a dozen or so parameters within **listener.ora** file.
show	Displays values for any of the "settable" parameters.

Listener logging is on by default on a server and is written to the file named **listener.log**. Server tracing provides detailed diagnostic information. It is enabled by setting the **TRACE_LEVEL** parameter of the **listener.ora** file to **USER**, **ADMIN**, or **SUPPORT** (listed in order of increasingly detailed trace information).

Oracle Net on the Client

Configure Oracle Net on the client using the same tools as you use on the server. You configure the same basic files, **tnsnames.ora** and **sqlnet.ora** (depending on how you set it up).

The 3 basic pieces of information the user supplies to connect to an Oracle server are: **user ID**, **password**, and **net service name**. The net service name is a *connect descriptor*.

There are 5 methods of names resolution for clients (ways to resolve the connection information the client needs to connect to the server).

They are:

- Oracle Internet Directory (OID)
- External Naming (using an external service to resolve names)
- Hostnaming
- Oracle Easy Connect
- Localnaming

Set the **NAMES.DIRECTORY_PATH** parameter in the **sqlnet.ora** file to override the default search path for names resolution.

Hostnaming – The user supplies the user ID and password and the name of the host to connect to. The hostname is resolved either by a local HOSTS file on the client machine or by a central operating-system service such as Domain Naming Service (DNS).

The 4 requirements for hostnaming are:

- TCP/IP must be the network protocol.
- Requires an external naming service (like the **HOSTS** file or DNS).
- The listener must have its **GLOBAL_DBNAME** parameter set to the machine name.
- No advanced features are supported (such as Oracle Connection Manager).

Easy Connect Naming -- Along with hostnaming, this is the other method that does not require client configuration. For TCP/IP networks only, 10g allows you to connect to Oracle without a configuration file by fully specifying the connect string in this format:

```
CONNECT  username/password@//hostname [:port_number] [/service_name]
```

The hostname is the sole require parameter. Here is an example:

```
CONNECT  mylogin/my_pass@//my_host:1521/PROD0001
```

Only include the double slashes (//) if you're referring to a web address (a URL). Advanced features such as load balancing and connect time failover do *not* work when connecting in this manner.

Localnaming -- This method relies on client configuration of a **tnsnames.ora** file. If it is not in its default location, the environmental variable **TNS_ADMIN** should point to the location of the **tnsnames.ora** file.

Performance Monitoring and Maintenance

Oracle 10g vastly improves your ability to proactively monitor an Oracle database for performance issues, and makes it much easier to do so through the new Enterprise Manager GUI. Key to this effort is the new Automatic Workload Repository (AWR).

The new background process, Memory Monitor (MMON), activates every 60 minutes (by default) to collect performance statistics directly from the SGA. It is aided by the new background process, Memory Monitor Light (MMNL), which performs the same function whenever buffer space is filled, rather than at periodic intervals. The performance info is stored in the AWR -- a set of tables owned by user id **SYSMAN** and stored in the new 10g tablespace **SYSAUX**.

By default, AWR statistics are kept for **7 days**. You can modify this through the EM Database Control panels or through the new package **DBMS_WORKLOAD_REPOSITORY** and its procedure **MODIFY_SNAPSHOT_SETTINGS**. This example sets the collection **interval** to 30 minutes and the **retention** period to 43,200 seconds (which is 30 days):

```
execute DBMS_WORKLOAD_REPOSITORY.MODIFY_SNAPSHOT_SETTINGS
(interval=>30, retention=>43200 ) ;
```

AWR statistics collection is enabled by default in Oracle 10g. Initialization parameter **STATISTICS_LEVEL** determines this. Here are its 3 possible settings:

- **BASIC** - Turns off AWR statistics collection and almost all 10g automated performance features.
- **TYPICAL** - The default, collects statistics best for most environments.
- **ALL** - Captures even more data, including execution plans and more timing information.

Automatic Database Diagnostic Monitoring (ADDM) automatically runs right after each statistics collection process. ADDM works top-down to compare the new statistics to the previous two collection points and comes up with **DB Time**, an overall measurement composed of **CPU time** and **wait time**.

You can look at ADDM results through the EM Database Control GUI panels. The Performance Overview panel shows:

1. CPU use
2. Top SQL statements
3. Top sessions

The Advisor Central screen also leads to ADDM recommendations and links to other key Oracle 10g advisors, including the SQL Tuning Advisor, the SQL Access Advisor, the Memory Advisor, the Mean Time To Recover (MTTR) Advisor, the Segment Advisor, and the Undo Advisor. Each advisor gives you performance recommendations and advice specific to its area of expertise. Access them through the Enterprise Manager (EM) GUI panels (easiest) or through their data dictionary views, packages, and the command line (harder but scriptable).

The SQL Tuning Advisor automates SQL statement tuning. It operates in two modes: Normal and Tuning. Normal is real-time, while Tuning 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. Note that user-defined SQL statements are not executed; they are merely analyzed.

The kinds of analysis the SQL Tuning Advisor does are:

- Top SQL, improving resource-intensive SQL statements
- Analyzing a SQL Tuning Set (STS), SQL statements and their execution info
- Analyzing statistical snapshots
- Analyzing saved snapshot sets, sets of related snapshots

The SQL Access Advisor helps determine which *indexes*, *materialized views*, and *materialized view logs* will aid one query or a workload of queries. It differs from the SQL Tuning Advisor in that its suggestions are directed towards database design issues, rather than tuning the SQL statement's structure and optimization. SQL Access Advisor operates in two modes: Limited and Comprehensive. The latter analyzes a full or complete workload.

The Memory Advisor gives recommendations on Oracle's memory use including the SGA and user memory structures. Oracle can automatically size the main SGA memory components including the database buffer cache (**DB_CACHE_SIZE**), the shared pool (**SHARED_POOL_SIZE**), the large pool (**LARGE_POOL_SIZE**), and the java pool (**JAVA_POOL_SIZE**). This is done through the feature called Automatic Shared Memory Management (ASMM). Set init parm **SGA_TARGET** to enable ASMM.

The Mean Time To Recovery (MTTR) Advisor explains how to minimize recovery time in case of instance failure. You can enable it and automatic checkpoint tuning by setting initialization parameter **FAST_START_MTTR_TARGET** to a non-zero value, or by not setting it at all.

The Segment Advisor identifies segments you can shrink through a segment shrink command. This reclaims unused space within the segment.

The Undo Management Advisor gives advice concerning use of the Undo Tablespace, which stores before-change data images. This undo data is used for:

- Read-consistency of data
- User **ROLLBACK** of a transaction
- Database recovery (to undo un-committed transactions)
- Flashback, a set of features that allows easy recovery of data incorrectly updated due to logical errors (user error in incorrectly changing data)

When a transaction starts, it is assigned an undo segment, owned by user **SYS**, in the undo tablespace. View **V$TRANSACTION** shows which transactions use which undo segments.

Undo contains 3 types of information:

Uncommitted transactions	Required for rollbacks and never over-written.
Committed transactions	Committed but needed to support the undo retention interval.
Expired Undo	Undo that is no longer needed to support an active transaction and is over-written when space is needed.

To enable automatic undo management, set initialization parameter **UNDO_MANAGEMENT = AUTO** (instead of the alternative **MANUAL**). This parameter is **not** dynamic, so to change it, you must stop and restart the instance (even if you are using an **SPFILE**). Initialization parm **UNDO_TABLESPACE** specifies the name of the undo tablespace. See it in SQL*Plus by:

```
SQL> show parameter undo_tablespace ;
        Change it by:
SQL> alter system set undo_tablespace=undo_batch ;
```

Set init parm **UNDO_RETENTION** to 0 to enable automatic undo retention tuning. Automatic undo retention tries to maintain at least 900 seconds (15 minutes) of undo records. The undo retention value is not guaranteed by the undo system.

You can guarantee that undo will always be available for your specified retention period by using the new keywords **RETENTION GUARANTEE:**

```
SQL> alter tablespace undotbs1 retention guarantee :
        Or turn off the retention guarantee:
SQL> alter tablespace undotbs1 retention noguarantee ;
```

Other Features

ADDM allows you to set up alerts to notify you when events occur or when thresholds (or metrics) are exceeded. The 4 default alerts for new Oracle 10g databases are:

- Tablespace Space Usage - A warning at 85% full and critical message at 97%
- Snapshot Too Old - A query ran out of undo space
- Recovery Area Low on Space - Flashback recovery area needs more space
- Resumable Session Suspended - A resumable operation becomes suspended

The Alert log is a good source of tuning information. It is Oracle's basic means to record important system events. Find it at the location specified by initialization parm **BACKGROUND_DUMP_DEST**.

Oracle background process trace files are written to initialization parm location **BACKGROUND_DUMP_DEST**, and user trace files are written to **USER_DUMP_DEST**.

The dynamic performance views (V$ views) and data dictionary tables also have many views useful for performance analysis.

Remember that Oracle's cost-based optimizer (CBO) determines data access paths for queries automatically. It requires database statistics to create optimal data access paths. If you created the database by the Database Configuration Assistant (DBCA), Oracle automatically schedules a job that collects table and index statistics to run daily between 10pm and 6am. Or, you can manually collect these statistics by running a job using the **DBMS_STATS** package and scheduling it via the **DBMS_SCHEDULER** package.

10g allows you for the first time to gather statistics on the data dictionary tables and on Oracle's internal memory tables (called fixed tables). Use Enterprise Manager to do this, or run new **DBMS_STATS** procedures **GATHER_DICTIONARY_STATS** and **GATHER_FIXED_OBJECTS_STATS**.

Backups and Recoveries

Archive Logging

A database can either be in Archivelog mode or Noarchivelog mode. When a database is in Archivelog mode and a log switch occurs, the Archiver background process (ARCn) 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, full online redo logs are *not* archived by ARCn, and redo information is over-written and lost. Archivelog mode means that no transaction logs 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). Noarchivelog mode databases require that the entire database be recovered to a specific point in time, which could result in lost transactions, and that the database be shut down to perform this recovery. Many kinds of recovery with Archivelog mode databases permit the database to be up and operating while recovery occurs.

To change the archivelog mode:

1. If using a textual parameter file (**PFILE**), set initialization 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 init parm **LOG_ARCHIVE_FORMAT**. Determine the reliability level for the writing of the archive logs by init parm **LOG_ARCHIVE_MIN_SUCCEED_DEST**.

The Flash Recovery Area

The Flash Recovery Area is a single area on disk to which all recovery-related files are written. Oracle 10g uses it to implement quick recovery from disk (as opposed to slower recovery from tape files). The flash recovery area contains:

- Control files
- Datafile image copies and backup sets made by Recovery Manager (RMAN)
- Archived log files
- Control file backups
- **SPFILE** file backups

Establish the flash recovery area by setting 2 init parms:

- **DB_RECOVERY_FILE_DEST_SIZE** - Size of the flash recovery area
- **DB_RECOVERY_FILE_DEST** - Location of the flash recovery area

You **must** set the first parameter before the second. To disable the flash recovery area, clear the value of **DB_RECOVERY_FILE_DEST** first, then you can clear **DB_RECOVERY_FILE_DEST_SIZE.**

Inspect the view **V$RECOVERY_FILE_DEST** to see what's in the flash recovery area.

Kinds of Backups and RMAN's Role

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

Inconsistent or online backups are made while the database is up. 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), then recovered (by application of 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.

A full backup includes all blocks of every datafile included in a whole or partial database 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.

Recovery Manager (RMAN) is Oracle's backup tool. It writes either image copies, duplicates of datafiles, or backup sets, and backups in an RMAN-internal format. Image copies can be quicker for recovery but can only be written to disk. They cannot be compressed. Incrementally updated backup applies changed blocks to an existing image copy. This reduces the number of files required for recovery. Backup sets may be written to disk or tape and also may be compressed to save space.

10g supports the new block-change tracking file that keep track of what data blocks are changed. This enables fast incremental backup, since only changed blocks need be backed up (instead of all blocks). RMAN backs up everything in your database except:

1. Password files
2. Text-based initialization parameter files (**PFILE**s)
3. The Oracle binaries themselves (which most sites back up by running operating-system level disk backup utilities)

RMAN allows you to configure its defaults for what to backup, how to back it up, where to put it, what level of parallelism to use, etc. Display the configuration with the RMAN **SHOW ALL** command.

You can create image copies and backup sets either through RMAN's line commands or through the EM Database Control GUI.

Recoveries

There are 6 categories of failures in a database environment that can occur:

Failure Type:	Meaning:
Statement Failure	A SQL statement fails. Oracle automatically handles this by sending an error back to the session issuing the incorrect statement.
User Process Failure	A user process fails. Process Monitor (PMON) automatically cleans up after the failed session by rolling back the in-flight transaction and releasing user locks.
Network Failure	A network component fails between the client and database server. The database may see this as a User Process Failure or as a client time-out.
Instance Failure	The instance stops without synchronizing internal SCNs (due to sudden power failure, for example). Oracle handles this automatically upon database **STARTUP** by applying online redo logs and then eliminating any un-committed transactions by applying undo records. This is sometimes called "rolling forward, then rolling back." It is also known as crash recovery. Tune how long instance recovery requires by using the Mean Time To Recovery (MTTR) Advisor and setting initialization parameter **FAST_START_MTTR_TARGET**.
User Error	Also called a logical error, this is when a user does something they don't intend to do. For example, they run a correct SQL statement to delete data -- but they didn't actually mean to delete the data!
Media Failure	This is when data is lost or corrupted, for example, by a disk drive or controller failure. You must perform media recovery to fix this.

This table shows you only have to intervene to recover from two key kinds of failure: user or logical error, and media failure.

Recovering from Logical Errors

10g offers 5 kinds of *flashback recovery* as a prime mechanism to recover from user error, plus a tool called LogMiner:

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.
LogMiner (not a flash-back feature)	Redo logs	Extracts DML and DDL from Redo Logs via view **$LOGMNR_CONTENTS**. Supplies both Undo and Redo SQL statements.

Flashback Query -- for the simplest use of flashback query, simply use the **AS OF TIMESTAMP** clause in a **SELECT** statement. For example, to see data from the EMPLOYEES table as of 30 minutes ago, issue:

```
SELECT * FROM hr.employees
AS OF TIMESTAMP (systimestamp – INTERVAL '30' MINUTE ) ;
```

You'll need a large enough Undo Tablespace and a long enough **UNDO_RETENTION** initialization parameter to have saved undo data back far enough to be able to retrieve the data you want. You can only retrieve data from as far back as you have saved it in undo.

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:

```
ALTER TABLE my_schema.table_name ENABLE ROW MOVEMENT ;
```

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

```
FLASHBACK TABLE my_schema.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're querying.

Flashback Drop -- recovers a dropped table to some previous point in time before the drop. It 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.** You can also issue the **SHOW RECYCLEBIN** command from within SQL*Plus. You may want to query **DBA_CONSTRAINTS** if working with more than one table, to see if there were logical relationships (dependencies) between them, before recovering any of them.

Only tables defined within locally managed tablespaces are placed into the recycle bin when dropped. Dependent objects for eligible tables (like its indexes) are also preserved and recovered, except for bitmap join indexes, referential integrity constraints, and materialized view logs.

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

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

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

```
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 how big you sized the recycle bin and how Oracle reuses its space. Objects in the recycle bin are deleted on a First-In-First-Out (FIFO) basis. Oracle gets the 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, PURGE INDEX,** and **PURGE TABLESPACE** commands. **DBA_FREE_SPACE** and **DBA_RECYCLEBIN** views are where to look to see what's in the recycle bin and their sizes.

Recovering from Media Errors

An instance will not start (or will abort if running) if:

- Any one control file goes missing
- All members of the current online redo log group fail
- Any datafile in either the **SYSTEM** or **UNDO** tablespaces becomes unavailable

To recover from a lost control file or a lost redo log file, use the multiplexed copies of these files. For example, if you lose a control file, **SHUTDOWN ABORT** the instance. Copy one of valid control files to the location of the missing or corrupted one. Then, **STARTUP** the instance. Remember that the instance's automatic instance recovery (or crash recovery) will handle the issues created by your **SHUTDOWN ABORT** automatically upon instance startup.

For a missing or bad online redo log file, follow these steps:

1. Query **V$LOGFILE** to determine which log group member is bad.
2. Archive the log file group's contents by:

   ALTER SYSTEM ARCHIVE LOG GROUP group_number ;
3. Copy one of the good members in to replace the bad member.

   Or, you can clear the log group and re-create the missing member by:

   ALTER DATABASE CLEAR LOGFILE GROUP group_member ;

If your database is in Noarchivelog mode and you lose any datafile, you must abort the instance, then restore the entire database. This includes all datafiles and control files (not just the bad or missing ones). The entire database will be restored and current to the point in time when the backup you restore from was made. Thus, you could lose some data. This procedure applies whether or not the datafiles you lost were critical (the **SYSTEM** and **UNDO** datafiles) or non-critical (any other datafiles).

Recovering a Noarchivelog mode database is rather like recovering an operating system file you backed up to a floppy disk – the data in the database is all set back to the time of the restored copy. If you cannot afford to lose any data, use Archivelog mode.

If your database is in Archivelog mode, the procedure you follow to recover depends on whether you lost critical (**SYSTEM** and **UNDO** datafiles) or non-critical (all other datafiles). If you lost a critical datafile, recover it by these steps:

1. **SHUTDOWN ABORT**
2. **STARTUP MOUNT**
3. Restore the lost datafile (copy it into place from a backup)
4. Recover the lost datafile by applying redo logs
5. Complete the process with **ALTER DATABASE OPEN**

The restore and recover steps are easily accomplished through the EM Database Control panels. Or you can use RMAN line commands that look similar to these for the restore and recovery (this example assumes that the datafile lost is datafile #1):

```
RMAN>  run {restore datafile 1; recover datafile 1} ;
```

If your database is in Archivelog mode and you lost a non-critical datafile, then you can keep the database up and **OPEN** while you restore and recover the non-critical datafile. This is called an online recovery and no committed transactions will be lost. What you need to do is:

1. Take the affected datafile(s) offline
2. Restore the datafile (copy it in from a backup)
3. Recover the datafile (apply the redo log to the datafile to bring its transactions and SCN up-to-currency)
4. Alter the datafile back online

This process is easiest through the EM Database Control panels. Here are the equivalent RMAN line commands (this example assumes that datafile number 6 was lost):

```
RMAN>  run { sql 'alter database datafile 6 offline' ;
             restore  datafile 6 ;
             recover datafile 6 ;
             sql 'alter database  datafile 6 online' ; } ;
```

Closing Remarks

If you understand the material in this Exam Manual, you should pass. Good luck!

Practice Questions

Chapter 1 Creating an Oracle Database

1. Which of the following System Global Area (SGA) components are required?

   A. Shared Pool

   B. Large Pool

   C. Java Pool

   D. Database Buffer Cache

   E. Flashback Log Buffer

   F. Redo Log Buffer

   Select the three best answers.

   - ❑ A. A
   - ❑ B. B
   - ❑ C. C
   - ❑ D. D
   - ❑ E. E
   - ❑ F. F

2. Which of the following tablespaces are required for an Oracle 10g database?

   A. SYSTEM tablespace

   B. REDO tablespace

   C. TEMP tablespace (assuming the SYSTEM tablespace is locally managed)

   D. TOOLS tablespace

   E. SYSAUX tablespace

   F. UNDOTBS1 tablespace (assuming the SYSTEM tablespace is locally managed)

   Select the three best answers.

   - ❑ A. A
   - ❑ B. B
   - ❑ C. C
   - ❑ D. D
   - ❑ E. E
   - ❑ F. F

3. Which two of the following statements are true about DBCA seed templates?
Select the best two answers.

- ❑ A. Seed templates allow you to create databases (empty of data) using DBCA.
- ❑ B. Seed templates are XML files.
- ❑ C. Seed templates contain both datafiles and redo logs files copied from an existing database.
- ❑ D. Using a seed template means that you cannot change the number of control files and redo log groups.
- ❑ E. DBCA's seed templates are the only way to clone Oracle 10g databases.
- ❑ F. Recovery Manager uses seed templates in cloning databases.

4. Assuming an Oracle database is based on underlying operating system files, which statement describes the proper ordering of the Oracle's data storage hierarchy, from the largest unit to the smallest?
Select the best answer.

- ❍ A. tablespaces -> segments -> extents -> operating system blocks -> data blocks
- ❍ B. segments -> tablespaces -> extents -> data blocks -> operating system blocks
- ❍ C. tablespaces -> segments -> extents -> data blocks -> operating system blocks
- ❍ D. tablespaces -> extents -> segments -> data blocks -> operating system blocks
- ❍ E. operating system blocks -> data blocks -> extents -> segments -> tablespaces

Chapter 2 PL/SQL

1. Which of the following answer choices is not a valid PL/SQL compiler initialization parameter for Oracle 10g?
Select the best answer.

- ❍ A. PLSQL_WARNINGS
- ❍ B. PLSQL_DEBUG
- ❍ C. PLSQL_OPTIMIZE
- ❍ D. PLSQL_OPTIMIZE_LEVEL
- ❍ E. PLSQL_CODE_TYPE

2. Which of the following statements about triggers is not true?
Select the best answer.

- ❍ A. DML event triggers can execute BEFORE or AFTER an INSERT, UPDATE, or DELETE.
- ❍ B. DML event triggers can execute either FOR EACH ROW (the default) or FOR EACH STATEMENT.
- ❍ C. Database event triggers are specified by the trigger keywords ON DATABASE, and fire either BEFORE or AFTER for each of any numerous database events.
- ❍ D. DDL event triggers fire either BEFORE or AFTER any of a dozen or so specified DDL events.
- ❍ E. Database event triggers are specified by the trigger keywords ON DATABASE, and fire either for any of numerous database events.

Chapter 3 Monitoring and Resolving Lock Conflicts

1. Which of the following statements about deadlocks are true?

   Select the two best answers.

   - ❑ A. It is up to application programs to detect and resolve deadlocks.
   - ❑ B. Deadlocks occur when two or more users (or programs) each have a resource the other needs and they both refuse to give them up.
   - ❑ C. You could recognize a deadlock by looking at the EM Database Control screen "Database Locks."
   - ❑ D. Deadlocks can never occur within Oracle.
   - ❑ E. When a deadlock occurs, the session that has the least work to lose is the one that is rolled back.
   - ❑ F. Setting the optimistic locking initialization parameter prevents deadlocks.

2. Which statement best describes what has happened with this SQL statement? SQL> lock table prod.employees in share row exclusive mode nowait ; lock table prod.employees * ERROR at line 1: ORA-00054: resource busy and acquire with NOWAIT mode specified.

   Select the best answer.

   - ❍ A. The LOCK TABLE statement had a syntax error.
   - ❍ B. The SQL statement tried to get a SHARE ROW EXCLUSIVE mode lock. It failed to obtain this lock immediately, and since NOWAIT was specified, an error was returned to the user.
   - ❍ C. The SQL statement failed because SHARE ROW EXCLUSIVE locks are not allowed on the PROD.EMPLOYEES table.
   - ❍ D. The SQL statement failed because the user did not have the privilege required to obtain a SHARE ROW EXCLUSIVE lock on the PROD.EMPLOYEES table.

Chapter 4 Controlling the Database

1. An Oracle database that has been fully shut down passes through several states when being started up and opened for users. Which answer correctly describes these steps?

   Select the best answer.

   - ❍ A. MOUNT, NOMOUNT, OPEN
   - ❍ B. NOMOUNT, MOUNT, OPEN-except if you use the STARTUP RESTRICT command when you start the database
   - ❍ C. MOUNT, OPEN
   - ❍ D. NOMOUNT, MOUNT, OPEN
   - ❍ E. MOUNT, STARTUP OPEN

2. Which four of the following statements correctly describe a database SPFILE?

   Select the best four answers.

   - ❑ A. An SPFILE is a text file you can update with any text editor.
   - ❑ B. An SPFILE supports dynamic changes to the instance.
   - ❑ C. SPFILEs can be created from PFILEs.
   - ❑ D. You can create a PFILE from your SPFILE by issuing single command.
   - ❑ E. An instance can use both an SPFILE and a PFILE at one time.
   - ❑ F. When starting up a database with an SPFILE that is not explicitly named, Oracle searches first for the file named spfile$ORACLE_SID.ora, then for the file named spfile ora, then for the PFILE with the default name of init$ORACLE_SID.ora.

Chapter 5 Oracle Database Security

1. User SYSTEM grants SELECT ANY TABLE to user BOB using the WITH ADMIN OPTION. User SYSTEM then grants UPDATE on table EMPLOYEES to BOB using the WITH GRANT OPTION. BOB then grants SELECT ANY TABLE and UPDATE on the EMPLOYEES table to user SUE. The SYSTEM user now drops user account BOB. What happens to SUE's SELECT ANY TABLE privilege and her UPDATE privilege on the EMPLOYEES table?

   Select the best answer.

   - ❍ A. SUE loses her SELECT ANY TABLE privilege, but retains her UPDATE privilege on the EMPLOYEES table.
   - ❍ B. SUE retains her SELECT ANY TABLE privilege, but loses her UPDATE privilege on the EMPLOYEES table.
   - ❍ C. SUE loses her SELECT ANY TABLE privilege, and loses her UPDATE privilege on the EMPLOYEES table.
   - ❍ D. SUE retains her SELECT ANY TABLE privilege, and retains her UPDATE privilege on the EMPLOYEES table.
   - ❍ E. You can't tell what happens to SUE's privileges unless you know whether the SYSTEM user dropped the user id BOB with the CASCADE REVOKE keywords.

2. Which of the following answer choices are valid ways to configure authentication for Oracle database users?

   Select the three best answers.

   - ❑ A. External authentication
   - ❑ B. Oracle Security Server authentication
   - ❑ C. Global authentication
   - ❑ D. Local authentication
   - ❑ E. Password authentication
   - ❑ F. Hostnaming method

Chapter 6 Oracle Net Services

1. The steps listed below describe how a client connects to Oracle Shared Server when the server uses the redirect method. Which of the following answer choices puts the steps in the proper order?

   1. Listener sends info back to the client, redirecting the client to a dispatcher port
   2. The dispatcher process sends an acknowledgment back to the client
   3. The client first contacts the Oracle server (after resolving the service name)
   4. PMON sends info to the Listener about the client connection and its dispatcher
   5. Client sends a connect signal to its assigned dispatcher

   Select the best answer.

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

2. Which of the following statements accurately describe Oracle Net Services features?

   Select the three best answers.

   - ❑ A. Connection Load Balancing enables better distribution of connections among dispatchers in a shared server environment.
   - ❑ B. Client Load Balancing enables better distribution of connections among dispatchers in a shared server environment.
   - ❑ C. Features like connect-time failover, connection load balancing, transparent application failover, and client load balancing require multiple listeners servicing the database.
   - ❑ D. Connect-Time Failover allows a client to connect to another listener if the initial connection to the first listener fails.
   - ❑ E. Transparent Application Failover automatically fails over and re-establishes failed application-to-service connections for large databases, but it cannot be used with Real Application Clusters.
   - ❑ F. Whether or not you use dynamic service registration, you can use load balancing and connect-time failover.

Chapter 7 Backup and Recovery Concepts

1. When does the Log Writer not write to the online redo log files?

   Select the best answer.

   - ❍ A. When a user commits a transaction
   - ❍ B. When the redo log buffer becomes one-third full
   - ❍ C. After the Database Writer writes new or modified database buffer cache records to the datafiles
   - ❍ D. When the redo log buffer contains about 1 megabyte of changed records (this total does not include inserted and deleted records)

2. Which information is not contained in the database control files?
Select the best two answers.

- ❑ A. Instance name
- ❑ B. Database name
- ❑ C. Tablespace names
- ❑ D. SYS and SYSTEM user id passwords
- ❑ E. Log sequence number

Chapter 8 Storage Structures

1. Which two of the following statements about bigfile tablespaces are true?
Select the two best answers.

- ❑ A. A bigfile tablespace can have as many as 1,022 datafiles.
- ❑ B. The maximum size of a bigfile tablespace is 32 terabytes.
- ❑ C. Bigfile tablespaces can be more efficient for large databases because they can reduce the time required for datafile header operations.
- ❑ D. In Oracle 10g, a tablespace can contain both bigfile and smallfile datafiles.
- ❑ E. If you're not using either ASM or an underlying logical volume manager that supports dynamic volume resizing and striping or mirroring, you should not use bigfile tablespaces.
- ❑ F. In migrating to Oracle 10g, you must update your SYSTEM tablespace to be a bigfile tablespace.

Chapter 9 Oracle Shared Servers

1. Which of the following statements is not true regarding Oracle Shared Server?
Select the best answer.

- ❍ A. You should configure the Large Pool so that you do not affect the performance of the Shared Pool.
- ❍ B. With Shared Server, a single server process can service many user processes.
- ❍ C. Oracle Corp considers the Shared Server a performance enhancement option.
- ❍ D. In a Shared Server environment, Oracle moves a user's bind variables, cursor information, and sort area from the Program Global Area (PGA) into the User Global Area (UGA).
- ❍ E. Applications that generate large results sets or much network traffic are poor cand dates for efficient use of the Shared Server feature.
- ❍ F. You cannot startup or shutdown Oracle from a Shared Server connection.

2. Which code snippet configures five TCP/IP dispatchers and one IPC dispatcher for a Shared Server environment?

   Select the best answer.

   - ❍ A. dispatchers="(pr=tcp)(di=5)(pr=icp)(di=1)"
   - ❍ B. dispatchers="(prot=tcp)(disp=5)(prot=icp)(disp=1)"
   - ❍ C. dispatchers=(prot=tcp)(disp=5)(prot=icp)(disp=1)
   - ❍ D. dispatchers=(pr=tcp)(di=5)(pr=icp)(di=1)
   - ❍ E. start_dispatchers="(prot=tcp)(disp=5)(prot=icp)(disp=1)"
   - ❍ F. start_dispatchers=(prot=tcp)(disp=5)(prot=icp)(disp=1)

Chapter 10 Proactive Maintenance

1. Which of the following answer choices is not an option on which SQL Access Advisor can focus?

   Select the best answer.

   - ❍ A. Indexes
   - ❍ B. Materialized Views
   - ❍ C. Both indexes and materialized views
   - ❍ D. SQL tuning sets

2. Select the four default ADDM alerts from the following options.

   Select the best four answers.

   - ❑ A. Tablespace space usage alert
   - ❑ B. Damaged control file alert
   - ❑ C. Snapshot too old alert
   - ❑ D. Archive log area out of space alert
   - ❑ E. Recovery area low on free space alert
   - ❑ F. Resumable session suspended alert

Chapter 11 Database Backups

1. Which of the following answer choices are true statements about backup and recovery?

   Select the three best answers.

   - ❑ A. Image copies cannot be written directly to tape
   - ❑ B. Image copies can be compressed
   - ❑ C. Backup sets cannot be written directly to tape
   - ❑ D. Backup sets are written in a proprietary RMAN backup format
   - ❑ E. Image copies consist of one or more files called "backup pieces"
   - ❑ F. Oracle writes image copies and backup sets to the flash recovery area

2. Which kind of recovery is best for easily viewing the contents of a table at some previous point in time?

   Select the best answer.

   - ❍ A. Flashback database
   - ❍ B. Flashback drop
   - ❍ C. Flashback table
   - ❍ D. Flashback query
   - ❍ E. An RMAN recovery on the tablespace level.
   - ❍ F. An RMAN tablespace point in time recovery (TSPITR)

Chapter 12 Database Recovery

1. How do you perform "crash recovery?"

   Select the best answer.

   - ❍ A. Restore, then recover, the lost or damaged datafiles
   - ❍ B. Re-run the failing SQL statement
   - ❍ C. Intercept the bad SQL return code in your program and ensure you have coded to handle this situation
   - ❍ D. Restore a valid copy of the damaged control file and then restart the database
   - ❍ E. Startup the database
   - ❍ F. Ensure all network connections are working (including the LAN and WAN connections)

2. What does this RMAN script do? RMAN> run { sql 'alter database datafile 5 offline' ; sql 'alter database datafile 7 offline' ; restore datafile 5, 7 ; recover datafile 5, 7; sql 'alter database datafile 5 online' ; sql 'alter database datafile 7 online' ; }

   Select the best answer.

   - ❍ A. It takes the database offline and performs media recovery of two datafiles.
   - ❍ B. It performs media recovery of two datafiles while leaving the database up and accessible.
   - ❍ C. It checkpoints the datafile headers on two datafiles while leaving the database up and accessible.
   - ❍ D. It takes the database down and checkpoints the datafile headers on two datafiles.
   - ❍ E. It synchronizes SCNs in the datafile headers with the control files for two datafiles.

Answers and Explanations

Chapter 1

1. Answers: A, D, F

Explanation A. This is a correct answer. The Shared Pool caches the most recently issued SQL statements and is a required component of the SGA.

Explanation B. This is not a correct answer. The Large Pool caches data for large operations such as Recovery Manager (RMAN) backup, and restore activities. It is an optional SGA component.

Explanation C. This is not a correct answer. The Java Pool caches the most recently used Java objects and application code, and is used when Oracle's Java Virtual Machine (JVM) option is used. The Java Pool is an optional component of the SGA.

Explanation D. This is a correct answer. The Database Buffer Cache is a required part of the SGA. It caches the data that has most recently been accessed by database users. The Database Buffer Cache makes user access to data faster, because those data blocks in the Database Buffer cache are memory-accessible (thereby avoiding slower disk access).

Explanation E. This is not a correct answer. The Flashback Log Buffer is an optional SGA component. The Flashback Log Buffer is used when the Flashback Database feature is enabled. It caches data that is later externalized (written) to the flashback logs residing in the flash recovery area.

Explanation F. This is a correct answer. The Redo Log Buffer is a required part of the SGA. The Redo Log Buffer stores transaction data for recovery purposes. The Redo Log Buffer data is eventually externalized (written) to the active Redo Logs on disk.

2. Answers: A, C, E

Explanation A. This is a correct answer. The SYSTEM tablespace is always required. Without it, an Oracle database will not start up. The SYSTEM tablespace holds Oracle's Data Dictionary. SYSTEM and SYSAUX are the only two tablespaces that are always created when you create an Oracle database.

Explanation B. This is not a correct answer. The REDO tablespace is not required for an Oracle database. If you picked this answer you may be confusing a "REDO tablespace" with Oracle's required "Redo Logs." The Redo Logs hold transactional data that may be required for database recovery. They do not reside in any so-called "REDO tablespace."

Explanation C. This is a correct answer. The TEMP tablespace is required --assuming the SYSTEM tablespace is locally managed (this is specified by the EXTENT MANAGEMENT LOCAL clause on the definition of the SYSTEM tablespace). The TEMP tablespace is used for sorting data in large sort operations.

Explanation D. This is not a correct answer. The TOOLS tablespace is not required, even though you will find that many or even most Oracle databases have a TOOLS tablespace. When present, TOOLS is used by convention as the storage area supporting various database tools.

Explanation E. This is a correct answer. SYSAUX is a new tablespace that is required in Oracle 10g databases. SYSAUX holds schema objects for various Oracle-provided features (such as XMLDB and the InterMedia option). SYSAUX and SYSTEM are the only two tablespaces that are always created when you create an Oracle database.

Explanation F. This is not a correct answer. The undo tablespace (UNDOTBS1) stores transaction information for read consistency and recovery purposes. It is not required, regardless of whether or not the SYSTEM tablespace is locally managed.

3. Answers: B, C

Explanation A. This is not a correct answer. Non-seed templates are used by Database Configuration Assistant (DBCA) to create databases that have no data. Seed templates are used specifically because you want the new database to contain data copied from an existing database.

Explanation B. This is a correct answer. All templates used by Database Configuration Assistant (DBCA) are XML files. This is true regardless of whether they are seed or non-seed templates.

Explanation C. This is a correct answer. Seed templates copy datafiles and redo logs from an existing database. Non-seed templates are empty of data.

Explanation D. This is not a correct answer. When using a seed template, you can change the database name, the datafile locations, the number of control files and redo log groups, and the initialization parameters.

Explanation E. This is not a correct answer. Oracle 10g also permits you to "clone" or copy databases by using Recovery Manager (RMAN) cloning features.

Explanation F. This is not a correct answer. Database Configuration Assistant (DBCA) controls, manages, and applies seed and non-seed templates. The DBCA main panel allows you to create and manage templates. Recovery Manager uses different techniques to clone or copy databases.

4. Answer: C

Explanation A. This is not a correct answer. One Oracle data block can correspond to one or more operating system blocks. Assuming your database is not using Oracle raw devices (in which case operating system blocks are irrelevant), Oracle data blocks map onto one or more operating system blocks.

Explanation B. This is not a correct answer. Segments reside in only one tablespace, whereas a tablespace can contain more than a single segment.

Explanation C. This is the correct answer. Tablespaces contain segments, each of which consists of extents. Extents consist of a set of contiguous blocks on the media. Assuming your database is not using raw devices, an Oracle block consists of one or more operating system blocks.

Explanation D. This is not a correct answer. Segments contain extents. Extents do not contain segments. An extent is a set of contiguous blocks on the media allocated to a single segment.

Explanation E. This is not a correct answer. The question asks to list the data storage entities from "the largest unit to the smallest." This ordering lists them from the smallest unit to the largest.

Chapter 2

1. Answer: C

Explanation A. This is not a correct answer. This is valid. It tells whether to display optional compile-time warnings. Enable it for easier PL/SQL debugging. Disable it for Production systems.

Explanation B. This is not a correct answer. This is valid. It forces subsequent PL/SQL compilations to be interpreted to include additional debugging information. Enable it for easier PL/SQL debugging. Disable it for Production systems.

Explanation C. This is the correct answer. It is not a valid option. There is no such parameter as PLSQL_OPTIMIZE.

Explanation D. This is not a correct answer. This is valid. It improves the performance of computing-intensive programs. Generally, you would turn this on both for development and production systems.

Explanation E. This is not a correct answer. This is valid. It tells whether to compile the PL/SQL code into the default interpreted byte code or native machine code. Native machine code offers faster execution time at the expense of longer compilation times. You must have a C compiler on your system to use native compilation (Oracle Corp does not supply it).

2. Answer: C

Explanation A. This is not a correct answer. The statement is true.

Explanation B. This is not a correct answer. The statement is true.

Explanation C. This is the correct answer. The statement is false. Database event triggers are indeed specified by the trigger keywords ON DATABASE and fire when any of a half-dozen specified events occur. However, the individual events do not support either BEFORE or AFTER triggers. Each event can only be used with a BEFORE trigger or an AFTER trigger, depending on the specific event.

Explanation D. This is not a correct answer. The statement is true.

Explanation E. This is not a correct answer. The statement is true.

Chapter 3

1. Answers: B, C

Explanation A. This is not a correct answer. The Oracle database automatically detects and resolves deadlocks (not application programs). But application programs can be written to respond to the SQL return code that indicates a deadlock, so that if the program becomes a victim due to a deadlock, that program can take appropriate action.

Explanation B. This is a correct answer. This is the dictionary definition of a deadlock. In Oracle, of course, the "resources" are likely to be rows containing data.

Explanation C. This is a correct answer. If you happened to see this EM panel at the exact moment when the deadlock occurs, you would be able to see it in progress.

Explanation D. This is not a correct answer. Deadlocks can occur within Oracle, but Oracle automatically detects and resolves them.

Explanation E. This is not a correct answer. The session that first detects the deadlock rolls back the statement waiting on the resource. (This differs from other databases for which this statement is true, such as DB2).

Explanation F. This is not a correct answer. Optimistic locking is an algorithm you can employ when writing application programs. If you write all programs to use it, deadlocks will never occur within the database. There is no initialization parameter that somehow "turns on" optimistic locking.

2. Answer: B

Explanation A. This is not a correct answer. The error message ORA-00054 indicates that the SQL statement failed because it required a resource it could not obtain in NOWAIT mode. NOWAIT mode means either the user immediately acquires the resource, or he/she receives the ORA-00054 message. The user will not wait to acquire the resource.

Explanation B. This is the correct answer. NOWAIT means the SQL statement either obtains the lock it requires immediately for the resource in the indicated lock mode, or the error message ORA-00054 is returned to the user.

Explanation C. This is not a correct answer. The SQL statement did not fail due to the kind of lock requested, or the table at which it was directed. As the error message ORA-00054 shows, the statement failed due to its inability to immediately obtain the requested lock on the table specified.

Explanation D. This is not a correct answer. The SQL statement did not fail due to the user having insufficient privileges. As the error message ORA-00054 shows, the statement failed due to its inability to immediately obtain the requested lock on the table specified.

Chapter 4

1. Answer: D

Explanation A. This is not a correct answer. The sequencing of the states is wrong. The proper sequence is NOMOUNT, MOUNT, OPEN.

Explanation B. This is not a correct answer. Even if you STARTUP RESTRICT on an Oracle database, it still passes through the same three states: NOMOUNT, MOUNT, OPEN. The only difference with STARTUP RESTRICT is that when the database achieves the OPEN state, only users with the RESTRICTED SESSION privilege will be able to access it.

Explanation C. This is not a correct answer. It leaves out the required NOMOUNT state. The correct three states in sequence are NOMOUNT, MOUNT, OPEN.

Explanation D. This is the correct answer. A shut down Oracle database always passes through these three steps in starting up and becoming fully accessible to users. NOMOUNT starts the instance but not the database. MOUNT accesses the control files and attaches database structures. OPEN makes the database generally available to users.

Explanation E. This is not a correct answer. STARTUP OPEN is not a database state but a SQL command that starts up an Oracle database. As a result of issuing the STARTUP OPEN command, the database will go through the three states: NOMOUNT,MOUNT, and OPEN.

2. Answers: B, C, D, F

Explanation A. This is not a correct answer. An SPFILE is a binary file that you should never edit. The PFILE is a text file that you can change with a text editor.

Explanation B. This is a correct answer. One of the biggest benefits of using SPFILEs (as opposed to the older PFILE technology) is that you gain the ability to dynamically alter many aspects of the environment.

Explanation C. This is a correct answer. SPFILEs can be created from PFILEs by issuing the CREATE SPFILE FROM PFILE command. This gives you flexibility in moving from PFILEs to SPFILEs.

Explanation D. This is a correct answer. PFILEs can be created from SPFILEs by issuing the command CREATE PFILE FROM SPFILE. This gives you flexibility in moving between SPFILEs and PFILEs.

Explanation E. This is not a correct answer. An active instance works off one configuration file. This may be either an SPFILE or a PFILE, but not both. Of course, you can retain both kinds of files on disk and start the instance using either, but the instance uses only the one you start it with at any one time.

Explanation F. This is a correct answer. This is the correct search order Oracle follows by default when looking for an SPFILE to startup.

Chapter 5

1. Answer: B

Explanation A. This is not a correct answer. System privileges granted using the WITH ADMIN OPTION remain even when their grantor is eliminated. Object privileges granted using the WITH GRANT OPTION are automatically revoked when their grantor is eliminated.

Explanation B. This is the correct answer. System privileges granted using the WITH ADMIN OPTION remain even when their grantor is eliminated. Object privileges granted using the WITH GRANT OPTION are automatically revoked when their grantor is eliminated.

Explanation C. This is not a correct answer. System privileges granted using the WITH ADMIN OPTION remain even when their grantor is eliminated.

Explanation D. This is not a correct answer. Object privileges granted using the WITH GRANT OPTION are automatically revoked when their grantor is eliminated.

Explanation E. This is not a correct answer. There are no CASCADE REVOKE keywords. You determine whether privileges are dropped for SUE by knowing whether they are system privileges or object privileges (assuming the WITH ... OPTION keywords were used when the privileges were granted).

2. Answers: A, C, E

Explanation A. This is a correct answer. Using external authentication, Oracle trusts and relies on the operating system to validate the user. Create a user id validated through external authentication by the keywords IDENTIFIED EXTERNALLY: SQL> CREATE USER myuserid IDENTIFIED EXTERNALLY ;

Explanation C. This is a correct answer. "Global authentication" is a category of authentication that refers to authentication by any of several different services. Use keywords IDENTIFIED GLOBALLY to specify this kind of authentication.

Explanation B. This is not a correct answer. Oracle Security Server can certainly be used for authentication, but in terms of how Oracle Corp categorizes authentication approaches, this is considered a subtype of Global Authentication. It is not referred to as "Oracle Security Server" authentication. SQL> CREATE USER myuserid IDENTIFIED GLOBALLY AS 'proper string here'.

Explanation D. This is not a correct answer. Oracle does not support an authentication method referred to as "local authentication."

Explanation E. This is a correct answer. The database authenticates the user itself by checking the user-supplied password against the one stored internally in the database. Create a password-authenticated user id by using the keywords IDENTIFIED BY, as in the example: SQL> CREATE USER myuserid IDENTIFIED BY mypassword.

Explanation F. This is not a correct answer. The "hostnaming method" is a method of resolving network service names and connection descriptors when clients try to connect to Oracle databases. It is not an authentication technique.

Chapter 6

1. Answer: C

Explanation A. This is not a correct answer. It has PMON sending information about the final result of this process to the Listener (step 4) in the middle of the sequence. The last action that occurs is that PMON sends information about client-to-dispatcher loading to the Listener. This allows the Listener to better perform load balancing.

Explanation B. This is not a correct answer. Step (3) starts the process (this is the client's initial attempt to contact the server). After the client initially contacts the server, the Listener sends info back to the client, redirecting the client to a dispatcher port. So steps (1) and (3) need to be exchanged to make this a correct sequence of steps.

Explanation C. This is the correct answer. The client contacts the Oracle server, and the Listener sends a message back to the client that tells it which port to find a dispatcher on. The client sends a connect attempt to this port, finding its assigned dispatcher. The dispatcher sends an acknowledgment back to the client, and PMON sends info on the client and its assigned dispatcher to the Listener.

Explanation D. This is not a correct answer. It places step (1) at the end of the series. Step (1) is where the Listener initially responds to the client request for a connection. It should be the second step in the process, occurring immediately after the client first asks to connect to the server.

2. Answers: A, C, D

Explanation A. This is a correct answer. This is a good short summary of the purpose of the Connection Load Balancing feature.

Explanation B. This is not a correct answer. This statement accurately describes Connection Load Balancing, not Client Load Balancing. Client Load Balancing distributes the load across multiple listeners by allowing clients to randomly select a listener from a list of listeners.

Explanation C. This is a correct answer. Advanced Oracle features for high availability and performance enhancement require multiple listeners on the database.

Explanation D. This is a correct answer. The statement accurately summarizes what Connect-Time Failover does.

Explanation E. This is not a correct answer. The description of what Transparent Application Failover (TAF) does is accurate, but it is not accurate to say that you can't use it with Real Application Clusters (RAC). In fact, TAF is primarily used with RAC systems.

Explanation F. This is not a correct answer. Dynamic service registration enables an Oracle database to automatically register its presence with an existing listener. You are required to use dynamic service registration to take advantage of advanced features like load balancing and connect-time failover.

Chapter 7

1. Answer: C

Explanation A. This is not a correct answer. This statement is true. A COMMIT always prompts the Log Writer to write to the redo log files on disk.

Explanation B. This is not a correct answer. This statement is true. The Log Writer will write to the redo log files on disk when the redo log buffer within the System Global Area (SGA)becomes one-third full.

Explanation C. This is the correct answer. This statement is incorrect. The Log Writer always writes its records to the online redo log files on disk before the Database Writer writes its records from the database buffer to the datafiles. This is the principle called "write-ahead logging." It is what guarantees the logs always enable you to fully recover a database.

Explanation D. This is not a correct answer. This statement is true. When the redo log buffer accumulates one megabyte of changed (not inserted or deleted) records, the Log Writer will externalize the redo log buffer to the online redo log files on disk.

2. Answers: A, D

Explanation A. This is a correct answer. The instance name is not contained in the control files because the control files contain only information about the physical database and its recovery characteristics.

Explanation B. This is not a correct answer. The control files contain the name of the database. This is essential information about the physical database characteristics.

Explanation C. This is not a correct answer. The control files contain the name of all the tablespaces contained in the database. This is essential information about the physical database characteristics.

Explanation D. This is a correct answer. The control files do not contain any passwords for any user id's. The control files contain only information about the physical database and its recovery characteristics.

Explanation E. This is not a correct answer. The control files do contain the current log sequence number. This is essential information for Oracle to keep track of points of internal consistency and recovery purposes.

Chapter 8

1. Answers: C, E

Explanation A. This is not a correct answer. A smallfile tablespace can have up to 1,022 datafiles-but by definition-a bigfile tablespace is only allowed to have 1 datafile.

Explanation B. This is not a correct answer. The maximum size of a bigfile tablespace depends on the block size you use for the file. For a block size of 8K, the maximum size of the bigfile tablespace would be 32 terabytes. But for other block sizes, the maximum bigfile tablespace sizes will vary.

Explanation C. This is a correct answer. Remember that a bigfile tablespace has 1 datafile. This means that there is only 1 datafile header to update, versus the many that could be required if the same data is placed in multiple datafiles (as in a smallfile tablespace). Thus, bigfile tablespaces offer efficiency for very large databases.

Explanation D. This is not a correct answer. A tablespace is either a bigfile tablespace or a smallfile tablespace. These are mutually exclusive alternatives used in defining a tablespace, and cannot be intermixed. There is no such thing as a "bigfile datafile" or "smallfile datafile." The words "bigfile" and "smallfile" apply to tablespaces, not to datafiles.

Explanation E. This is a correct answer. The purpose of bigfile tablespaces is to provide very large file facilities for Oracle databases, and this requires proper support from the underlying volume manager to work efficiently. Oracle's automatic storage management (ASM) facility can also provide this necessary underlying support.

Explanation F. This is not a correct answer. Oracle 10g requires that the SYSTEM and SYSAUX tablespaces be smallfile tablespaces. You do not convert these tablespaces to bigfiles when upgrading to 10g.

Chapter 9

1. Answer: C

Explanation A. This is not a correct answer. The statement is true. If you do not configure a Large Pool, Oracle places the User Global Area (UGA) in the Shared Pool. Therefore, configure a Large Pool when using Shared Server so that you do not affect the performance of the Shared Pool.

Explanation B. This is not a correct answer. The statement is true. The whole purpose of the Shared Server feature is to gain greater scalability by having each Shared Server manage many user processes. This contrasts the dedicated server architecture, where there is a 1-to-1 correspondence between user processes and server processes.

Explanation C. This is the correct answer. The statement is not true. Oracle Corp emphasizes that the Shared Server facility is a scalability feature, not a performance feature. Shared Server addresses constraints on scaling to large numbers of users. It does not improve performance.

Explanation D. This is not a correct answer. The statement is true. The Shared Server feature puts certain PGA information into UGAs, which are then placed into the Large Pool. Stack space is the only part of the PGA that remains in the PGA in the Shared Server environment.

Explanation E. This is not a correct answer. The statement is true. Shared Server works best for short transactions. Large results sets, extensive data warehousing queries, and queries that generate lots of network traffic are poor candidates for the Shared Server feature.

Explanation F. This is not a correct answer. The statement is true. One limitation of the Shared Server feature is that you can't perform certain kinds of operations from shared connections. Startup, shutdown, and certain kinds of recovery are good examples of options you can't perform.

2. Answer: B

Explanation A. This is not a correct answer. The allowable abbreviations for protocol are prot or pro, not pr. Similarly, the allowable abbreviation for dispatcher is dis or disp, not di.

Explanation B. This is the correct answer. The abbreviation prot is allowed for protocol, and the abbreviation disp is allowed for dispatchers. This code establishes 5 dispatchers for the TCP/IP protocol and one dispatcher for the IPC protocol.

Explanation C. This is not a correct answer. The characters after the dispatchers= parameter must be placed within double quotation marks. The correct coding is: dispatchers="(prot=tcp)(disp=5)(prot=icp)(disp=1)".

Explanation D. This is not a correct answer. The allowable abbreviations for protocol are prot or pro, not pr. Similarly, the allowable abbreviation for dispatcher is dis or disp, not di. Furthermore, the characters after the dispatchers= parameter must be placed within double quotation marks.

Explanation E. This is not a correct answer. The parameter code to start dispatchers is dispatchers, not start_dispatchers.

Explanation F. This is not a correct answer. The parameter code to start dispatchers is dispatchers, not start_dispatchers. Also, the character string that defines the dispatchers should be enclosed within double quotation marks (").

Chapter 10

1. Answer: D

Explanation A. This is not a correct answer. This is a valid option you can select when running the SQL Access Advisor. The SQL Access Advisor can give you feedback about which indexes could reduce the time required to access data.

Explanation B. This is not a correct answer. This is a valid option you can select when running the SQL Access Advisor. The SQL Access Advisor can give you feedback about which materialized views could reduce the time required to access data.

Explanation C. This is not a correct answer. This is a valid option you can select when running the SQL Access Advisor. The SQL Access Advisor can give you feedback about which indexes and materialized views could reduce the time required to access data.

Explanation D. This is the correct answer. SQL tuning sets are a group of SQL statements collected together for tuning analysis. You input SQL tuning sets to the SQL Tuning Advisor -not to the SQL Access Advisor. The SQL Access Advisor provides schema design recommendations, while the SQL Tuning Advisor provides advice on how to rewrite or tune SQL statements.

2. Answers: A, C, E, F

Explanation A. This is a correct answer. This is one of the default ADDM alerts. It is enabled by default for newly-created 10g databases, but it is disabled by default for databases upgraded to 10g from prior releases. The tablespace space usage threshold only applies to locally managed tablespaces (not dictionary-managed ones).

Explanation B. This is not a correct answer. It is not a valid default ADDM alert.

Explanation C. This is a correct answer. This is a default ADDM alert that tells you whenever the ORA-01555 error message occurs. If you receive this alert, one solution is to increase the space allocation in the undo tablespace.

Explanation D. This is not a correct answer. It is not a default ADDM alert.

Explanation E. This is a correct answer. This default ADDM alert is raised whenever the flash recovery area is low on space.

Explanation F. This is a correct answer. This is a default ADDM alert that alerts you when an operation that can be resumed goes into a suspended state.

Chapter 11

1. Answers: A, D, F

Explanation A. This is a correct answer. Image copy backups cannot be written to tape; only RMAN backup sets can be written to tape.

Explanation B. This is not a correct answer. Image copies cannot be compressed; only RMAN backup sets can be compressed. Image copies always require more storage space than compressed RMAN backup sets, but of course image copies are quicker to restore because they do not have to be uncompressed to be used.

Explanation C. This is not a correct answer. RMAN backup sets can be written to either tape or disk. Image copies, on the other hand, can only be written directly to disk (not to tape).

Explanation D. This is a correct answer. RMAN writes backup sets in its own internal format. You cannot meaningfully view them or edit them.

Explanation E. This is not a correct answer. RMAN backup sets are written in files called "backup pieces."

Explanation F. This is a correct answer. Oracle backups (whether image copies or backup sets) can be written to the flash recovery area.

2. Answer: D

Explanation A. This is not a correct answer. Flashback database is best for rolling back an entire database to some prior point in time. Since it puts the entire database back to some previous point in time, you would not use it just to view the contents of one table at some prior point.

Explanation B. This is not a correct answer. Flashback drop allows you to easily recover a dropped table. You would not normally use it to view the contents of a table at a prior point in time. It is a recovery tool for dropped tables.

Explanation C. This is not a correct answer. Flashback table recovers a table to some prior point in time. This would allow you to view the contents of the table at some previous point in time -but it also changes the contents of the table. The question does not ask you to change your table's current contents; it just asks to view those contents at some earlier point.

Explanation D. Flashback query is the best way to view a table's contents at some earlier point in time. You just do a regular SQL query, but add to the query the keywords AS OF TIMESTAMP. This retrieves the table data from some previous point in time without the need to permanently change or recover the table's contents.

Explanation E. This is not a correct answer. Viewing table contents at an earlier point in time is precisely what flashback query is designed for. You could perform an RMAN restore/recover to view table data from some previous point in time, but the process is time-consuming and complex, plus it permanently changes the data of one or more tables.

Explanation F. This is not a correct answer. Viewing table contents from an earlier point in time is precisely what flashback query is designed for. You could perform an RMAN TSPITR recovery to view table data from some previous point in time, but the process is time-consuming and complex, plus it permanently changes the data of one or more tables.

Chapter 12

1. Answer: E

Explanation A. This is not a correct answer. This statement describes how to perform "media recovery" (the kind of recovery you perform when a disk drive fails or datafiles are lost from a database). It does not describe how to handle crash recovery for the database.

Explanation B. This is not a correct answer. This describes how you might get a SQL statement that failed to work (under certain circumstances). It does not describe how to handle crash recovery for the database.

Explanation C. This is not a correct answer. This describes how a program responds to a SQL statement error. It does not describe how to handle crash recovery for the database.

Explanation D. This is not a correct answer. This describes how to recover from a situation where one of the control files has gone missing or become damaged. It does not describe how to handle crash recovery for the database.

Explanation E. This is the correct answer. "Crash recovery" is also known as "instance recovery." Oracle performs this kind of recovery, if needed, automatically when you start the instance. Oracle uses the online redo log files and undo data in the undo tablespace to automatically perform crash recovery when it is deemed necessary.

Explanation F. This is not a correct answer. This describes how you might fix a network error. It does not describe how to handle crash recovery for the database.

2. Answer: B

Explanation A. This is not a correct answer. The commands in this script take only two datafiles offline, not the entire database.

Explanation B. This is the correct answer. The script takes two datafiles offline. Then, it restores them and recovers them. Finally, it alters the two datafiles back online. This is called "media recovery," and is used when non-critical datafiles become lost or damaged.

Explanation C. This is not a correct answer. The purpose of the script is not simply to checkpoint the datafile headers. The RMAN restore and recover commands indicate that the purpose of this script is to perform media recovery on two missing or damaged datafiles.

Explanation D. This is not a correct answer. The purpose of the script is not to checkpoint the datafile headers. The RMAN restore and recover commands indicate that the purpose of this script is to perform media recovery on two missing or damaged datafiles.

Explanation E. This is not a correct answer. The script performs media recovery of two datafiles. It is true that as a result of this process it brings the SCNs in the datafile headers up-to-currency with the rest of the database (this is part of the definition of media recovery).