Exam
Manual

LearnSmart

(1Z0-052) Oracle 11g Administration I

This LearnSmart exam manual covers the most important concepts you need to master in order to successfully complete the Oracle 11g Administration 1 exam (1Z0-052). In this manual, you will find content related to the following exam objectives:

- Exploring the Oracle Database Architecture
- Preparing the Database Environment
- Administering User Security
- Managing Schema Objects
- And more!

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

Oracle DBA 11g: Administration I (1Z0-052) LearnSmart Exam Manual

Product ID: 12050
Production Date: July 14, 2011

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@learnsmartsystems.com

International Contact Information

International: +1 (813) 769-0920

United Kingdom: (0) 20 8816 8036

Table of Contents

Abstract

This Exam Manual prepares you for the Oracle 11g certification exam #1Z0-052, "Oracle Database 11g: Administration I." Passing this exam plus one other makes you an Oracle 11g Certified Associate (an "OCA").

This Exam Manual is your "Cliffs Notes" key to the test material. It is a short, highly-targeted summary of what's on the exam. It can't cover everything, but if you understand and memorize its contents, you will be very far along towards passing the exam.

This exam covers Oracle architecture and components, installing Oracle and configuring its networking, creating and managing databases, database storage structures and schema objects, security and administering users, data integrity and concurrency, database maintenance, performance management, data movement, and backup and recovery.

What to Know

This exam covers everything pertaining to basic database administration of Oracle databases. You need to know --

- Oracle's architecture, the memory structures and processes in an instance, and the physical structures of a database.
- How to plan for an Oracle installation, install the Oracle software, and create databases with the Database Configuration Assistant. This includes configuring Oracle and its network environment.
- How to manage Oracle on a daily basis, including how to start and stop Oracle, use its dictionary and performance views, and read its log.
- How to administer Oracle security. Know about user accounts, privileges, roles, and profiles. This includes the principle of Least Privilege and security auditing.
- Understand physical storage in tablespaces and how space management works. Know about schema objects including tables (and constraints), indexes, views, and temporary tables.
- Know how data updates and concurrency work. This includes locking, locking conflict resolution, and how undo data supports transactions.
- Know about database maintenance through optimizer statistics, the advisory framework, AWR (Automatic Workload Repository), and thresholds and alerts.
- Know the basics of performance management with Automatic Memory Management (AMM) and the Advisors.
- Understand Oracle's tools to move data including Oracle Data Pump, SQL*Loader, and external tables.
- Three sections of the test cover Oracle backup and recovery. Understand concepts such as why databases fail, checkpoints, redo logs, archive logs, the flash recovery area, and ARCHIVELOG mode.
- Understand the ways to back up databases including consistent, inconsistent, and incremental backups.
- Know what the Data Recovery Advisor is and how to use it to recover control files, redo log files, and data files.

Tips

- The test consists of multiple-choice questions with one correct answer. It also contains multiple-choice questions where you select 2 or 3 correct items out of a list of 5 or 6 possible answers.
- For multiple-answer questions, you must select every correct answer and none of the incorrect answers in order to get credit for correctly answering the question. There is no "partial credit."
- Be sure to answer every question -- there is no penalty for guessing.
- Read each question carefully and read all alternatives before picking an answer. Often the differences between answers are very slight and you don't want to pick a wrong answer just because you didn't read the question carefully.
- The answers to many questions will not be immediately evident. In this case it helps to use a process of elimination. Eliminate answers you know are wrong, and you'll often then be able to take your best guess from the remaining couple alternatives.
- You can always mark a question you're not sure about for later review.
- It is 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:
You need to verify the exam requirements at Oracle Corporation's web site because the company reserves the right to change them at any time. Usually they only change the tests when they come out with a new version of the Oracle database, but on occasion they do make minor modifications to the exams between releases.

To view the exam requirements, go to Oracle's certification home page here. (Sometimes Oracle redirects incoming links so you may encounter a web page where you select your country first. Then select "Certification" from the menu on the resulting page. Make sure your browser accepts cookies.)

Hands-on Experience:
You can either get hands-on experience with Oracle 11g at work, or you can install the free version of Oracle database on your personal computer. The free version of Oracle is called "Oracle Express." You can download it to run on either Windows or Linux.

Download Oracle Express Edition from www.oracle.com and enter the words "oracle express download" in the Search Box.

Oracle Express Edition on your personal computer allows you to work with Oracle SQL and can provide sufficient experience to pass this exam. At the time of writing, 10g is the latest version of Oracle Express you can download – but this is perfectly fine for the purposes of this particular exam.

Books:
Search at Amazon online and you will find several books that are written specifically to help you pass this exam. Read reviewers' comments to select which you prefer. Or go to a bricks-and-mortar bookstore and peruse the books yourself. It is important to select a book you like and have confidence in… you will be spending a lot of time studying it.

Here are two popular certification books for this exam –

OCA: Oracle Database 11g SQL Fundamentals I Exam Guide: Exam 1Z0-051. By John Watson and Roopesh Ramklass (Osborne Oracle Press Series).

- OCA Oracle Database 11g Administration I Exam Guide: Exam 1Z0-052. By John Watson (Osborne Oracle Press Series).
- OCA: Oracle Database 11g Administrator Certified Associate Study Guide: (Exams 1Z0-051 and 1Z0-052). By Biju Thomas (Sybex). This book covers both the exams you need to become an Oracle Certified Associate.

Practice Questions:
Practice questions get you ready to face the real exam. They give you a feel for what the tests are like, and they also help you conceptualize your knowledge of Oracle in terms of the kinds of questions you'll face in the real exam.

We recommend the practice tests available from LearnSmart. LearnSmart has earned a reputation for quality practice tests that prepare candidates well for certification exams.

Oracle Corporation offers ten free sample questions at their certification web site. These questions show you what the test questions look like -- but the sample is far too small to help you learn what you need to know to pass the exam.

1.0 Exploring the Oracle Database Architecture

An Oracle **relational database management system** or **RDBMS** consists of two key components:

1. The Instance
2. The Database

An **instance** consists of **memory structures** and **background processes**. When you start up a database, Oracle starts the instance, and then the instance mounts and connects to or "opens" the database. While the instance is transitory in computer memory, the **database** consists of permanent disk files.

There can be a one-to-one relationship between an instance and a database, or there can be a many-to-one relationship – called a **Real Application Cluster** or **RAC** configuration. RAC clusters Oracle databases and provides for *scalability*, *performance*, and *fault tolerance*.

Oracle instances and databases can also be related by **Streams**. Streams propagate transactions between database servers. Propagation can be *uni-directional* or *bi-directional*. Streams provides for *fault tolerance* and *performance tuning*.

Data Guard relates a *primary database server* to one or more *standby databases*. Standby databases can be **physical standbys**—byte-for-byte copies of the primary database, kept synchronized by the application of *change logs* or *redo logs* shipped to and applied against the standby. Or they can be **logical standby's**, kept synchronized by SQL statements propagated to the standby through Streams.

When a user connects to Oracle, his or her **client-side system** runs a **user process**. This user process on the client computer connects across a network or the internet to a **server process** running on the database server. This server process then runs SQL statements against the database and returns its results across the network to its user process partner on the client computer. In a **three-tiered architecture**, user processes may actually be programs running on an **application server** such as **Oracle Application Server**. Regardless, each server process has its own non-shared memory area called the **Program Global Area** or **PGA**.

A single server process can handle:

- One user process (called *Single-Server architecture*)
- More than one user process concurrently (called *Shared Server architecture*)

1.1 Instance Memory

The **Instance** consists of two components:

1. Memory structures
2. Background processes

The main memory allocation 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 three required components and three optional components. The three required SGA components are listed in **boldface** below, while the three optional components are in regular typeface:

SGA Component	Use
Database Buffer Cache	Keeps the most-recently referenced data in memory
Shared Pool	**Library Cache**: Stores recently-executed code **Data Dictionary Cache**: Stores data dictionary information **PL/SQL Area**: Stores recently used PL/SQL objects **SQL Query & PL/SQL Function Result Cache** (new in 11g): Stores results of recent SQL and Function executions
Redo Log Buffer	Keeps transaction log information in memory (also known as **redo records**)
Large Pool	Caches data for intensive operations like **Recovery Manager** or **RMAN** and Shared Server
Java Pool	Caches most-recently used Java objects and code
Streams Pool	Caches queued message requests for Oracle Streams

The SGA, the Database Buffer Cache, Shared Pool, Large Pool, Java Pool, and Streams Pool are dynamic in size and can be automatically managed by Oracle. *The Redo Log Buffer pool is static and fixed in size at start up.* It cannot be automatically managed.

Caching means 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** or **LRU** algorithm to determine what information to keep in memory areas like the Database Buffer Cache and the Shared Pool For the Database Buffer Cache, this minimizes data access from disk; for the Shared Pool, it reduces redundant SQL execution and disk access.

Oracle can automatically manage the sizing of the SGA for you (recommended) or you can do it yourself manually. 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.

1.2 Background Processes

An **instance** consists of *memory structures* and *background processes*. Let's discuss the background processes now.

Each Oracle **background process** handles one or more functions for the oracle database system. Five are required, while many others are optional and may or may not be present depending on what Oracle features you use.

The *five required* background processes are:

Name:	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 or **dirty** data blocks from the **Database Buffer Cache** in SGA memory to the datafiles on disk
CKPT	Checkpoint	Manages **Checkpoints** by updating database files. Read about what checkpoints do below

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

You'll need to know what these background processes do and how their work fits together to manage queries and database transactions. We'll explain all this in detail as we continue.

The many *optional* background processes include the following:

Name:	Background Process	Use
MMAN	Memory Manager	Automatically manages and sizes key memory-using components for 11g's **Automatic Memory Management** feature.
MMON	ManageabilityMonitor	Gathers/analyzes performance statistics used by the **Automatic Workload Repository** or **AWR.**
MMNL	MMON 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 **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 Architecture* is discussed later).
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 NOARCHIVELOG mode databases.
RECO	Recoverer	Recovers failed **distributed transactions** – transactions that span two or more databases.
CJQn	Job Queue Monitor	Assigns jobs to the **Job Queues** for the 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.**
DBRM	Database Resource Manager	Manages resources for Oracle's **Resource Manager.**

The Windows operating system uses **threads** instead of **background processes** like Unix and Linux—but Oracle Corp. stills refers to them as "background processes", so that's the terminology you need to memorize. The Windows Service **OracleService*Instance_Name*** is associated with each instance. This Service must be started to start up the instance. Each listener running under Windows is also a separate Windows Service. Under Windows, it is possible to have the database Windows Service up and running while the database itself has not been started or is down.

Vital points to remember about the background processes:

- SMON **mounts** and **opens** the database and performs **crash recovery** (automatic transaction recovery upon start-up)
- PMON cleans up after an aborted transaction -- it performs **rollback**.
- DBWn writes when:
  - There are no free buffers
  - There are too many dirty pages in the buffers
  - At 3-second timeouts
  - At a **Checkpoint** event
  - The instance shuts down in an orderly manner
  - A tablespace changes status (into *Backup Mode*, *Offline*, or *Read Only*)
- DBWn does **not** write because of a completed transaction or **COMMIT !**
- LGWR flushes the log buffer to disk when:
  - A COMMIT occurs (a **COMMIT** successfully completes a transaction)
  - The buffer is 1/3 full
  - Just before any time that DBWn writes
- ARCn reads the online log files that LGWR writes
- CKPT only does **full checkpoints**:
  - On request
  - At an orderly database shutdown
- CKPT continually updates Oracle's **control files** with current checkpoint position
- MMON gathers a statistics snapshot and launches **Automatic Database Diagnostic Monitor** or **ADDM** *every hour by default*

1.3 Database Files

While the **instance** consists of transient memory and background processes, the **database** consists of persistent files on disk. The three key types of files that are *required* to be present in the database are:

- **Datafiles** - They hold the data.
- **Control Files** - These 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.
- **Redo Log Files**– These contain *change records* ("redo" information) used to ensure against data loss. For example, they can be applied during database recovery to bring damaged datafiles up-to-date with the rest of the database.

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, such as sorting data). *Each datafile belongs to one and only one tablespace; a tablespace may own one or more datafiles.*

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 processes write updated (**dirty**) pages from the database buffer cache to the datafiles to keep those datafiles accurate and externalize data changes to disk storage.

Server processes read from the datafiles while DBWn writes to them. No other processes touch the datafiles under normal conditions.

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 terminology, 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 ensure a system-wide point of data consistency. It does this using **System Change Numbers** or **SCNs**, the unique, ascending numbers assigned to transactions. A common SCN between the control files and all datafile headers indicates a database-wide point of consistency. Any datafile with a differing SCN is not in sync or current with the rest of the database.

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!* (Example: you run SQL statements that create or change tablespaces or datafiles). You can set Oracle to ensure that all database backups include the controlfile through a feature called **control file autobackup**.

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) processes update its archive log name and log sequence number.

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.*

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, writing to one after another. Whenever LGWR points to a new redo log group to write to, this is called a **log switch**. A log switch does not cause a checkpoint.

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 over-written. 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** write 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 over-written. *Should there be no disk space available to create the archived redo log file, the database will hang until space becomes available.*

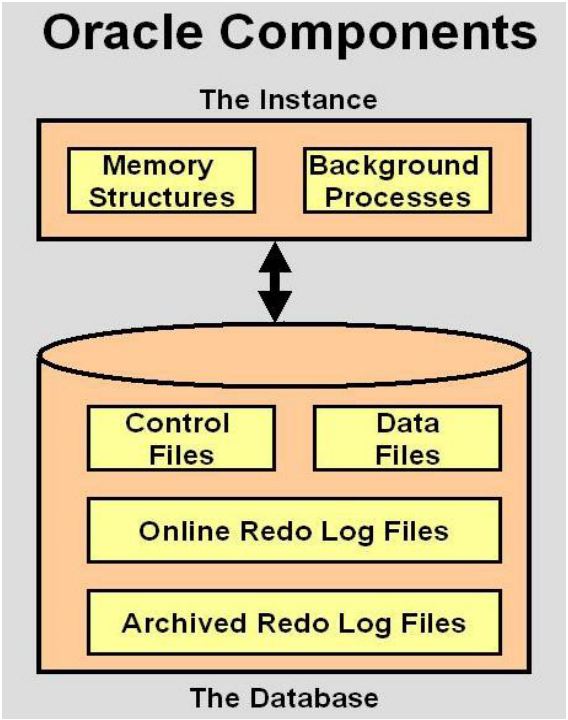
Online redo log files that are currently being used (written to) are termed the **current log files**. They are **active**. The online redo log files that are not presently being written to are **inactive**.

Redo records are essential to recover all database transactions in the event of a database or media failure. They keep track of database changes so that none are ever lost. For this reason, Oracle databases that require complete recoverability should be run in Archivelog mode.

Other Files – these may also be present in the database:

- **Parameter File** – Oracle starts & configures an instance based on these parms
  - **PFILE –** a static, textual, editable form of parameter file
  - **PFILE** – a dynamic, binary form of parameter file
- **Password File** – Authenticates users without using Oracle's Data Dictionary
- **Archive Redo Log Files** – ARCn copies **Online Redo Log files** to these to permanently keep a record of all database changes
- **Alert Log** – The essential log of database and instance events
- **Trace Files** – Data from background processes for info on errors or events

This simple diagram summarizes the key components of the Oracle database and its instance(s):



1.4 The Data Dictionary

Oracle's **Data Dictionary** (D/D) is a set of control tables for the system. They hold **metadata**—data *about* data. The D/D is stored in two key *required* **tablespaces**—**SYSTEM** and **SYSAUX**. Three sets of views allow users to interrogate the D/D:

- **DBA_** Info about every object in the database
- **ALL_** Info about all objects in the database you have permission to see
- **USER_** Info about all objects in the database you own

So to view information about all tables in the database, you would query the ***DBA_*** views (not the ***ALL_*** views). The D/D is created by running a script during installation. It is owned by user id **SYS**.

In contrast to the Data Dictionary tables, the **dynamic performance views** or **V$ views** are populated from the instance or the control file. They materialize upon startup and disappear upon shutdown. They are not permanent like the Data Dictionary tables. V$ views hold performance information from the time of startup. Their names begin with **V$** and are usually singular. D/D views usually have plural names. There is some overlap in the information contained in the V$ and D/D views.

2.0 Preparing the Database Environment

Oracle Universal Installer or **OUI** is the tool for installing Oracle software and, optionally, for creating a database. It invokes the **Database Configuration Assistant** or **DBCA** to optionally create and configure a database. Both tools are graphical and require Java. OUI installs a **Java Runtime Environment** or **JRE** and also a Perl interpreter for its use. Check for Java by issuing this command to the operating system**:**

```
java -version
```

Oracle Base is the root-level directory for Oracle installs. It contains one or more **Oracle Home** directories within which different releases and products are installed. The **Oracle Inventory** directory or folder keeps track of all Oracle products and their versions installed on the computer. It is usually in an entirely different location from Oracle Base and Oracle Home.

Run OUI by **setup.exe** under Windows or the **runInstaller.sh** script under Unix or Linux. Environmental variables need to be set to run OUI and its products. Various options are available for where to set these environmental variables. Windows usually uses the **Registry**. Under Unix or Linux, set them for all users in the **global login profile** (eg: **/etc/profile**).

Key installation procedure files to remember are:

- **products.xml** - Lists all the products that can be installed off a DVD
- **oraparam.ini**- System requirements, also points to **products.xml**
- **oraInst.loc** – Points to the **Inventory** in Unix/Linux
- **inst_loc** - Registry key that points to the **Inventory** in Windows
- **setup.exe**- Runs OUI under Windows

- **runInstaller.sh** - Runs OUI for Unix/Linux. Here are three key options:
    1. **ignoreSysPrereqs- Ignores install prerequisite checks**
    2. **responsefile- Uses a** *response file* **for** *silent installs*
    3. **destinationFile- Creates a response file for silent installs**
- **orainstRoot.sh** - Run this script as prerequisite to installing under Unix/Linux
- **root.sh**- Run this script to adjust Unix/Linux permission bits on Oracle Home

Key **environmental variables** you must set to install and use Oracle:

Variable	Use
ORACLE_BASE	Top level directory for installing all Oracle products and versions.
ORACLE_HOME	Top level directory for an individual Oracle product install. Can support several databases of the same product version and release.
PATH	Directory path to the Oracle installed software.
LD_LIBRARY_PATH	Directory path to the Oracle installed libraries for Unix & Linux.
DISPLAY	Sets display characteristics for use of graphical interfaces. Required for Unix and Linux, not used for Windows. *You **must** have this variable set correctly under Unix/Linux prior to running the graphical installer (OUI).*

Optimal Flexible Architecture or **OFA** is Oracle's recommended, default set of directories or folders for where to install Oracle and its database files. Among OFA naming conventions are:

File Type	OFA Name	Example
Controlfile	controlNN.ctl	control01.ctl, control02.ctl
Redo log files	redoNN.log	redo01.log, redo02.log
Datafiles	tablespace_nameNN.dbf	system01.dbf, my_tablespace.dbf

Key client-installed tools included with Oracle distribution are:

1. SQL*Plus - Text-based window for SQL, PL/SQL, and SQL*Plus commands
2. SQL Developer - Graphical tool for the same purposes as SQL*Plus but with wizards

While SQL Developer requires Java, SQL*Plus does not. iSQL*Plus is no longer part of the Oracle distribution. To use either SQL Developer or SQL*Plus you need:

1. User name (your **user id** or **user account**)
2. Password
3. **Connect identifier** (this identifies the Oracle database you want to interact with)

Other key Oracle tools include:

Tool	Use
Oracle Universal Installer (OUI)	Graphical tool for install of Oracle software and optional creation of a database. Can launch DBCA, network configuration tools, etc.
Database Upgrade Assistant (DBUA)	Like DBCA but for upgrading existing databases rather than installing and configuring new ones.
Oracle Enterprise Manager (OEM)	Oracle's graphical tool for managing all its products and the Oracle environment, sometimes just called **EM**.
OEM Database Control	OEM tool to control and manage an individual database.
OEM Grid Control	OEM tool to control and manage entire an network of databases and their supporting tools and products.
OEM Application Server Control	OEM tool to control and manage Oracle Application Server.
Oracle Application Server (OAS)	A J2EE-compliant application server product from Oracle. It is a completely separate product from Oracle RDBMS.
Net Manager	Graphical tool for Oracle Net network management.
Net Configuration Assistant	Another graphical tool for Oracle Net network management. Some overlap with the Net Manager.
Recovery Manager (RMAN)	Oracle Corp.'s product for Oracle database backups and recoveries.
Oracle Secure Backup	For backup of the entire Oracle environment including Oracle Databases, Oracle Application Servers, operating systems, etc.
Oracle Enterprise Linux (OEL)	A Linux operating system supplied and supported by Oracle Corporation. Based on Red Hat Enterprise Linux.
SQL*Loader	The Oracle data load utility. Works with all kinds of input files including those dumped out by other database products and operating system "flat files."
Data Pump (or Datapump)	Primary tool for dumping & reloading Oracle table data. New in 10g. Uses its own proprietary data file format. Cannot use the file format of the older Export/Import utilities. Uses **background processes** to run.
Export/Import Utilities	Primary tool for dumping & reloading Oracle table data prior to the introduction of Datapump in Oracle 10g. Uses its own proprietary data file format (not Datapump compatible). Uses **server sessions** to run, therefore less efficient than Datapump's background processes.

Many Oracle tools are written in Java and require Java runtime. Examples include OEM, OUI, DBCA, DBUA, Net Manager, Net Configuration Assistant, Oracle Application Server, SQL Developer, and others.

3.0 Creating an Oracle Database

To create an Oracle database the steps are:

1. Install the Oracle software (the "Oracle binaries")
2. Create the instance
3. Create the database
4. Create the Data Dictionary objects
5. Create the Data Dictionary views

3.1 Database Configuration Assistant (DBCA)

Use DBCA to perform all the steps above, except for step one, installing the Oracle binaries. Before launching DBCA you must have properly set environmental variables. These include **ORACLE_BASE**, **ORACLE_HOME**, **PATH**, and for Unix/Linux **LD_LIBRARY_PATH** and **DISPLAY**. If you don't set **DISPLAY** properly under Unix/Linux, DBCA can't use your monitor in graphical mode and will abort.

If you are going to use OEM Database Control to manage your database, you should also configure a *database listener*. DBCA checks whether one is available for a Database Control connection to the database. Use the Net Configuration Assistant tool with all defaults to quickly and easily create a listener. Major decisions DBCA will confront you with during database creation include:

- **Database Templates** – Whether to create your database based upon one of these models
- **Database Identification** – Name your database via parameter **DB_NAME** and its *System Identifier* or **SID**
- Select whether you'll use *Database Control* or *Grid Control* to manage the database. Grid Control will only be available if a **Grid Control Agent** is detected running on the machine. (*Note- the exam and this Exam Manual focus only Database Control. Grid Control offers all the exact same GUI panels as Database Control plus more*.) The difference between Database Control and Grid Control is that the former manages a single database while the latter manages many databases as well as the larger environment.
- You can choose to store database data in any of four basic ways:
    1. *File System files* - The regular filesystem provided by your operating system
    2. *ASM* - Oracle's **Automated Storage Management** instance
    3. *Raw Devices* - Chunks of disk storage written to directly by Oracle
    4. *Clustered File Systems* - Disks mounted concurrently to more than one computer
- Whether you want to create a **Flash Recovery Area** for easy database backup/recovery
- Whether you want DBCA to write the database creation scripts for you to run later, or whether you want these scripts run immediately from within DBCA

DBCA automatically creates a number of user ids or **user accounts** for the new database. They include:

User Id	Use
SYS	Creates and manages the Data Dictionary
SYSTEM	Power DBA user id used for most DBA work
DBSNMP	Used for external monitoring
SYSMAN	Used by OEM for database management
others	Other less critical user ids for management purposes

The scripts DBCA creates (and optionally runs) for you include:

Script	Use
sql.bsq	Creates the data dictionary by invoking other scripts
cat*.sql	Several scripts run immediately after database creation
catalog.sql	Creates the data dictionary objects and views
catproc.sql	Creates internal oracle PL/SQL procedures and packages
CreateDB.sql	The database creation script
ocp11g.sql	Runs many post-database creation scripts, such as those below
CreateDBfiles.sql	Creates the **USER** tablespace that is used as a default for unassigned user ids
CreateDBCatalog.sql	Creates views on the data dictionary and PL/SQL packages
emRepository.sql	Creates objects required by OEM Database Control
postDBCreation.sql	Generates a **Server Parameter file** from the **init.ora** or textual **parameter file.** This file configures an oracle database when you start it up. This script also runs **OEM Configuration Assistant** to configure Database Control

The **CREATE DATABASE** SQL statement itself is run from the **CreateDB.SQL** script. It has dozens of parameters, but all have defaults. So you could actually run just:

CREATE DATABASE ;

and have a valid database. CREATE DATABASE creates several required Oracle data files. At a minimum these include a controlfile, two online redo log files, and two datafiles for the **SYSTEM** and **SYSAUX** tablespaces. The SYSTEM and SYSAUX tablespaces hold the data dictionary and other system components.

The CREATE DATABASE command also can define a default temporary tablespace for all users, the undo tablespace, online redo log files, the character set, and set initial passwords for special user ids **SYS** and **SYSTEM**.

The **init.ora** file is a **parameter file** or **PFILE** that configures an Oracle database when you start it. It is a list of keywords and values you can directly edit with any text file editor. There are about 300 parameters you can code. You'll need to be familiar with the several dozen most important for the exam.

While you can still use an init.ora file in 11g, Oracle has largely replaced it by the **Server Parameter file**. This is a binary file that you cannot edit. It is generated from the init.ora file. The benefit of the **SPFILE** is that it allows dynamic alteration of most of the parameters. Using init.ora, you can only change parameters by shutting down the database, editing the file, and re-starting the database.

All parameter file keywords have default values except one – the *database name* or **DB_NAME**. The DB_NAME can be up to 8 characters long and contains only letters and digits. It must begin with a letter.

You can change all the parameters after you create a database except one --the *database block size* or **DB_BLOCK_SIZE**. This defaults to 8K which is acceptable for most applications. Thus you can alter any database characteristics you want after creating the database – even the character set -- but you cannot alter its block size once you've created it.

3.2 Manually Creating A Database

An alternative to DBCA is to manually create a database. DBCA is easier, more automated, and less error-prone, plus it gives you **templates** on which you can model your database. But manual creation give you greater control, which is sometimes necessary for creating large databases or databases with special requirements.

You create a database manually by issuing the **CREATE DATABASE** command through a SQL*Plus session. Additionally you have to run the various creation scripts listed in the table above. The steps may be summarized as:

1. Install the Oracle software (the Oracle binaries)
2. Create a textual parameter file (the configuration file or **PFILE**)
3. Use the parameter file to start an instance
4. Use the instance to create the database via SQL*Plus
5. Run script(s) to create the Data Dictionary objects
6. Run script(s) to create the Data Dictionary views
7. Run script(s) to install any options

4.0 Managing the Oracle Instance

Start up a single-instance database by these steps:

1. Start OEM *Database Control* - The single-database graphical interface control tool
2. Start the database *Listener* - It listens for in-coming database connections
3. Start the *Database* - The instance and the datafiles on disk

4.1 Database Control

Database Control or **DBC** can communicate using **HTTPS**, the secure internet protocol, so you can remotely and securely administer Oracle databases from any browser. Each instance of DBC controls *one database only* and requires *one port* to connect in on (versus **Grid Control** that manages multiple databases). Three environmental variables must be set to manage DBC: **ORACLE_HOME**, **ORACLE_SID**, and **PATH**. Here are the DBC control commands:

```
emctl   start   dbconsole
emctl   stop    dbconsole
emctl   status  dbconsole
```

To connect using DBC you need to know the *host name* and TCP *port number*:

HTTPS://*host_name*:*port_number*/EM

4.2 The Listener

The Oracle **listener** is a background process that monitors a port for incoming database connections. It communicates through Oracle's communcation protocol, **Oracle Net**. Control the listener through any of these means:

- The **lsnrctl** command line utility
- Database Control
- The Services Panel – under Windows only

The default name of the listener is **LISTENER**. Here are common **lsnrctl** commands:

```
lsnrctl   start    [listener_name] - Starts the named listener or LISTENER
lsnrctl   stop     [listener_name] - Stops the named listener or LISTENER
lsnrctl   status   [listener_name] - Status for the named listener or LISTENER
```

4.3 Database Startup and Shutdown

You can start the database through OEM Database Control or by logging into SQL*Plus. There are several ways you can be authenticated to manage the database, as shown by how you login with SQL*Plus:

SQL*Plus Command	**Authentication Type**
sqlplus /nolog	**No authentication**—NOLOG means no database connection attempt. This gets you into a SQL*Plus session without a database connection attempt.
connect / as sysdba connect / as sysoper	**Operating System authentication**—login is verified by membership in the OS group that owns the Oracle software.
connect user/pass[@connect] as sysdba connect user/pass[@connect] as sysoper	**Password File authentication**—user id and password verified by the Oracle **password file** (a standard OS file external to the Oracle database).
connect user/pass[@connect] connect user/pass[@connect]	**Oracle Data Dictionary authentication**—requires an open database, uses D/D oracle security.

SYSDBA and **SYSOPER** are *not* user ids. They are privileges that can be granted. In Oracle 11g, only user id **SYS** has these privileges by default until they are granted to other user ids.

Starting a database with the **STARTUP** command passes it through these three stages in this order:

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

There are several kinds of database shutdown, based on the **SHUTDOWN** command parameters:

SHUTDOWN [NORMAL | TRANSACTIONAL | IMMEDIATE | ABORT] ;

SHUTDOWN Parm	Use
NORMAL (default)	No new database connections are allowed. Only shuts down the database after all users have voluntarily disconnected. Of little practical use, since it waits for every user to log off.
TRANSACTIONAL	No new connections. In-flight transactions complete normally, no new transactions are started. Database shuts down after all sessions end.
IMMEDIATE	No new sessions, current sessions are terminated. Active transactions are rolled back, then the database is shut down.
ABORT	Immediately cuts off the database. No data written to disk, no file handles closed. Requires *automatic instance recovery* by SMON upon next start up (also known as *crash recovery*).

4.3 Database Initialization Parameters

To control how an Oracle database is configured, Oracle provides a group of **initialization parameters** or *init parms*. There are about 30 **basic parameters**, parms for short, that you will usually set, while there are about 300 **advanced parameters** (parms that you rarely need to set). There are also over 1,500 **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 the undocumented parameters for the Exam.

You can view all instance parameters either through OEM Database Control or through the views **V$PARAMETER** and **V$SPPARAMETER**.

Here are the basic parameters you need to know for the Exam:

Basic Initialization Parm	**Use**
BACKGROUND_DUMP_DEST	Gives the location for the **Alert Log** event file (if not over-ridden by setting **DIAGNOSTIC_DEST**)
CLUSTER_DATABASE	Yes for a database driven by more than one instance (a **Real Application Cluster** or **RAC** system)
COMPATIBLE	Allows an instance to run as if it were an older version of Oracle software
CONTROL_FILES	Names and locations of the mirrored control files
DB_BLOCK_SIZE	Default block size for data files. ***This is the only parm that cannot be changed once a database is created!***
DB_CREATE_FILE_DEST	If using **Oracle-Managed Files** or **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. *This is the only parm that has no default value*
DB_RECOVERY_FILE_DEST	For using the **Flash Recovery** feature, provides the location where recovery files reside
DB_RECOVER_FILE_DEST_SIZE	For using the **Flash Recovery** feature, specifies the size of the recovery file area
DB_UNIQUE_NAME	Globally-unique database name in the organization
DIAGNOSTIC_DEST	Directory used by 11g's new **Automatic Diagnostic Repository** or **ADR** problem reporting and solving software. Overrides pre-11g dump destinations if coded (such as **BACKGROUND_DUMP_DEST** and **USER_DUMP_DEST**)
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
MEMORY_MAX_TARGET	Maximum amount of memory for Oracle 11g that can be dynamically assigned. Increasing this parm requires shutdown and restart of Oracle
MEMORY_TARGET	Amount of memory for Oracle 11g. 11g's **Automatic Memory Management** or **AMM** will automatically and dynamically use this memory for SGA and PGAs as required for optimum performance
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** or **PGA.** *Recommend using **MEMORY_TARGET** instead in 11g*
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 remote 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 Oracle 9i and earlier)
SESSIONS	Maximum number of sessions that can connect to the database
SGA_TARGET	Maximum **System Global Area** or **SGA** size, from which Oracle allocates internal components. *Recommend using **MEMORY_TARGET** instead in 11g*
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**
USER_DUMP_DEST	Directory for Oracle to write trace files (if not over-ridden by setting **DIAGNOSTIC_DEST**)

These parameters can be set in one of either of these two kinds of initialization parm files:

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

The advantage of the SPFILE over the PFILE is that you can dynamically change initialization parms (you can change them without recylcing the database). Here are their differences:

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 taking 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 modern way of setting instance parms that supports many new features

When you start up an Oracle database, Oracle determines which **PFILE** or **SPFILE** file to use by this search order. *Oracle uses with the first one it finds*:

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%\dbs** or **%ORACLE_HOME%\database** on Windows systems. You can 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, but only 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.

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 Parm	Use
SCOPE=MEMORY	Changes ONLY apply as long as the instance is still up
SCOPE=SPFILE	Changes take effect ONLY after the instance is shut down and restarted. *Required for **static parameters.** Cannot be specified if instance was started with a PFILE*
SCOPE=BOTH (default)	Changes apply immediately plus will also apply to all future use of the instance

5.0 Configuring the Oracle Network Environment

Oracle Net is Oracle's communication protocol. In terms of the **Open Systems Interconnect** or **OSI** communications model, it supplies levels 5, 6, and 7 functionality. It runs on top of any layer 4 interface, such as the internet's TCP. Oracle 11g limits support to a smaller set of underlying communications protocols than previously:

- TCP
- TCP with secure sockets
- Windows Named Pipes or NMP
- Sockets Direct Protocol or SDP

Oracle Net's **Two Task Common** or **TTC** layer handles presentation layer functions and data conversions between the user and server processes.

A **listener** is an Oracle background process that listens on a specified port for incoming database connections. Stopping a listener means that no new server processes will be launched, but existing sessions can continue just fine. The listener and the instance must run on the same computer unless you're using RAC—in this case any listener in the cluster can connect users to any instance on any computer in the cluster. There are two methods of **listener registration**:

- **Static**: You hard-code a list of instance names in the **listener.ora** file
- **Dynamic**: Upon start-up, instances locate listeners and **register** with them

Dynamic registration is preferred (especially in clustered environments) and is the newer approach. The PMON background process makes dynamic registration happen. You could force registration by issuing this command:

```
ALTER SYSTEM REGISTER;
```

If the listener is not in its default location of $**ORACLE_HOME/network/admin**, use the **TNSADMIN** environmental variable to point to its location.

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.

You can configure oracle networking and listeners through several tools: OEM Database Control, Net Manager, and the Net Configuration Assistant. The latter two are used on client machines.

The key files to configure for Oracle networking are:

File Name	**Use**
listener.ora	Configures listener(s).
tnsnames.ora	Optional. Used for the name resolution method called **Local Naming.** Translates the local service name into a full network address consisting of the: *protocol*, *address*, *port*, and *service* or *instance name*.
sqlnet.ora	Optional. Can choose the names resolution method, also sets parameters.

For a client to connect to the database server there needs to be some form of name resolution. **Name resolution** translates the information users provide for a connection into the detailed information required by Oracle Net and the underlying communications protocol. Oracle 11g supports:

Resolution	Protocols	Note
Easy Connect	TCP	No configuration needed. Does not support advanced features like *load balancing* or *connect-time failover*. The format is: SQL> connect userid/pass@host_name:port_number/instance Example: SQL> connect scott/tiger@my_box:1522/ocp11g.
Local Naming	All	Requires **tnsnames.ora** file (with *service aliases*). Supports all advanced features but it is time-intensive to keep tnsnames.ora files updated in many places at large sites.
Directory Naming	All	Uses a centralized **LDAP** (**Lightweight Directory Access Protocol**) server. Oracle offers **Oracle Internet Directory** or **OID** for this purpose.
External Naming	All	Like Directory Naming but uses 3rd party products like Sun's Network Information Services or others.

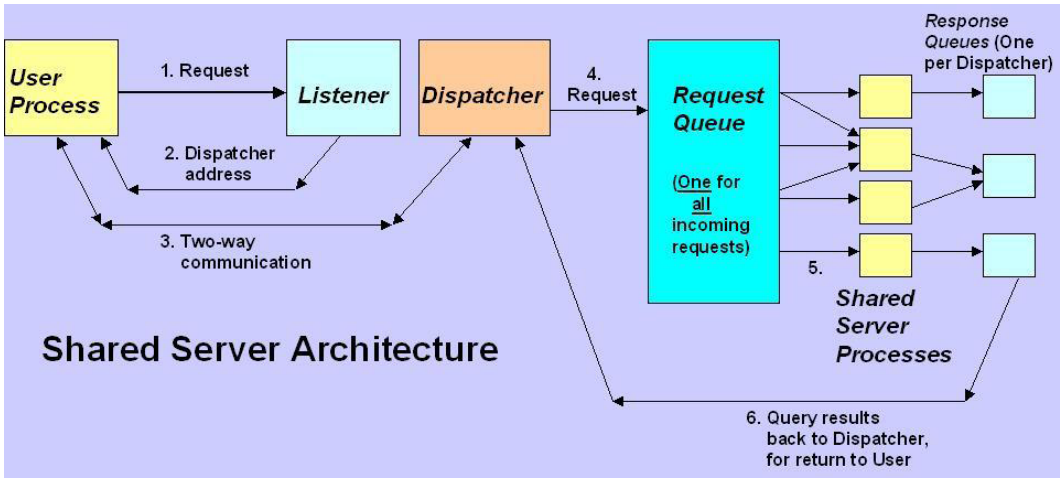
As well as client-to-database communications, Oracle Net also supports database-to-database communications through **database links**. They can be **private** (in your own schema) or **public** (for use by everyone allowed). Here's the command template:

```
CREATE  DATABASE LINK  link_name
        CONNECT TO user_id  IDENTIFIED BY password  USING 'connect_string';
```

5.1 Dedicated versus Shared Server Architectures

In Oracle's **Dedicated Server architecture**, one server process is allocated per one user process. In the **Shared Server architecture**, one server process can service many user processes. This offers scalability because it *conserves memory* and *reduces context switching* between processes.

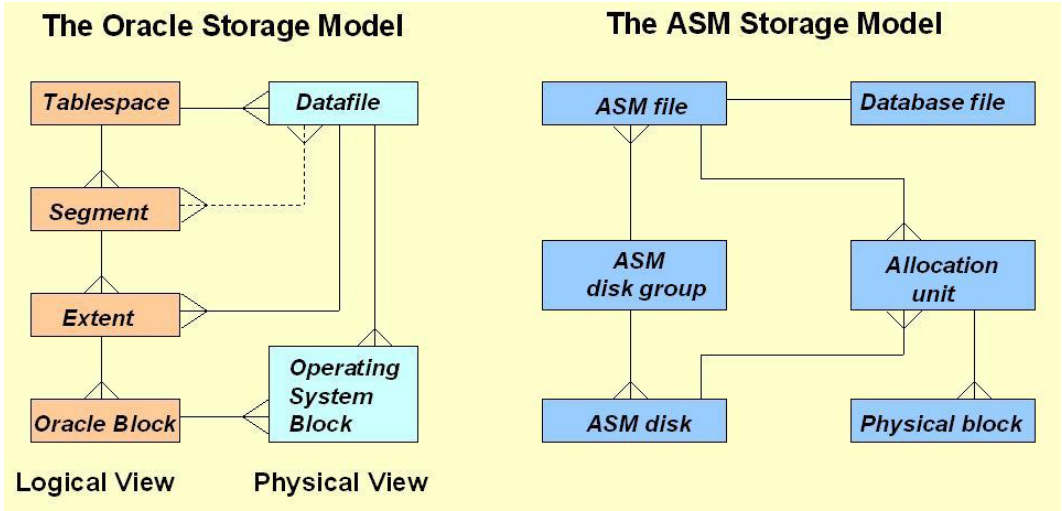
This diagram shows how **Shared Server architecture** works. A **listener** pairs each initial user process request to a **dispatcher** (each of which may handle multiple user processes). All dispatchers place SQL requests into a single shared **Request Queue**. **Shared Server processes** take incoming request from that queue. They place results into **Response Queues**, each of which corresponds to one dispatcher. Dispatchers take results from their response queues and communicate them to the **user processes**:



Shared server architecuture stores *session data* in a part of the **Systems Global Area** called the **User Global Area**, instead of in individual **Program Global Areas**. This is what saves memory. The reduced-size PGA still stores individual users' *stack space*.

6.0 Managing Database Storage Structures

The diagram below illustrates Oracle's essential storage relationships. **Tablespaces** consist of one or more **datafiles**. *Each datafile belongs to one and only one tablespace.* A **segment** is a storage structure for one kind of data. It consists of one or more **extents**, each of which is an *allocation unit of contiguous blocks* on disk. Oracle **blocks** or pages may consist of one or many **operating system blocks**, so they are some multiple of an OS block size.



The common **segment types** are: Table, Index, Table Partition, Index, Index Partition, Undo, Rollback, LOB Segment, LOB Index, LOB Partition, Cluster, and Nested Table. **Partitions** refer to tables or indexes that are split across more than one tablespace, usually because they are big and require partitioning for effective management. **Undo segments** hold "*before change*" images of data so that data can be rolled back to its pre-change state. (**Rollback segments** used to perform this work but are obsolete in 11g.) **LOB's** refer to Large Binary Objects.

6.1 Automated Storage Management or ASM

ASM comes with Oracle. It is a **logical volume manager**, software that decouples physical storage from its logical view and helps manage storage. ASM runs as a separate Oracle instance, called the **ASM Instance**. ASM stores *datafiles*, *control files*, online and archived *redo logs*, *backup files*, and *Datapump files*. But it cannot store the Oracle binaries (ORACLE_HOME), or the alert or trace files.

See the diagram above for ASM components. One or more disks are assigned to an ASM **disk group**. ASM then automatically allocates that space to files and manages them itself. There is a one-to-one relationship between an ASM file and a datafile. ASM automatically and transparently stripes files (not volumes). While mirroring within ASM is optional striping is not.

ASM's allocation unit default size is 1M, but this can be increased up to 64M for appropriate applications (like data warehousing). ASM will **automatically rebalance** data across disks as they are added to or dropped from disk groups.

6.2 Oracle Managed Files or OMF

Oracle Managed Files or **OMF** is a separate feature from ASM. Nevertheless, OMF is often used with ASM because it makes the creation of tablespaces and their underlying datafiles very simple. To use OMF, specify these configuration parameters:

- DB_CREATE_FILE_DEST - Default location for all datafiles
- DB_CREATE_ONLINE_LOG_DEST_n - Default location for online redo log files
- DB_RECOVERY_FILE_DEST - Location for archived redo & backup files

In addition to file locations OMF generates file names and sizes. By default OMF datafiles are automatically named, 100 megabytes in size, and set to autoextend in size as needed. You can always override OMF defaults, including the datafile name, by specifying it on the **CREATE TABLESPACE** command.

OMF provides automatic disk space management for Oracle 11g. Together ASM and OMF make the DBA's traditional job of file management much easier. *ASM is as fast as raw devices yet it automates the physical data placement chores that used to consume much DBA time and effort.* It is nearly always used for clustered systems (RAC) where more than one instance connects to a single database, and its top performance makes it popular for simple single-instance databases as well.

6.3 Tablespaces

Tablespaces are the logical data containers that ultimately map onto one or more **datafiles**. They can store any one of three kinds of objects: *permanent* data, *temporary* data, or *undo* segments. Create tablespaces using OEM Database or Grid Control or from the command line (as in SQL*Plus). Tablespaces can hold regular data (**permanent tablespaces**) or be used for operations like sorting (**temporary tablespaces**). Here is an example traditional CREATE TABLESPACE statement:

```
CREATE TABLESPACE my_tablespace
        DATAFILE 'e:\app\oracle\oradata\orcl11g\my_tablespace.dbf'
        SIZE 100M    AUTOEXTEND ON NEXT 10M    MAXSIZE 200M
EXTENT MANAGEMENT LOCAL   SEGMENT SPACE MANAGEMENT AUTO ;
```

AUTOEXTEND allows the tablespace to grow up the specified **MAXSIZE**. **EXTENT MANAGEMENT LOCAL** manages extent space through a *bitmap*, similar to how **SEGMENT SPACE MANAGEMENT AUTO** does on the segment level. Both are way more efficient than the older data dictionary management option and both are now defaults in Oracle 11g. Tablespaces are of type **SMALLFILE** by default, which allows multiple datafiles for one tablepace. Alternative **BIGFILE** only allows one datafile for the tablespace but it can grow quite large.

You can rename tablespaces, take them offline (or back online), make them read-only, resize their underlying datafiles, and drop them:

```
ALTER TABLESPACE my_ts  RENAME TO ts_new_name ;
ALTER TABLESPACE my_ts  OFFLINE  [NORMAL | IMMEDIATE | TEMPORARY];
ALTER TABLESPACE my_ts  [READ ONLY | READ WRITE];
ALTER DATABASE  DATAFILE df_name  RESIZE n[M | G | T] ;
DROP TABLESPACE my_ts  [INCLUDING CONTENTS  [AND DATAFILES] ] ;
```

The options are altering a tablespace offline are:

1. NORMAL - Updates data first (dirty buffers are written by before offlining).
2. IMMEDIATE - Offline immediately without data update. *Requires recovery later*.
3. TEMPORARY - Checkpoints all datafiles it can. *May or may not* require recovery later.

Making a tablespace read-only means that DML statements can no longer update the tablespace's data and will fail if they try. You can still drop the objects in the tablespace, though. Oracle's backups are smart enough not to back up the tablespace in its read-only state.

Resizing a tablespace upwards only succeeds if space is available, while resizing it downwards can only succeed if the space in the file is not already in use by extents allocated to a segment. Interesting, the only way to tell whether you're upsizing or downsizing is by knowing the current tablespace size, since no keyword on the statement indicates this.

Dropping a tablespace fails if there are any objects in it. Therefore use **INCLUDING CONTENTS** to drop any included objects along with the tablespace. **AND DATAFILES** additionally deletes the underlying datafiles if possible (Windows requires stopping the Service **OracleServiceSID** first due to the way it handles locking).

Oracle automatically monitors various aspects of any database. Among them, the MMON background process alerts you when tablespaces are running out of space. By default it warns when a tablespace is 85% full and issues a critical alert when one is 97% full. You can view all **alerts** from the OEM Database Control Alerts panel.

7.0 Administering User Security

To log into Oracle users must have **user ids** or **user accounts.** User accounts have these attributes, all of which have defaults except the first two:

Attribute	Use
Username	Required. Must be unique in the database, and conform to standard object naming conventions (Begins with a letter, up to 30 characters long, contains only letters, digits, dollar sign ($), the underscore (_) and pound sign (#).
Authentication	Required. *All database users must be authenticated during login.* Here are the ways to determine whether the user can login: 1. *Operating System & Password File* – Instance authenticates users by seeing if they're a member of the OS group that owns the Oracle software. These user account passwords also reside in a special **password file**, which is a regular operating system file outside of Oracle so it can be used by the instance when the Oracle database is down. 2. *Password* – Oracle handles authentication through its Data Dictionary. 3. *External* – An external service or product handles authentication, such as Oracle's **Advanced Security Option** with *Kerberos* or *Radius* servers, or *Windows' native authentication service*. 4. *Global* – A standards-conformant LDAP server like *Oracle Internet Directory* handles authentication. A centralized, system-wide method.
Default Tablespace	Where user's objects will reside by default. A system-wide default tablespace should be defined in the **CREATE DATABASE** statement. *Otherwise it defaults to SYSTEM (which is considered bad practice).*
Tablespace quotas	A **quota** is the amount of space user is allowed in any tablespace he may use. A user must have a quota for a tablespace before he can create any objects in it.
Temporary Tablespace	A shared temporary space area providing space beyond the user's session storage area (which is called the *Program Global Area* or *PGA*). Usually temporary tablespaces are shared by all users on the system. Users do **not** need quotas to use them. *Unless set otherwise the default system-wide temporary tablespace is SYSTEM (which is considered bad practice).*
User Profile	Controls password settings and resource usage.
Account Status	9 possible statuses: **open**, **locked**, **expired**, **expired & locked**, **expired (grace), locked (timed)**, **expired and locked (timed)**, **expired (grace) & locked**, **expired (grace) & locked (timed)**. **Grace** refers to a *grace period* wherein users can change their account to conform to requirements. **Timed** refers how long the state is applied to the user account.

Note from the above table that a user's *default tablespace* and *temporary tablespace* both default to the **SYSTEM** tablespace if not set otherwise. Using SYSTEM for either is a bad practice and you'll therefore want to ensure all users have assigned default and temporary tablespaces. The **CREATE DATABASE** command provides clauses by which you can assign a default tablespace and a default temporary tablespace for all users of the database and thereby easily avoid incorrect use of SYSTEM.

Here is an example of creating a user account:

```
CREATE USER user_id  IDENTIFIED BY password
DEFAULT TABLESPACE ts_name  TEMPORARY TABLESPACE ts_temp
QUOTA 500M ON ts_name,  QUOTA UNLIMITED ON another_ts
PROFILE  profile_name
PASSWORD  expire
ACCOUNT  UNLOCK ;
```

A user account cannot do anything–even connect to the database–without being given rights to perform actions called **privileges**. Privileges are given through the **GRANT** statement and withdrawn through the **REVOKE** statement. The two classifications of privileges are:

Privilege Category	Use
Object	These give the ability to run DML against tables and related objects, and to execute PL/SQL objects.
System	The 200+ system privileges control actions that affect data dictionary definitions. They affect sessions, the database or instance, and the ability to create tables or users. *The ANY privileges that grant permissions against objects in every user account in the database are all system privileges.*

The **WITH GRANT** option allows a user granted an object privilege to grant it to others. The **WITH ADMIN** option allows a user granted a system privilege to grant it to others.
A REVOKE of an object privilege automatically cascades to everyone granted that privilege through that command change. A REVOKE of a system privilege does not cascade.

Example: John GRANTs Raj CREATE TABLESPACE. Raj GRANTs it to Mary Ellen. When John REVOKE's this system privilege from Raj, Mary Ellen retains the privilege because revocation of system privileges does not cascade.

Example #2: John GRANT's Raj EXECUTE on a PL/SQL program he wrote, plus full DML rights on one of his schema views. Raj GRANT's those two privileges to Mary Ellen. When John REVOKE's those two privileges from Raj, Mary Ellen looses them as well. Revocation of object privileges cascade through to all users to whom those privilege(s) have been granted.

7.1 Roles

Roles are groups of system and/or object privileges that can be **GRANT**'d and/or **REVOKE**'d as a unit. This saves the huge workload of individually granting and revoking privileges to individual user ids. Instead, you create a role, and then assign the role to a list of user accounts. The SQL GRANT and REVOKE statements use the same syntax when applied to a role as they do when issued against a specific user id. Roles are not schema objects.

By default any role granted to a user is enabled for that user. You can change this by:

```
ALTER  USER user_id   DEFAULT  ROLE  NONE ;
```

There are 50 predefined roles in an Oracle database (the exact number depends on the install options you select). The important ones to know are:

Predefined Role	**Use**
PUBLIC	Always granted to every database user account. To give *everyone* a privilege, GRANT it to **PUBLIC**. PUBLIC is unique in that it does not show up in the D/D view DBA_ROLES.
CONNECT	Replaced by **CREATE SESSION** in current Oracle releases.
RESOURCE	Also obsolete. Creates data and procedural objects (like tables and PL/SQL procedures). Includes UNLIMITED TABLESPACE.
DBA	Used so that DBA's can manage Oracle, includes most system privileges, except for STARTUP and SHUTDOWN.
SELECT_CATALOG_ ROLE	Allows for viewing the data dictionary but does not also allow for viewing of data.
SCHEDULER_ADMIN	System privileges needed for managing Oracle's Scheduler job service.

7.2 Profiles

Profiles *enforce password restrictions* and optionally *impose resource usage limits*. A profile can be assigned in the **CREATE USER** statement as in the example above, or through the **ALTER USER** statement. Profiles only impose resource usage limits if the configuration parameter **RESOURCE_LIMIT** is set to **TRUE**. The default for this parameter is FALSE so by default profiles do not restrict resource use. (A better means to manage resources and impose limits is to use Oracle's more sophisticated **Resource Manager**.) Password limits are *always* enforced regardless of this setting.

The profile password limits are:

Profile Keyword	**Password Impact** *(time periods are all in Days)*
FAILED_LOGIN_ATTEMPTS	Number of invalid tries allowed before an account locks
PASSWORD_LOCK_TIME	Number of days to lock a locked account
PASSWORD_LIFE_TIME	Number of days before a password expires
PASSWORD_GRACE_TIME	Number of days "grace time" given an expired account before it is locked
PASSWORD_REUSE_TIME	Number of days before a password can be reused
PASSWORD_REUSE_MAX	Number of times a password can be reused
PASSWORD_VERIFY_FUNCTION	Function to run that verifies password complexity

The profile resource limits are:

Profile Keyword:	Resource Limit Impact:
SESSIONS_PER_USER	Number of concurrent logins for one user account
CPU_PER_SESSION	CPU time maximum for a session's server process before ending
CPU_PER_CALL	CPU time for one SQL statement for a sesssion's server process
LOGICAL_READS_PER_SESSION	Number of blocks a session can read before termination
LOGICAL_READS_PER_CALL	Number of blocks one SQL statement can read
PRIVATE_SGA	Kilobytes for SGA session data (w/ Shared Server)
CONNECT_TIME	Session duration maximum in minutes
IDLE_TIME	Maximum session idle time in minutes
COMPOSITE_LIMIT	Weighted sum maximum of several of the above limits

8.0 Managing the Schema Objects

The exam objectives for this section are:

- Create and Modify tables
- Manage Constraints
- Create indexes
- Create and use temporary tables

With the exception of "Create and use temporary tables," these are duplicate exam objectives for the other OCA exam, *Oracle SQL Fundamentals I 1Z0-051*, on table and index DDL and constraints. So as to avoid duplication please refer to chapters 11 and 12 of the Exam Manual for that exam. Review how to create and alter tables and views, the table constraints and how to use them, and the chart listing Oracle's data types.

Here we'll add this quick review of key points:

- **Object names** including tables should begin with a letter, be up to 30 characters in length, and consist only of letter, digits, the dollar sign ($), underscore (_), and pound sign (#)
- Object names cannot be the same as reserved keywords
- Use double quotes to surround non-standard object names
- **Typecasting functions** can be used to explicitly identify data types

- **UNIQUE** constraints allow a column to contain multiple nulls
- **PRIMARY** constraints enforce key uniqueness and do not allow null keys at all. They are equivalent to UNIQUE plus NOT NULL
- PRIMARY and UNIQUE cause automatic index creation if their indexes do not exist
- A **foreign key constraint** is defined on the child table and must point to a valid primary or unique constraint on a parent table
- Constraints can be **enabled** or **disabled** (turned on and enforced for new rows or not)
- Constraints can be **validated** or not (already enforced for existing table rows or not)
- By default constraints are *enabled* and *validated*
- Constraint violations roll back the entire SQL statement that violates them, but not affect other SQL statements in the transaction

8.1 Namespaces

Namespaces define a group of objects, within which the name of each must be unique. Tables, views, sequences, private synonyms, procedures, and functions are all in the same namespace and so must be unique within a user's schema. This means, for example, that you cannot name a table and a view the same, nor a synonym and a procedure, in the same schema.

Indexes, constraints, clusters, triggers have their own namespaces and could therefore duplicate those names. So you could name an index the same as its underlying table in your schema, for example.

8.2 Indexes

Indexes *enforce constraints* and *aid data retrieval performance.* They help Oracle more quickly locate and access data in the database and can sometimes avoid expensive sort operations. But indexes slow down SQL **INSERT'**s since these must now also update the index(es).

B-tree indexes are best for data with a wide range of values–**high-cardinality data–**while **bitmapped indexes** are best for low-cardinality data. B*tree indexes can be unique or non-unique, and can include more than a single column (called **composite** or **compound indexes**). They can be ascending or descending on key, compressed or function-based, or even reverse-key. A **reverse-key index** is exactly what it sounds like. A key of **12345** is automatically reversed by Oracle to be the key of **54321**. Bitmap indexes can be composite, function-based, or ascending or descending. They cannot be unique, compressed or reverse-key.

8.1 Temporary Tables

Temporary table data is only viewable by the session that inserts it (though anyone can see its data dictionary definition). This transient data automatically disappears either at session's end or transaction's end.

Temporary tables are fast-performing because they are not segments in permanent tablespaces. Instead they exist in the PGA of the session that creates them–in session memory–and spill over into disk space if necessary in the user's assigned temporary tablespace. DML against temporary tables does not generate redo, which also increases their performance.

Temporary tables are good for fast manipulation of data when that data does not need to be kept beyond the end of the session—for example, for manipulating data for data warehouse reports or for sorting. Here is how to create a temporary table:

```
CREATE GLOBAL TEMPORARY TABLE table_name
        (column_name_1 dataytype [,column_name_2 datatype…] )
        [ON COMMIT { DELETE | PRESERVE } ROWS ] ;
```

The **ON COMMIT** clause tells how long the temporary data should kept**:**

- DELETE- Deletes data at end of the transaction
- PRESERVE- Deletes data at the end of the user's session

9.0 Managing Data and Concurrency

This exam objective covers some of the same material as chapter 10 "Updating Data Through DML" in the exam *Oracle 11g SQL Fundamentals I 1Z0-051*. Review the ACID properties of transactions and how commit processing works. The PrepLogic Exam Manual for test 1Z0-051 covers this so due to space limitations we won't repeat it all here other than this quick summary. Remember that the purpose of the ACID principle is so that the Oracle database can guarantee **data integrity** or the accuracy of the data:

ACID Principle:	**Meaning:**
A is for **Atomicity**	**Atomicity** means that all parts of a transaction must complete, or else none of them complete
C is for **Consistency**	**Consistency** means that the results of a query must be consistent with the state of the database at the time the query started
I is for **Isolation**	**Isolation** means that an incomplete transaction must not be visible to any user or program except the one currently working on it
D is for **Durable**	**Durability** means that once a valid transaction completes (is committed), the database must never lose it

For this exam concentrate on the internals of how data retrieval and updates operate. The **database buffer cache** holds datafile **blocks** in memory for quicker access than disk would provide. **Server processes** (*dedicated* or *shared*) access datafile blocks from the database buffer cache or read them from the datafiles on disk into the cache if they're not already there. The background processes Database Writer or **DBWn** write *dirty* or updated database buffer cache blocks to disk at appropriate times. *A COMMIT does not force the database writers to write anything to disk, but it does force the log writer LGWR to write the log buffer to disk.* A database **checkpoint** causes DBWn to write all dirty blocks to disk. **Full checkpoints**–database-wide checkpoints–only occur upon request or the orderly shutdown of the database.

Undo data logically consists of *before images* of data that will be changed. (Undo was once called *Rollback data* in earlier Oracle versions.) **Redo** is the logging of updates. Redo protects all block changes, so it applies to table, index and even undo block changes. Thus a rollback operation itself generates redo as it executes. *Redo includes data for both committed and uncommitted transactions.* Redo protects everything Oracle does.

INSERT's generate minimal undo data—all that need be recorded is the new rowid to the undo block. **DELETE**'s generate lots of undo data, because the whole deleted row must be written to undo so that the deletion can be reversed or undone (*rolled back*) if needed. The amount of redo generated by **UPDATE**'s vary by how many columns are updated and their internal sizes.

9.1 Administering PL/SQL Objects

SQL is a powerful set-oriented language. But it lacks procedural structures and facilities for building user interfaces. **PL/SQL** is Oracle's proprietary 3rd generation language that adds these features. The three programming languages that can run within the Oracle database engine are *SQL*, *PL/SQL*, and *Java*.

Stored PL/SQL or **named PL/SQL** is stored as a named object in the Data Dictionary. It is stored on the database server and compiled for fast execution performance. **Anonymous PL/SQL** is run on a dynamic or ad hoc basis. It can be stored either on a client or server. Since it must be compiled before it is run anonymous PL/SQL does not perform as well as stored or named PL/SQL.
PL/SQL objects include:

- **Procedure**: A code block that has optional input arguments
- **Function**: A code block that returns a single value and is not directly EXECUTE'd
- **Package**: A group of related procedures and functions. Every package has a:
  - Specification: Tells input arguments and their data types
  - Body: The PL/SQL code that implements the package functionality
- **Trigger**: A special PL/SQL object that **only** runs either immediately before or after a specified event occurs

Database triggers are PL/SQL programs that can only be run immediately before or after an event occurs. They run or *fire* automatically due to the event occurrence. They cannot be run independently of the event to which they are coupled. The events they can be defined for include DML statements (INSERT, UPDATE, DELETE), DDL statements (CREATE, ALTER, DROP, TRUNCATE), database operations (SERVERERROR, LOGON, LOGOFF, STARTUP, SHUTDOWN), and SUSPEND (when a resumable operation is temporarily suspended). You have to write the code in triggers yourself, but in return you get great flexibility.

Triggers are useful to:

- Audit user actions
- Enforce data edits
- Enforce data constraints
- Enforce security

9.2 Locking

Locking is a means of ensuring **data integrity** or data accuracy. It is critical because more than one user or program may try to update the same data at the same time. Locking is Oracle's way of policing concurrent updates. Oracle's locking is efficient and largely transparent. It guarantees the highest possible level of **concurrency** (simultaneous use of the data by multiple users).

There are two kinds of locks. **Exclusive** locks are required for update operations, while **shared** locks are for reading data. Shared locks prevent someone else from taking an exclusive lock on the data you're working with, so it doesn't change while you're in the middle of reading it, for example.

Locks are serviced on a first-come, first-served basis. Where locks conflict, an **enqueue mechanism** makes those later in the incoming stream wait. **Lock contention** means there are processes waiting on locks in the queue.

Normal **SELECT** statements require no locks so they always succeed. Update DML (**INSERT**, **UPDATE**, **DELETE**) may have to wait in a contentious situation. A DML statement acquires shared locks on the objects it targets and exclusive locks on the rows involved. The **SELECT… FOR UPDATE** tries to take exclusive locks so it is the same as update DML. If you don't want your SELECT … FOR UPDATE to wait for objects to become available for its locks, add optional parameter **NOWAIT**. Or add **WAIT <n>**, where **n** is the maximum number of seconds to wait for a lock before failure.

A DDL update always requires an exclusive lock on the object its affects.
You can identify lock contention in the V$ views, but an easier way is the graphical interface of OEM Database Control. The **Instance Locks** panel identifies locking processes and sessions, the object(s) they lock, the kind of lock (exclusive or shared), and other key locking details.

To reduce locking contention, write programs that:

- Only lock data when necessary (use non-locking SELECTs whenever possible)
- Lock only the data they require (don't lock tables you don't update with SELECT … FOR UPDATE or other update DML)
- Lock the data as briefly as possible
    - Decrease duration of transactions by frequent COMMITs
    - Keep non-essential processing out of your transactions
- Programs updating multiple tables should all update those tables in the same processing order. For example, if one program updates tables A, B, and C in that order, so should any other update program your team writes.

Deadlocks occur when two processes each have locked a resource the other needs in order to proceed, and neither will give up the lock it already has.

For example, say Program 1 has locked Table A for update, and it needs to keep its lock on Table A while acquiring an update lock on Table B. Meanwhile, Program 2 has locked Table B for update, and it needs to acquire an update lock on Table A to complete. You can see that these two processes are **deadlocked**. Neither can complete because it is blocked by the other's lock on a table it needs to complete. If Oracle didn't have a means to deal with this, these two programs would wait forever on each other's lock.

Oracle automatically detects and resolves all deadlocks—you do nothing about them yourself once they occur. Oracle generally rolls back (causes to fail) the process that has done the least amount of work. This is the most efficient means to resolve deadlocks.

10.0 Managing Undo Data

Undo data is a pre-update version of any data that is in the process of being changed. Think of it as "before change" images of data. Oracle uses undo for:

- A **read-consistent image** of the database for all queries
- A means to guarantee data integrity and uphold the ACID principles
- A way to *roll back* data to its pre-change state if a transaction fails
- For **flashback queries** (queries that give a view of data at a prior point in time)

Undo data used to be kept in *rollback segments*, which were manually managed by DBAs. Today the **undo tablespace** and Oracle's **automatic undo management** automate all this and replace the old rollback segments.

Enable automatic undo management by setting configuration parameter **UNDO_MANAGEMENT=AUTO** (which is the default in 11g). You also need to identify **one active undo tablespace** by the configuration parameter **UNDO_TABLESPACE**. *You can have many undo tablespaces in the database but only one can be* ***active*** *(in use) at any one time.*
Undo tablespaces are like other tablespaces—datafiles can be added, they can be resized, moved, taken on- and off- line, and renamed. Use the keyword **UNDO** to identify them:

```
CREATE  UNDO  TABLESPACE  tablespace_name
        DATAFILE datafile_name  SIZE initial_size
        [ RETENTION  {NOGUARANTEE  |  GUARANTEE} ] ;
```

If you create the undo tablespace with Database Configuration Assistant or DBCA, it will set autoextend on and the maximum size to unlimited. If you issue the **CREATE UNDO TABLESPACE** statement above, use the **DATAFILE** clause to set autoextend on. Oracle recommends autoextend on. You can toggle this value on or off at will.

You can optionally set oracle configuration parameter **UNDO_RETENTION** to a value in seconds. This is a target for keeping inactive undo data and it determines when data becomes expired. **Guaranteed undo retention** is specified for an undo tablespace when you create it as above or enable it for the tablespace later. UNDO_RETENTION should be set if you enable guaranteed undo retention for the undo tablespace.

In RAC environments, more than one instance works against a single database. *Each RAC instance* ***must*** *have its own undo tablespace.* Configuration parameter UNDO_TABLESPACE will be set to a different tablespace for each instance.

10.1 How Undo Works

Oracle automatically creates undo segments in the undo tablespace as transactions come into the system. Each transaction is assigned to one undo segment when it starts. Segments grow as needed (Oracle adds extents to the undo segments as needed). *Transactions only ever use one undo segment, but one undo segment can support multiple transactions*. Undo data is classified as:

- **Active** – It might be needed to roll back a transaction in progress
- **Unexpired** – The transaction has been committed but the undo still might be needed
- **Expired** – The transaction has been committed and the undo is no longer needed

As Oracle requires more undo space, it will extend the undo tablespace if possible. It will also reclaim space by overwriting expired undo. Unexpired undo will only be overwritten if there is a shortage of undo space. Of course, active undo can never be overwritten.

If a DML statement runs out of undo space, its successful portion will be rolled back. The remainder of the transaction remains intact and uncommitted. The error message will be "*ORA-30036 Unable to extend segment in undo tablespace.*"

The other error you might encounter is "*ORA-1555 Snapshot too old*." This means Oracle was unable to find the preupdate version of the data needed to guarantee read consistency for a query. Oracle's default mode of operation favors transactions over queries. You may be able to eliminate this error by making the undo tablespace larger. You could also eliminate this error by turning on guaranteed undo retention for the undo tablespace so that transactions finish within the undo retention time. (The price is that some transactions could fail.)

For both errors, you should ensure that autoextend is on for the undo tablespace. Increase the size of the undo tablespace. Add more datafiles to it if necessary or even move it to different larger disks for more space. The point to having multiple undo tablespaces – even though only one is in use or active at any one time – is to provide flexbility in resizing the undo as required.
Monitor and correctly size the undo tablespace through OEM Database Control. The **Automatic Undo Management** panel has an **Undo Advisor** that helps you size the undo. Or use the view **V$UNDOSTAT** for viewing undo statistics directly.

10.2 Flashback Query

Flashback query is an Oracle feature that allows you to perform a query against the database as of some earlier time. *For any such query to work you must have the undo data still available as of how far back the query goes.*

Issue a SELECT statement with the **AS OF** clause to use flashback query:

```
SELECT  *  FROM my_table  AS OF  some_timestamp ;
```

This example displays my data as of one hour ago:

```
SELECT * FROM employees  AS OF  timestamp (systimestamp – 1/24) ;
```

Here's how to reclaim data into a table as of one hour ago, if you accidentally deleted it:

```
INSERT INTO employees
        (SELECT * FROM employees AS OF timestamp (systimestamp – 1/24)
        MINUS  SELECT * FROM employees)  ;
```

This is very useful as a way to recover data if from a *logical user error* like accidentally deleting data from a table. It is a fast, easy recovery method *on the table level*. For it to work you must have have **UNDO_MANAGEMENT=AUTO** and set **UNDO_RETENTION** to a value high enough to save the undo you want to go back to. Autoextend on and a lot of space available to the undo tablespace help, and consider a retention guarantee for the undo tablespace.

Note that flashback query will **not** help if the user dropped the table, because **DROP** is a DDL operation that not only eliminates the data but also removes the table's definition from the Data Dictionary. Use a different Oracle feature called **flashback drop** to bring back a table and its rows if it is dropped by a user in error. Flashback drop works on the basis that since Oracle 10g the **DROP TABLE** command no longer immediately drops a table, instead it just renames it into Oracle's **Recycle Bin**. Here's an example of how to use flashback drop to bring back a table and its data that we accidentally dropped:

```
SQL> drop table employees;

Table dropped.

SQL> select count(*) from employees                    -- Oh no! My boss will kill me! ;
select * from employees
              *
ERROR at line 1:
ORA-00942: table or view does not exist

SQL> flashback table employees to before drop          -- Flashback Drop to the rescue ;

Flashback complete.

SQL> select count(*) from employees;

  COUNT(*)
----------
       107
```

11.0 Implementing Oracle Database Security

The **principle of least privilege** states that everyone should only have access to the minimal resources to perform their work and that anything not specifically permitted is forbidden. This principle underlies Oracle's security system.

Auditing enhances security by tracking and logging user account activity. Oracle supports four basic approaches to security auditing:

1. **SYSDBA Auditing**: Audits every statement issued by anyone with either **SYSDBA** or **SYSOPER** credentials. To enable it, change configuration parameter **AUDIT_SYS_OPERATIONS** from its default value of **FALSE** to **TRUE**. Operationally you'll want to ensure **separation of duties** – that the people using SYSDBA and SYSOPER credentials cannot change the output audit records. On Windows, these **SYS** records are viewed by the **Windows Application Log**. In Unix/Linux, they are written to the directory location set by the configuration parameter **AUDIT_FILE_DEST**.

2. **Database Auditing**: Tracks use of some privileges and commands, access to some tables, and some login attempts. Enable it by setting configuration parm **AUDIT_TRAIL** from **NONE** or **FALSE** to the destination for the audit records: **OS**, **DB**, **DB_EXTENDED**, **XML**, or **XML_EXTENDED**. The big decision you have to make is whether the records will go to an operating system flat file or to the database table in the data dictionary called **SYS.AUD$.** You can view the audit records through OEM Database Control or through the **DBA_AUDIT_TRAIL** dictionary view.

   Configure database auditing by the **AUDIT** command. You can generate records **BY SESSION** (one record per session for each incident type) or **BY ACCESS** (once per each incident). Records can also be generated **WHENEVER SUCCESSFUL** or **WHENEVER NOT SUCCESSFUL**. Here are examples:

   ```
   AUDIT  CREATE  ANY  TABLE  ;
   AUDIT  CREATE  ANY  TABLE  BY ACCESS ;
   AUDIT  SELECT ON employees WHENEVER SUCCESSFUL ;
   ```

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

AUDIT_TRAIL	Use
NONE or FALSE	Default setting. *If you use DBCA in 11g and take its defaults DBCA will set this by default to **DB** instead.*
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.
XML	Same destination as **OS** but formatted with XML tags.
XML_EXTENDED	Same as XML but with bind variables and the SQL statement text.

3. **Auditing by Triggers**: Chapter 9 described how database triggers automatically execute either immediately before or after database events. Triggers always fire upon event occurrence and cannot be avoided so they are good for secure auditing. Triggers are programs so they are customizable and can write whatever you want in the audit records. They also can execute upon update DML: INSERT, UPDATE, and DELETE. Tiggers are very flexible but cause overhead.

4. **Fine-Grained Auditing (FGA)**: FGA provides row-level tracking of DML updates to tables. Configure it via the **DBMS_FGA** package and its procedures to enable, disable, configure and drop **FGA policies**. View FGA results through dictionary view **DBA_FGA_AUDIT_TRAIL**. View **DBA_COMMON_AUDIT_TRAIL** shows both FGA audit records and those output by standard Database Auditing (#2 above).

11.1 Other Security Issues

DBCA has enhanced default security settings in 11g. If you accept them these automatically turn on the audit trail, configure stronger settings in the default user profile, and turn revoke certain grants from the **PUBLIC** role.

You should also be aware of these configuration parameters that can affect security:

Config Parm	Default	Meaning
UTL_FILE_DIR	NULL	If set can allow users access to operating system files & directories through Oracle
REMOTE_OS_AUTHENT	FALSE	If set can allow users to remotely connect to the database without a password
O7_DICTIONARY_ACCESSIBILITY	FALSE	If set can allow users to see Data Dictionary tables they may not need to see
REMOTE_LOGIN_ PASSWORDFILE	NONE	If set can allow users to remotely access a database as SYSDBA or SYSOPER

12.0 Database Maintenance

Two classes of statistics are critical to maximizing Oracle's performance:

1. Object Statistics
2. Instance-level statistics

12.1 Object Statistics

Object statistics are collected on tables, columns, views, indexes and other objects and are critical to Oracle's **optimizer** so that it can devise the most efficient SQL **execution plans**. *They are stored in the Data Dictionary* and are viewable in views such as **DBA_TABLES**, **DBA_TAB_COLUMNS**, **DBA_INDEXES**, and **INDEX_STATS**.

Object statistics are not gathered in real time but rather at specific intervals as well as on demand. The trade-off you face is the processing time and overhead for gathering statistics versus the cost of the optimizer creating inefficient execution plans based on outdated statistics.

Run command line procedures from the package **DBMS_STATS** (such as **GATHER_SCHEMA_STATS** and **GATHER_TABLE_STATS**). Or run the less-sophisticated **ANALYZE** command like this:

```
ANALYZE  TABLE employees COMPUTE STATISTICS ;
```

You can compute *full object statistics* (slower but comprehensive) or *estimated statistics* (faster but possibly less accurate). OEM Database Control offers the **Gather Optimizer Statistics Wizard.** OEM DBC ultimately runs DBMS_STATS procedures on your behalf.

Oracle recommends automatic periodic collection of statistics. If you used DBCA to create your database, a job for automatic statistics collection was set up in the Oracle job **Scheduler**. The **Default Maintenance Window** or **DMW** the Scheduler uses for maintenance jobs is:

- Mon thru Fri night 2200 to 0200 (4 hours long)
- Sat and Sun 0600 to 0200 (20 hours long)

Oracle's **Resource Manager** facility automatically ensures that *no more than 25%* of machine resources are dedicated to these maintenance **Autotasks**. You can adjust the maintenance window and all these parms through OEM Database Control.

12.2 Instance Statistics

While object statistics are gathered on specific database objects and stored in the Data Dictionary, **instance statistics** are accumulated in memory and by default written to disk *at one hour intervals* by the **Manageability Monitor** or **MMON** background process. These statistical **snapshots** are stored in the **Automatic Workload Repository** or **AWR**. By default every hourly snapshot is stored *for 8 days* before being overwritten. The AWR exists as a set of tables under the **SYSMAN** schema within the required **SYSAUX** tablespace. You cannot move AWR out of the SYSAUX tablespace.

The configuration parameter **STATISTICS_LEVEL** drives how many statistics are gathered:

STATISTICS_LEVEL	Use
BASIC	Disables AWR statistics collection, automatic daily analysis, and *Alerts* based on the findings. Not recommended.
TYPICAL (default)	Auto-gathers statistics needed for Oracle's self-management and tuning, and runs daily object analysis task during the **Maintenance Window.** Default and recommended**.**
ALL	Collects all possible statistics. Useful for brief periods when actively tuning SQL statements. Too much overhead otherwise.

OEM DBC uses the **SYSMAN** user account and AWR stores its statistics in the SYSMAN schema. To change SYSMAN's stored encrypted password requires **two** steps:

1. ALTER USER SYSMAN IDENTIFIED BY new_password ;
2. emctl setpasswd dbconsole

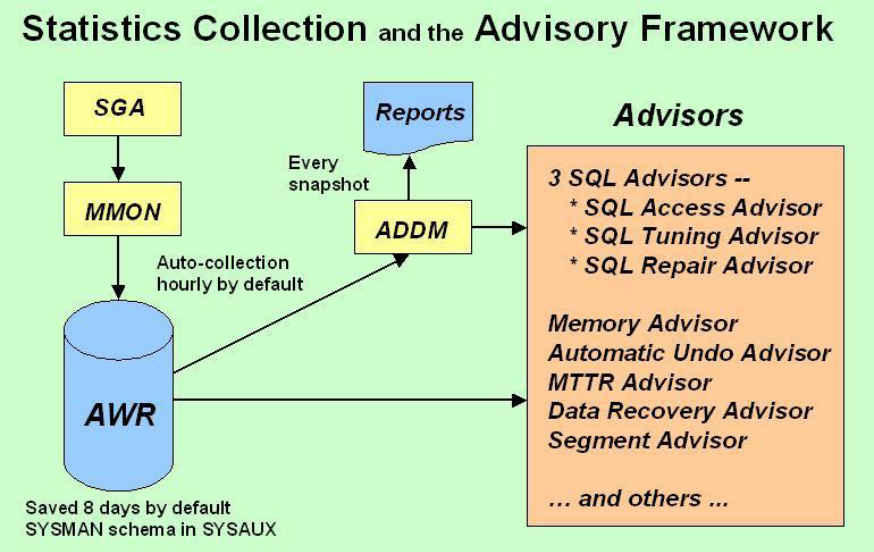
While by default statistics snapshots are automatic every hour, you may optionally create a baseline upon demand. A **baseline** compares a pair of snapshots. It turns raw statistics into useful **metrics** (correlations between raw numbers.) Baselines are across-time comparative views of the statistics. You create baselines (not Oracle's automated system), decide how long to keep them, and manually delete them when you don't need them anymore. OEM Database Control is the easy way to create baselines. Underneath its GUI it runs **DBMS_WORKLOAD_REPOSITORY** procedures like **CREATE_SNAPSHOT**, **MODIFY_SNAPSHOT_SETTINGS**, **CREATE_BASELINE**, etc. Of course you can manually run these procedures from the command line yourself as an alternative to OEM DBC. The data dictionary view **DBA_HIST_SNAPSHOT** contains many baseline results.

12.3 The Advisors

The **Advisory Framework** uses AWR statistics as input and produces a variety of reports and recommendations for database, instance, and SQL tuning. The centerpiece of the Advisory Framework is the **Automatic Database Diagnostic Monitor** or **ADDM**. ADDM automatically produces reports every time a snapshot is taken (hourly by default). You can also direct ADDM to produce additional reports. The reports highlight problems and recommend corrective actions. They often tell you to run one of the other Advisors to key in on a specific problem. By default reports are deleted after 30 days. Key Advisors are:

- SQL Access Advisor - Database design recommendations
- SQL Tuning Advisor - SQL statement tuning recommendations
- SQL Repair Advisor - Recommendations on "safe plans" across Oracle versions
- Memory Advisor - Memory advice and **MEMORY_TARGET** settings
- Undo Advisor - Helps size the Undo tablespace
- MTTR Advisor - **Mean Time To Recovery** advice for crash recovery
- Data Recovery Advisor - Advice on how to best recover data
- Segment Advisor - Recommendations on when and what to reorganize

This diagram summarizes the statistics collection and advisory framework:



Oracle's **AutoTask system** includes three tasks that are scheduled by default:

- Gather optimizer statistics
- Run Segment Advisor
- Run SQL Advisor

These are configured by DBCA by default when creating a database and run thru the Scheduler during the Default Maintenance Window.

Oracle's **Alert system** is based on hundreds of **thresholds**. Some you set, some are set by default. The *MMON background process* raises alerts when threshold are exceeded. View the Alerts through OEM Database Control or the view **DBA_OUTSTANDING_ALERTS** and eventually through **DBA_ALERT_HISTORY**. The package **DBMS_SERVER_ALERT** provides a command line interface to the Alert system. The default notification system is to display alerts, but you could also trigger OS commands, PL/SQL programs, or email. To configure alerts: (1) Select a Notification Method (2) Create a Rule for the event (3) Subscribe to the Rule.

13.0 Performance Management

In Oracle 11g, memory management is fully automated through **Automatic Memory Management** or **AMM**. Simply set two configuration parameters:

- MEMORY_TARGET : Size of memory for Oracle's use (SGA and PGA)
- MEMORY_MAX_TARGET: Maximum memory size that can be dynamically set

MEMORY_TARGET is dynamic. You can adjust it up to the size of **MEMORY_MAX_TARGET**. Since MEMORY_MAX_TARGET is a static parameter you can only re-set it by shutting down and restarting. Therefore set MEMORY_MAX_TARGET to the maximum size you think you might need to increase Oracle's overall memory. With AMM Oracle adjusts available memory as required for best performance between the SGA and PGA, and also between the several pools within the SGA.

Recall that the MMON background process by default collects hourly statistics for the Automatic Workload Repository or AWR. Oracle uses this information for AMM's operations. Through OEM Database Control's **Advisor Central** panel you can access the **Memory Advisors** that will help you optimally size MEMORY_TARGET and MEMORY_MAX_TARGET. Select the **SGA Tab -> Advice** and you'll see graphical output from the view **V$SHARED_POOL_ADVICE**. The **SGA Size Advice** panel estimates percentage improvements for DB Time for various increases to Oracle's memory. Of course, you must have configuration parameter **STATISTICS_LEVEL** set to **TYPICAL** or **ALL** for the Advisors to do their work.

There is one configuration parameter that is static and cannot be dynamically resized (either automatically by Oracle or manually by you). This is the log buffer. The size for **LOG_BUFFER** is fixed at startup and resizing it requires shutdown. Fortunately the default is appropriate for most systems.

Remember that the PGA is the non-shared memory associated with a server process. In the Dedicated Server Architecture it stores session-specific data including:

- Temporary tables
- Sorting rows
- Merging bitmaps
- Variables
- Private SQL areas
- Call stack

In Shared Server architecture, the *call stack space* is the one component that still exists in the private PGA rather than in the SGA.

13.1 Prior Oracle Releases

In Oracle 10g, memory management was less automatic. You set two memory configuration parameters, one for the SGA and one for the PGA. Oracle would automatically manage memory within each of these two parms, but could not move memory between them if needed. The two parms were:

- SGA_TARGET: Memory for the SGA (and all its internal pools)
- PGA_AGGREGATE_TARGET: Memory for PGA

To enable Oracle's ability to assign PGA to sessions on demand, **WORKAREA_SIZE_POLICY** also had to be set to **AUTO—**this was the default.

Oracle would dynamically assign memory from **SGA_TARGET** to its pools including the Shared Pool, the Database Buffer Cache, the Large Pool, the Streams Pool, and the Java Pool. It would assign **PGA_AGGREGATE_TARGET** memory to PGAs as needed, keeping within that overall target size.

Before Oracle 10g, you had to manually manage memory. This involved calculating and sizing the individual pools within the SGA. This required much time and expertise and is no longer covered in the exam.

13.2 Invalid Objects

Both procedural objects and data objects depend on underlying data objects. For example, procedures and views depend on their underlying tables. If the definitions of the underlying tables were to change, this renders the dependent procedures and views not usable. They have to be recompiled to get back to being synchronized with their underlying tables.

Thus there are two classes of objects that can be disrupted by changes to underlying objects:

- Procedural objects
- Data objects

Procedural objects include:

- Procedures
- Functions
- Packages
- Triggers
- Object types

When a **procedural object** no longer matches its underlying object, it becomes **INVALID**. When a user tries to use the procedural object, Oracle will automatically recompile the object in an attempt to dynamically render it valid. This may or may not be successful (depending on the state of the underlying object). If recompilation is successful, the user work continues. If recompilation fails, the user receives an error message.

As an alternative to Oracle's auto-recompile, you could manually recompile an invalid object yourself:

```
ALTER  object_type  object_name  COMPILE ;
```

Example:

```
ALTER  package  my_hr_package  COMPILE ;
```

You could recompile all invalid objects by running script **utlrp** when connected as **SYSDBA**, like this:

- Unix/Linux SQL> @?/rdbms/admin/utlrp
- Windows SQL> @?\rdbms\admin\utlrp

Research the invalid objects by looking at the **STATUS** column in view **DBA_OBJECTS**. This lists all invalid objects:

```
SELECT owner, object_name, object_type FROM dba_objects
    WHERE status = 'INVALID' ;
```

The view **DBA_DEPENDENCIES** gives information on which underlying objects may be causing the difficulty. Column **REFERENCED_NAME** identifies the objects referenced.

13.3 Unusable Objects

While *procedural objects* can become *invalid*, indexes disrupted by changes to the object(s) on which they depend are called **unusable**. Unlike procedural objects, Oracle does not try to automatically fix unusable indexes. You have to manually rebuild them yourself.

Indexes typically become unusable because the rowids in the underlying table change. This occurs, for example, if a table is moved. Recall that indexes pair their keys with the associated rowids to provide quick direct access to table blocks and their rows. You can list unusable indexes by this query:

```
SELECT owner, index_name FROM dba_indexes
    WHERE status='UNUSABLE' ;
```

What happens to a user who runs a SQL statement that tries to use an unusable index? It depends on the setting of configuration parameter **SKIP_UNUSABLE_INDEXES**. If set to TRUE, its default, Oracle will complete the SQL statement by finding an alternative access path. This will likely be slower than if the index were usable, but it gives the user a valid result all the same. The exception would be if the index were required to enforce a constraint. For example if the primary key index becomes unusable, the table becomes locked for DML.

If you set SKIP_UNUSABLE_INDEXES to FALSE, any query that tries to use an unusable index fails with an error message.

Note that while Oracle will try to dynamically recompile **INVALID** objects for users, it does not dynamically rebuild **UNUSABLE** indexes. You must always manually rebuild the indexes yourself.

To rebuild an index, run the **ALTER INDEX… REBUILD** statement:

```
ALTER  INDEX  index_name  REBUILD ;
```

By default, Oracle locks the underlying table for DML processing during the rebuild, rebuilds the index into the same tablespace, and generates redo logs.

Add the **ONLINE** keyword to override DML locking and make the table available during the index rebuild. This requires over two times the index space to work though. Add the **TABLESPACE** keyword if you want to rebuild the index into a new tablespace. Finally you can speed up the rebuild by turning off logging with **NOLOGGING.** This tells Oracle not to generate redo during the index rebuild so you should immediately back up the index tablespace after this process completes.

Taken together, these options give you great flexibility when rebuilding indexes. Rebuild them quickly (at the cost of more space) or in place to save space (at the cost of time). Log to generate redo or speed the process without any logging.

This example statement keeps the index available for use while rebuilding it, and also turns off logging to speed the rebuild. You'll have to have at least double the original index's disk space for this statement since it keeps the index online during the rebuild:

```
ALTER INDEX my_index REBUILD ONLINE NOLOGGING ;
```

14.0 Backup and Recovery Concepts

The five kinds of database errors that can occur are:

1. **Statement Failure:** When a SQL statement fails it is automatically rolled back by the database. Any other DML statements in the transaction remain intact and uncommitted.

2. **User Process Failure:** Background process PMON automatically cleans up after any user session or program that fails. It rolls back any active transaction automatically. Remember that a rollback releases any and all locks the transaction applied.

   There are many possible causes for user process failure. **Network Failure** is example. Minimize it by configuring your network with multiple listeners, network interface cards, and transmission routes.

3. **User Error:** Unfortunately the database considers these not to be errors, since Oracle has done what the user asked for. Technologies to recover from user error include flashback query, flashback drop, flashback database, Log Miner, Datapump, and incompelete recovery. These tools rely on *undo* and *redo* data (as well as database backups and flashback logs, on occasion).

4. **Media Failure:** This occurs when a disk fails. Often data file(s) must be **restored** from backup, then **recovered** by applying logs to bring them up to the current point in time with the rest of the database. **Multiplexing** or mirroring the control file, online redo logs, and archived redo logs minimizes the need to recover them. Oracle does not multiplex filesystem datafiles. Instead use operating system facilities like mirroring, RAID disk arrays, or Oracle's **Automatic Storage Management** or **ASM**.

5. **Instance Failure:** This occurs when the database is not shut down in an orderly manner (for example, as the result of a power failure or other crash). Let's discuss how this works.

14.1 Instance Recovery

Oracle automatically performs instance recovery when it starts if *instance failure* has occurred. How does it know to do this? The **System Change Numbers** or **SCNs** stored in some of the datafile headers will not match those kept in the control files. This proves that the database is not at a point of **database-wide consistency**.

The **checkpoint** operation forces a point of database-wide consistency. Remember that **full checkpoints** only occur:

1. Upon your request (e.g.: ALTER SYSTEM CHECKPOINT;)
2. When the database is shutdown in an orderly fashion (by: SHUTDOWN {NORMAL | IMMEDIATE | TRANSACTIONAL};)

So instance recovery automatically occurs after an abnormal database shutdown (such as from a power failure or from you issuing a **SHUTDOWN ABORT** command).

Here's how a database starts. This shows how instance recovery works:

1. STARTUP command.
2. SMON process reads the control file when the database gets to **MOUNT** mode.
3. If any SCNs in the datafile headers and control files do not match, then instance recovery commences. **Roll Forward** re-applies all change records from the redo logs to ensure the datafiles are caught up to the current SCN. *This means all committed changes are now in the database. But uncommitted changes could still be in the database.*
4. The database is opened for use. Its status goes from **MOUNT** to **OPEN**.
5. While the database is open, instance recovery completes by **Roll Back**, which removes any uncommitted transactions from the database.

Through the instance recovery process Oracle guarantees that the database is never corrupt.

You can control the **Mean Time To Recovery** or **MTTR** of instance recovery. Set the value of configuration parameter **FAST_START_MTTR_TARGET** as follows:

FAST_START_MTTR_TARGET Setting	Use
0 (default)	Maximizes ongoing performance by minimizing DBWn writes to disk. MTTR will be lengthened.
1 to 3600 seconds (maximum value)	This value gives Oracle a target time within which it will try to perform instance recovery. Setting this also turns on **checkpoint auto-tuning** – by which Oracle uses statistics to increase DBWn writes without reducing database performance if possible.

Database Control provides the **MTTR Advisor** to get advice and set this value optimally. Or interrogate the view **V$INSTANCE_RECOVERY.** The column **ESTIMATED_MTTR** tells what Oracle estimates the MTTR to be with your current settings. Column **WRITES_MTTR** tells the price you're paying in additional DBWn writes to meet the MTTR target.

While **full checkpoints** only occur upon orderly database shutdown or at your request, **partial checkpoints** occur automatically upon any number of events. Partial checkpoints synchronize just some of the datafile headers in the database. They occur when taking a tablespace or datafile offline, dropping a segment, truncating a table, or putting a tablespace into backup mode.

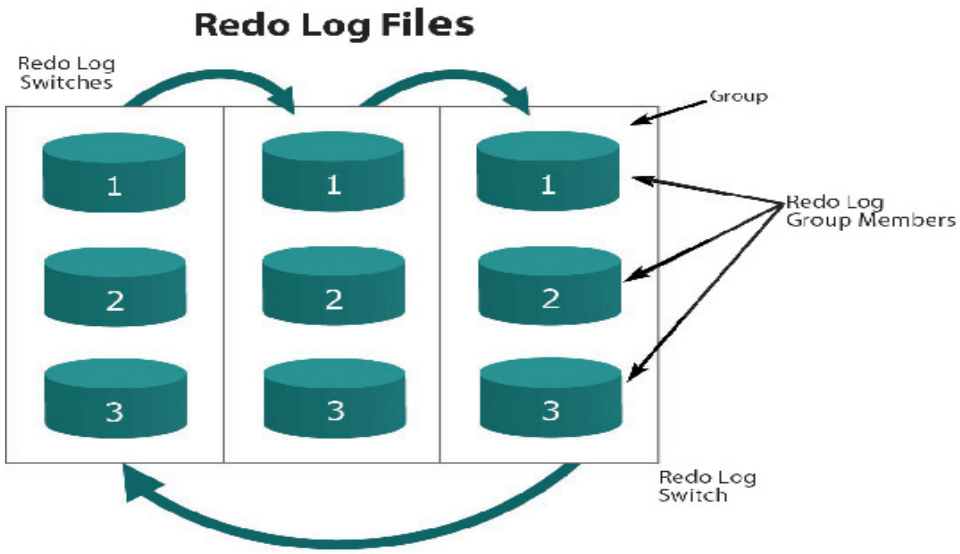
Partial checkpoints mean that some, but not all, *dirty* or changed pages in the database buffer cache are written to datafiles on disk by DBWn background processes.

Online redo **log switches** have **not** caused a checkpoint since Oracle release 8i. Log switches occur as we'll describe in the next section, or upon your manual command:

```
ALTER SYSTEM SWITCH LOGFILE ;
```

14.2 Redo Log Files and Archivelog Mode

Redo logs contain the information necessary to re-apply or re-do database changes. As the diagram below shows, Oracle writes to online redo log files, one after another, in circular fashion. Online redo log files are gathered into **groups** for the purposes of multiplexing and safety. Each **member** of the group mirrors the other(s). *Oracle requires at least two online log file groups and one member per group to function.* But you should definitely multiplex online redo log files, meaning you should have more than one log file member per group.



Now you can see how Oracle guarantees that database changes are never lost. All database changes are written to the **log buffer**, which exists in memory. The Log Writer background process LGWR writes those changes to the online redo log files upon any **COMMIT** (as well as just before any time that the Database Writers DBWn write any changed pages to disk). Thus we have **write-ahead logging**, a technique whereby Oracle can reconstruct any COMMIT'd database update from the redo logs if necessary.

You can reconfigure the online redo logs while the database is up and running. For example, you could add members to the online redo log groups or even add more groups (up to the limits established in the controlfile through **CREATE DATABASE** command parameters **MAXLOGMEMBERS** and **MAXLOGFILES** respectively). In contrast, recall that you cannot change the control files while the database is up. The database must be either in **NOMOUNT** mode or completely shut down to alter the control files. Monitor the logs through the dynamic views **V$LOG** and **V$LOGFILE**.

If the database is in **NOARCHIVELOG** mode (the default), then Oracle overwrites redo log records as it reuses the online redo log files. To be able to recover the database to the point of the last committed transaction, change the database into **ARCHIVELOG** mode. Now the **Archiver Processes ARCn** write the contents of any full **online redo log file** to disk in the **archived redo log files.** You used to have to set configuration parm **LOG_ARCHIVE_START** to start the archiver processes, but since 10g Oracle automatically starts at least 4 for you when you put the database into ARCHIVELOG mode. The maximum number of archiver processes is 30.

To change a database from the default into ARCHIVELOG mode, here's what you do:

1. Shutdown the database in orderly fashion
   SHUTDOWN {NORMAL | TRANSACTIONAL | IMMEDIATE} ;

2. Start to mount mode
   STARTUP MOUNT ;

3. Change to archivelog mode
   ALTER DATABASE ARCHIVELOG ;

4. Open the database
   ALTER DATABASE OPEN ;

5. Perform a full backup

The last step, performing a full backup, is absolutely necessary. Any previous backup you made while in NOARCHIVELOG mode is no longer valid! Remember that you can only switch to ARCHIVELOG mode after a clean shut down and when the database is in **MOUNT** mode.

14.3 The Flash Recovery Area

The **flash recovery area** is the default disk destination for backup and recovery files. By default these files are written there:

- **Recovery Manager** or **RMAN** backups
- Archived log redo files
- **Database flashback** logs (as used, for example, for recovering an entire database via **Database Flashback**)

Set two configuration parameters to enable the flashback recovery area:

Parameter	Use
DB_RECOVERY_FILE_DEST	The file system directory or ASM disk group assigned to the flashback recovery area. This can be shared among more than one database, each of which will have its own subdirectory structure.
DB_RECOVERY_FILE_DEST_SIZE	Maximum size for the flashback recovery area.

The Database Control panel **Flash Recovery** makes it easy to set these parameters and monitor the size and contents of the flashback recovery area.

15.0 Performing Database Backups

Understanding Oracle's backup and recovery terminology is essential to comprehending how backup and recovery works with Oracle databases. *With Oracle, words have specific meanings that often differ from how they are used with other database products or when discussing relational databases generally.* That is why we'll emphasize Oracle terminology in this section.

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

Remember that an orderly shutdown of the database will checkpoint the system, ensuring that all SCNs match. This is the basis of a consistent backup. You must have **SHUTDOWN NORMAL**, **TRANSACTIONAL**, or **IMMEDIATE** prior to a consistent backup. **SHUTDOWN ABORT** will *not* do.

nconsistent or **online backups** are made while the database is up. Since control file SCNs may vary from datafile header SCNs, if media failure occurs, this kind of backup is first **restored** (where datafile(s) are copied in from a backup medium), then **recovered** (by applying the logs to bring the datafile(s) to a point of consistency with the rest of the database). At the conclusion of recovery all SCNs match. Online backups are also called **open** or **hot backups**.

A **whole database backup** includes all datafiles and the control file. A **partial database backup** lacks all the files required to make it a whole database backup. A whole database backup may -- or may not -- be a consistent database backup. In other words the presence of all files in a database backup does not imply anything about whether the transactional state of all those files is consistent.

A **full backup** includes all blocks of every datafile included in a whole or partial database backup. (This differs from a **whole database backup**; the full backup does *not* necessarily mean that the complete database has been backed up!) In contrast to a full backup, an **incremental backup** makes a copy of all data blocks changed since a previous backup. Incremental backups or "incrementals" are identified as either *level 0* or *level 1* backups. **Level 0** is the base backup level that contains all database blocks, while **level 1** represents the blocks changed since either the level 0 backup or the last level 1 backup. *If no level 0 backup has ever been run and you run a level 1, Oracle will automatically perform a level 0 backup for* you—since level 1 backups are useless for restore/recovery purposes without a level 0 to apply them to. In older Oracle releases, you could specify incrementals higher than level 1. 11g supports numbers higher than level 1 but this is only for backward compatibility.

By default level 1 incremental backups back up all changed blocks since the last level 0 or level 1 backup. A **cumulative incremental backup** is a level 1 backup that backs up all changes since the baseline level 0 backup, regardless of whether you have run any intervening level 1 backups. The benefit to the cumulative incremental backup is that you may be able to apply fewer incremental backups when recovering a database, thus reducing **the mean time to recover** or **MTTR**. The cost is that it may take longer to run than a non-cumulative level 1 backup, since it may contain more changed blocks.

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

1. **Restore** the data from backup (copy data over from a backup file of some sort, possibly transforming it into the proper format for a database datafile, and putting it into proper location as a datafile within the database).

2. **Recover** the database by applying redo log data to bring any lost datafiles up to consistency with the rest of the database.

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

Remember that in NOARCHIVELOG mode, Oracle overwrites the online redo log file data as needed. Thus NOARCHIVELOG mode databases could potentially lose transactions upon recovery, depending on the circumstances. NOARCHIVELOG databases can only be backed up when closed down; they do not support online backups. The **Recovery Manager** or **RMAN** backs up NOARCHIVELOG mode databases when they are in the **MOUNT** state, unavailable for general use. Remember that, in Oracle's terminology, you can never "recover" a NOARCHIVELOG database because you cannot apply redo logs to its restored datafiles.

ARCHIVELOG mode databases retain all transactions through their archived redo logs – so a database can be guaranteed never to lose a single transaction. You can backup ARCHIVELOG databases online. An important Oracle feature is that it is quite possible to have a database for which you only run online backups–never a cold backup–and yet the database and its data is 100% protected from media failure.

Oracle Corp distinguishes between **user-managed backups** and **server-managed backups**. **User-managed backups** mean performing a manual backup yourself using your own software. Usually this means using operating system commands like **cp**, **tar**, or **cpio** under Unix and Linux, or **copy** or **winzip** under Windows. Perhaps your site wrote a shell script that performs the user-managed backup based on these commands.

Server-managed backups involve using Oracle's software to perform the backups. Oracle's primary tool for this purpose is the **Recovery Manager** or **RMAN**. When you backup or recover a database using the Database Control GUI, internally this generates and runs an **RMAN script**, a set of automated RMAN commands. Server-managed backups have many key advantages over user-managed backups including:

- Space savings on the backup media since RMAN never backs up unused database blocks
- Incremental backups (which only back up blocks that have changed since a prior backup)
- Management and control through the Database Control graphical interface
- Automation of backup/recovery processes through RMAN
- A complete scripting language (in RMAN) for backup and recovery automation
- RMAN facilities for intelligent restore/recovery, which reduce error and automate it

15.1 User-Managed Backups

To make a **user-managed consistent backup**, follow these steps:

1. Shutdown the database cleanly (NORMAL, TRANSACTIONAL, or IMMEDIATE)
2. Copy (or "backup") the controlfiles
3. Copy the datafiles
4. Copy the online redo log files

You'll always want to copy also the *archived redo log files,* the *parameter file,* and the *password file*. You never need to back up or copy *tempfiles*, since they contain no data that has persistence beyond the sessions that created them.

To make a **user-managed open backup**, follow these steps:

1. Back up the controlfile with **ALTER DATABASE BACKUP CONTROLFILE**. You can create either a **binary backup** or a **textual file backup** controlfile

2. Put each tablespace and its underlying datafile(s) into **backup mode** by issuing

3. ALTER TABLESPACE tablespace_name BEGIN BACKUP ;

4. Backup the datafile(s) underlying tablespaces in backup mode with operating system copy commands, such as **cp** under Unix/Linux or **copy** under Windows

5. Take each tablespace out of backup mode after copying its datafile(s) to backup

6. Archive the online redo log files

If you opt for textual control file backups, these are written to the directory set by configuration parameter **USER_DUMP_DEST** (or **DIAGNOSTIC_DEST** if it is set). If you ever have to recover a control file with them, you would put the instance in **NOMOUNT** state, then use them to create a new binary controlfile.

When a tablespace is in backup mode, Oracle writes out the complete block image to the log buffer, instead of minimal change vectors. The result is much more log activity for any tablespace in log mode (and more log switches during the back up program).

Always remember that you can *only* perform online backups (user-managed or server-managed) on databases that are in ARCHIVELOG mode.

15.2 Server-Managed Backups

You can perform server-managed backups through the Database Control graphical interface or by RMAN directly. If you use Database Control it generates RMAN scripts for you, and runs them through its own job scheduler You can keep (and alter) these generated scripts as you like.

RMAN generates three kinds of backups:

1. **Backup Set:** a proprietary format containing one or more database datafiles that excludes never-used blocks.

2. **Compressed Backup:** compressed Backup Sets.

3. **Image Copy:** an exact copy of the datafile to back up. These are typically larger than backup sets but can be immediately interchanged with the datafiles they back up without the need for an RMAN **restore** operation.

RMAN launches server processes to perform backups called **channels**. Multiple concurrent channels is how RMAN gets backup parallelism. Each channel directs its output either to **disk** or to **SBT_TAPE**. **SBT_TAPE** can refer to a variety of automated tape or tertiary backup devices.

RMAN backs up the controlfile, the **SPFILE** (dynamic parameter configuration file), the datafiles, and the archivelog files. *RMAN never backs up **online redo log files** or **tempfiles**.*

To run RMAN yourself you issue the **rman** command. You'd typically run it as user **SYS**, since RMAN scripts usually include database SHUTDOWN and STARTUP commands. Any RMAN script to run you refer to by the @ symbol:

```
rman  target  sys/oracle@orcl11g  @my_rman_script.rman
```

RMAN scripts typically group commands within a **run block**. This example RMAN script creates an offline consistent database backup. Remember that the default target for backups is the flashback recovery area:

```
run  {
shutdown immediate ;
startup mount ;
allocate channel d1 type disk ;
backup as backupset database
format 'e:\orabackups\offline_full_whole.bus' ;
alter database open ;    }
```

This next sample RMAN script backs up the entire database online. It uses two channels to demonstrate parallelism. The archived redo log files are backed up and deleted afterwards. RMAN connects to the Oracle database as a user process (the same as any other), and it launches server processes to perform the backup work. For this reason it differs from user-managed backups—it does **not** have to put tablespaces into **backup mode** to perform online backups:

```
run  {
allocate channel t1 type sbt_tape ;
allocate channel t2 type sbt_tape ;
backup as backupset filesperset 6 database ;
backup as backupset archivelog all delete all input ;  }
```

Information on server-managed backups is *always* stored in the database's control file. It may additionally be stored in a separate Oracle database known as the **Recovery Manager Catalog**. These **catalog backups** are required to take advantage of many of RMAN's advanced features.

You can see what backups exist through RMAN's **LIST** command and see what backups are needed by the **REPORT** command. Here are examples:

```
LIST BACKUP OF DATABASE ;
REPORT NEED BACKUP ;
REPORT OBSOLETE ;
```

Useful RMAN commands for managing backups include:

RMAN Command	Use
CROSSCHECK	RMAN compares its repository to actual backups and marks those that are missing as **EXPIRED**.
DELETE EXPIRED	Deletes references to EXPIRED backups from RMAN's Catalog.
DELETE OBSOLETE	Applies a *retention policy* to delete backups no longer needed and remove their Catalog information.
CATALOG	Allows you to add backups to RMAN's Catalog—for example, use this to add user-managed backups to the Catalog.

You can review RMAN reports through the Database Control panel **View Backup Report**. The panel **Manage Current Backups** gives you the graphical interface to the RMAN commands above to manage backups and their RMAN Catalog entries.

You also need to manage the **flashback recovery area**, since this is the default target to where backups are written. The views **V$RECOVERY_FILE_DEST** and **V$FLASH_RECOVERY_AREA_USAGE** help with this. Or use Database Control's **Flash Recovery** panel.

16.0 Performing Database Recovery

Oracle automatically runs its **Health Monitor** when certain conditions occur, or when the DBA requests it. **HM** stores its data in the file system – *not* in the Oracle database – in the directory set by configuration parameter **DIAGNOSTIC_DEST**. You can view the Health Monitor data either through Database Control's **Advisor Central -> Checkers tab**, or by running Oracle-provided package **DBMS_HM**.

The **Data Recovery Advisor** or **DRA** is an Oracle facility for diagnosing and repairing database problems. The DRA uses the Health Monitor data to provide advice on problems and perform recoveries for single-instance databases. DRA cannot yet work with clustered databases (Real Application Cluster or RAC systems) or with Data Guard standby databases.

You can access DRA either through the RMAN command line or through Database Control's **Perform Recovery** panel. The steps for using DRA are:

1. Assess Failures: Collect data about failures by running the Health Monitor
2. List Failures: The **LIST FAILURE** command lists failures and classifies by severity
3. Advise Failures: The **ADVISE FAILURE** command prompts DRA to generate an RMAN script to fix the failure
4. Execute Repair: The **REPAIR FAILURE** command runs the RMAN script(s) to fix the failure(s)

Remember that the ADVISE FAILURE command can only generate scripts to handle those conditions listed by the most recently-run LIST FAILURE command.

The kinds of problems DRA can fix include damage to the *controlfiles*, *datafiles*, and *missing redo log file groups*. In **NOMOUNT** state DRA repairs the controlfiles, and in **MOUNT** or **OPEN** state it repairs **non-critical datafiles**.

The datafiles underlying the **SYSTEM** and **UNDO** tablespaces are **critical**. Whether by DRA or other means, they can only be fixed when the database is down or closed (specifically, in the **MOUNT** state). If they become damaged while the database is in operation the database will abort.

16.1 Kinds of Recovery

Controlfile: The database will abort if *any one* of the multiplexed controlfiles becomes bad or invalid. If you attempt to start a database with a bad controlfile, it will stop in the **NOMOUNT** step. To restore a bad controlfile, you can simply copy it from one of your good ones (since they are multiplexed, you will have more than one). *The database must be down to perform a valid controlfile copy.* Ensure that the **CONTROLFILES** configuration parameter is set properly to reflect how many controlfiles the database has before you start it up. This procedure summarizes these steps for recovering from the loss of a controlfile:

1. Shutdown the database (**SHUTDOWN ABORT** will likely be necessary)
2. Copy over one of the good multiplexed controlfiles to replace the bad one
3. Start the instance in **NOMOUNT** mode (**STARTUP NOMOUNT**)
4. If needed, change the **CONTROLFILES** parameter to reflect the new controlfile
5. Shutdown in orderly fashion (**SHUTDOWN NORMAL**)
6. Start the database normally (**STARTUP**)

Online Redo Log File Member: As long as there is at least one valid member in each online redo log group, the database continues to function. You'll know that a redo log file member is bad if it has the status **INVALID** in the view **V$LOGFILE**. To fix this, just delete and recreate all members of the affected group by a command like this one for group 2:

```
ALTER DATABASE CLEAR LOGFILE GROUP 2 ;
```

If the group is current or active, you may need to switch the logfile, checkpoint, and archive first:

```
ALTER SYSTEM SWITCH LOGFILE ;
ALTER SYSTEM CHECKPOINT ;
ALTER SYSTEM ARCHIVE LOG ALL ;
```

Datafile in NOARCHIVELOG Mode: In NOARCHIVELOG mode databases, the online redo log files are overwritten and their contents are not archived to permanent storage. Therefore you cannot rollforward (you cannot apply redo from archived redo log files). So, the only way to fix a datafile is either to drop it from the database, or to restore the entire database. In Oracle terminology, there can be no recovery, there can only be a full database restore.

This is the same as if you backed up one of your personal files from your laptop's hard disk to a write-once CD or DVD. When you copy the file back from the CD or DVD to your hard disk, the file is current as of the time you backed it up. Any changes to the file on hard disk you may have made since you made your backup are lost.

NOARCHIVELOG mode databases are simple to manage because you don't have to worry about backing up the archived redo logs and managing their disk space. But you can see that their recoverability is limited. ARCHIVELOG mode is much more popular for this reason.

Datafile in ARCHIVELOG Mode: ARCHIVELOG mode databases provide for online restore and recovery of all non-critical database files. A **non-critical datafile** is defined as all database datafiles except for those belonging to the **SYSTEM** and **UNDO** tablespaces.

To fix a non-critical datafile while the database remains up and open:

1. Take the affected tablespace/datafile(s) offline
2. Restore the damaged datafile(s) from the backup media
3. Recover the damaged datafile(s). This means applying redo to them to bring them to currency with the rest of the database. In other words, bring their System Change Numbers or SCNs up to that of the rest of the database by reapplying any changes not reflected in the restored datafile(s)
4. Bring the affected tablespace/datafile(s) back online

Non-critical datafile(s) can also be fixed while the database is down and closed.

Critical datafiles underlying the **SYSTEM** and **UNDO** tablespaces can *only* be fixed when the database is down. The specific steps to follow are:

1. MOUNT the database (by **STARTUP MOUNT)**
2. Restore the damaged datafile(s) from backup media
3. Recover the damaged datafile(s) by applying any required redo logs
4. OPEN the database (by **ALTER DATABASE OPEN**)

17.0 Moving Data

Oracle provides several ways to move data between systems and to load data into databases. The exam focuses on the SQL*Loader utility program, external tables, and Data Pump.

17.1 SQL*Loader

SQL*Loader allows you to bulk-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 **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**. In either case SQL*Loader *runs as a user process*. **Direct path** is faster than conventional path. 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 their **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 using SQL **INSERT** statements. It is slower than Direct path but lacks its restrictions.

Invoke SQL*Loader by the **sqlldr** command, followed by various options. SQL*Loader uses several files for its operations:

- Control: Required. The control file directs SQL*Loader operations.
- Log: Required. This file logs progress and outcomes.
- Data: Optional . Place your load data in-line here.
- Bad: Optional . Holds records that failed validation. If you do not specify a Bad file, SQL*Loader creates one for you.
- Discard: Holds records rejected due to bad format, constraints, etc.
- Reject: Holds records rejected due to record selection criteria.

Here is a simple example of a SQL*Loader control file. **INFILE** specifies the input file name for the data to load, **FIELDS** describes the field separators, and the column names in parentheses describe the columns to load:

```
load data
infile 'e:\bulk_load_data\emp_input.csv'
into table emp
fields terminated by "," optionally enclosed by '"'
(empno, empname, sal, deptno)
```

You run this SQL*Loader program by invoking it from the command line:

```
sqlldr  userid=scott/tiger  control=controlfile_name.ctl
```

To run this as a direct path load, just add the proper parameter:

```
sqlldr  userid=scott/tiger  control=controlfile_name.ctl  direct=true
```

Of course, the user id running SQL*Loader must have proper privileges on the table(s) it tries to load.

17.2 Directory Objects

Directory objects provide a pathname for an external directory or folder. They are required to use external tables and Data Pump, as described in subsequent sections.

Directory objects can be created and used by anyone granted the proper privileges. But they are all owned by user **SYS** and reside in the Data Dictionary. View **ALL_DIRECTORIES** lists them. This example SQL*Plus session shows how to create directory objects:

```
E:\>sqlplus  /  as sysdba
        …
SQL> GRANT CREATE ANY DIRECTORY TO scott;
Grant succeeded.
SQL> connect scott/tiger;
Connected.
SQL> CREATE DIRECTORY scotts_directory AS 'C:\TEMPDB' ;
Directory created.
SQL> GRANT READ ON DIRECTORY scotts_directory TO team_users ;
Grant succeeded.
SQL> GRANT WRITE ON DIRECTORY scotts_directory TO team_developers;
Grant succeeded.
```

17.3 External Tables

External tables are tables that reside *in operating system files*—completely outside of the Oracle database. From inside oracle you gain access to them through directory objects.
Advantages to external tables are:

- Avoids the cost and performance hit of loading the data into Oracle
- Allows you to query existing file system tables using SQL

- (including joins, views, subqueries)
- You can bulk write to external tables using Data Pump. Use a **CREATE TABLE … AS SELECT** statement referring to Data Pump to accomplish this

With external tables you cannot:

- Use SQL update DML to update their data
- Build indexes on them
- Apply constraints or triggers

This example creates an external table that is similar to HR.DEPARTMENTS. The key clause that defines an external table is **ORGANIZATION EXTERNAL**:

```
CREATE TABLE dept_external (
              deptno   NUMBER(6),
             dname    VARCHAR2(20),
             loc      VARCHAR2(25)  )
ORGANIZATION EXTERNAL
(TYPE oracle_loader
             DEFAULT DIRECTORY admin
             ACCESS PARAMETERS
 ( RECORDS DELIMITED BY newline
             BADFILE 'ulcase1.bad'
             DISCARDFILE 'ulcase1.dis'
             LOGFILE 'ulcase1.log'
             SKIP 20
             FIELDS TERMINATED BY "," OPTIONALLY ENCLOSED BY '"'
  (deptno             INTEGER EXTERNAL(6),
   dname              CHAR(20),
   loc                CHAR(25)  )  )
LOCATION ('ulcase1.ctl')  )
REJECT LIMIT UNLIMITED ;
```

17.4 Data Pump

Data Pump was introduced in Oracle 10g and supercedes the older **Export and Import Utilities** (which are still shipped with 11g). Data Pump's export and import utilities are called **expdb** and **impdb**, respectively. Even though it may be launched by user processes like SQL*Plus or OEM Database Control, Data Pump is a *server-side process*. This is faster than *client-server user processes* such as SQL*Loader and the old Export/Import utilities. The external file formats written to by Data Pump and the older Export/Import utilities are not compatible—neither system can read or write the other's.
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 **expdb** and **impdb** utilities), through the PL/SQL package **DBMS_DATAPUMP** and its procedures, or through the OEM Database Control graphical interface.

Data Pump has several modes of operation. Its **Direct Path** is analogous to SQL*Loader's Direct Path. It bypasses the database buffer cache, does not generate undo, and optionally does not generate redo. Rather than perform a COMMIT, it just writes above the table's **High Water Mark** or **HWM** and adjusts the HWM when it's done.

Data Pump's **External Table Path** is rather like SQL*Loader's conventional path. It uses the database buffer cache, generates both undo and redo, and operates like regular SQL statements.

Each Data Pump job uses two queues:

- Control queue: For controlling the work and dividing it into tasks
- Status queue: For monitoring and status

Data Pump writes to three types of files:

- Dump files: Contains data and metadata (formatted with XML tags)
- Log files: Tracks job activities
- SQL files: Contains extracted SQL DDL statements for objects

You specify the directory or directories Data Pump uses in several different manners. From highest down to lowest priority they are:

- Data Pump job file settings
- Data Pump job global parameter
- **DATA_PUMP_DIR** environmental variable
- **DATA_PUMP_DIR** directory object

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}
- Transport Tablespace **TRANSPORT_TABLESPACES =** {list}

Data Pump has many special features. It uses *parallelism*, both through multiple Data Pump worker processes and through the number of parallel execution servers for each worker process. It can *remap objects* between output and input, renaming objects or changing their schemas. Output files can be *encrypted* and *compressed*.

Data Pump's **Network Mode** facilitates fast data transfer between machines and across networks. The Data Pump export job on one computer writes to a database link that is used as input by a Data Pump import job on another computer.

Data Dictionary views for managing and tracking Data Pump 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

There are many parameters to the **expdb** and **impdb** utilities. These lists focus on key ones for the test. Here are the key **expdb** 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, **expdb** 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. Cannot 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 or SCN.
FLASHBACK_TIME	Export the data as of this consistent Time.
FULL	= Y means export the full Database.
HELP	Displays 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** (moving tablespaces between databases).
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 (**impdb**) utility. Here are some other parameters for **impdb**:

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 for when an import table already exists: **APPEND, IGNORE, REPLACE, SKIP,** or **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.*

Here's an example of how to use Data Pump. This example shows the background work you need to do to make it work. First make a directory or folder on your operating system to which you will write the Data Pump files:

```
C:> mkdir data_transit
```

Then create the required directory object in Oracle and assign required permissions for use:

```
sqlplus / as sysdba

SQL> CREATE DIRECTORY my_dir AS 'C:\data_transit';
SQL> GRANT READ ON DIRECTORY my_dir TO scott;
SQL> GRANT WRITE ON DIRECTORY my_dir TO scott;
```

Now you can export data from a table into that directory with Data Pump (assuming that the user id you use has the proper table privileges). You should see the output data as a **.dmp** file after execution, and you'll also see a **.log** file:

```
expdp scott/tiger directory=my_dir tables=hr.employees
```

Similarly, you can use Data Pump to import the data into a like table:

```
impdp scott/tiger directory=my_dir tables=scott.employees
```

18.0 Intelligent Infrastructure Enhancements

The OEM **Support Workbench** is the graphical interface for viewing and managing the **Automatic Diagnostic Repository** or **ADR**. The ADR is the centralized storage area for all diagnostic information including the instance's Alert log, trace files, dumps, and Health Monitor reports. It's used for diagnostics and problem solving and its data is often sent to **Oracle Support Services** when one creates a **Service Request** or **SR**.

The ADR exists in regular operating system files under the root directory set by configuration parameter **DIAGNOSTIC_DEST**. Under this root folder each database and instance has its own sub-directory. The **ADR_HOME** for an instance might look like this:

```
ORACLE_BASE/diag/database_name/instance_name
For example: /u01/app/oracle/orcl11g/orcl11g
```

Manage and view ADR with either the command line tool **ADR Command Line Interface** (**ADRCI**) or OEM Database Control's **Support Workbench**. The latter is graphical and much easier to use. The Support Workbench interfaces with Oracle's web-based **Metalink** support service to manage **Support Requests** or **SR**'s.

Database Control tracks patches for you and even downloads them, to act as a complete patch management system. A big improvement over earlier Oracle releases is that the interface makes it easy to upload all required information in the ADR to Oracle Worldwide Support Services. This was time-consuming and error prone with prior manual methods.

18.1 Patches

There are three classes of Oracle patches. From most specific to most comprehensive they are:

Patch Type	Use
Interim	These address a specific individual problems
CPU (Critical Patch Update)	Cumulative, regression-tested patches for a specific release level. Includes all dependent patches
Patch Sets	Cumulative set of regression-tested patches that increment the product release level. E.g.: from 11.1.0.6 to 11.1.0.7

Patches can be applied either from the command line or through OEM Database Control. The latter provides an easier interface and greater integration with Metalink and Oracle Support Services. Internally OEM Database Control and its Support Workbench issue the **opatch** command to the command line on your behalf. The Support Workbench **Patch Advisor** gives advice on which patches are needed, downloads them, and coordinates their application. Database Control's **Software Patching** panels lead you through this process:

```
Select patch          ->
Select destination    ->
Set credentials       ->
Stage or Apply        ->
Schedule              ->
Summary
```

If you instead work with patches from the command line, you'll want to check prerequisites prior to applying the patch:

```
$ORACLE_HOME/Opatch/opatch  lsinventory  –detail
```

and then eventually apply the patch:

```
$ORACLE_HOME/Opatch/opatch  apply  patch_to_apply
```

Practice Questions

Chapter 1

1. Which are required SGA components? Choose all that apply:

   - ❍ A. Log Buffer
   - ❍ B. Java Pool
   - ❍ C. Large Pool
   - ❍ D. Shared Pool
   - ❍ E. Database Buffer Cache
   - ❍ F. Streams Pool
   - ❍ G. Program Global Area

Chapter 2

1. Match each Oracle environmental variable with its description:

   - ❍ A. ORACLE_SID
   - ❍ B. ORACLE_HOME
   - ❍ C. ORACLE_BASE
   - ❍ D. DISPLAY
   - ❍ E. LD_LIBRARY_PATH

__ Must be set for monitor to work properly in Unix/Linux when using OUI
__ Top-level directory for installing multiple Oracle products
__ Top-level directory install of an individual Oracle product
__ Location of dynamically linked libraries
__ Identifier for the Oracle instance

Chapter 3

1. Whether run through DBCA or at the SQL*Plus command line, which files are created by the **CREATE DATABASE** command? Choose all that apply:

   - ❍ A. The static parameter file (the **PFILE**)
   - ❍ B. The dynamic parameter file (the **SPFILE**)
   - ❍ C. The controlfile
   - ❍ D. The online redo log files
   - ❍ E. The password file
   - ❍ F. The operating system authentication files
   - ❍ G. The SYSAUX tablespace datafile
   - ❍ H. The SYSTEM tablespace datafile

Chapter 4

1. Which of the following statements are true about **SPFILE's**? Choose all that apply:

   - ❍ A. You can directly edit them in any text editor to change parameters
   - ❍ B. You can output their contents to PFILE's by issuing a command
   - ❍ C. They allow you to change any Oracle parameter while the database is running
   - ❍ D. Most sites use them instead of PFILE's because of their additional features
   - ❍ E. The default scope for ALTER SYSTEM is BOTH
   - ❍ F. If the instance is started with a PFILE, then ALTER SYSTEM… SCOPE=SPFILE will fail

Chapter 5

1. Which of these statements about Shared Server Architecture is true? Choose all that apply:

   - ❍ A. There is one input queue per dispatcher
   - ❍ B. There is one response queue per dispatcher
   - ❍ C. Dispatchers communicate with user processes to handle user queries
   - ❍ D. Each user process has its own server process
   - ❍ E. Shared Server Architecture can make more efficient use of server memory

Chapter 6

1. On which device types can you place database datafiles? Choose all that apply:

   - ❍ A. Raw devices
   - ❍ B. Local filesystems
   - ❍ C. Clustered filesystems
   - ❍ D. ASM
   - ❍ E. All of the above

Chapter 7

1. Match each authentication type with its proper description.

   - ❍ A. Password authentication
   - ❍ B. Password file authentication
   - ❍ C. Global authentication
   - ❍ D. External authentication

__ Authenticates users from a password file saved in the operating system's filesystem
__ Authenticates user accounts from security information contained in the Data Dictionary
__ Authenticates users from a service such as a Kerberos or Radius server, or the Windows native authentication service
__ Authenticates users from an LDAP-compliant directory server such as OID

Chapter 8

1. You have a large table with high cardinality data. Which one of the following index types would be most effective? Choose the best answer:

   - ❍ A. Bitmap index with composite key
   - ❍ B. Bitmap index
   - ❍ C. Function-based B*tree index
   - ❍ D. Function-based bitmap index
   - ❍ E. B*tree index

Chapter 9

1. You start a SQL*Plus session with all default options enabled. You issue these SQL statements, then your laptop's power cuts off after you issue the last statement and have sent it to the database server. What is the result?

```
SQL> select * from hr.job_history ;
SQL> update hr.job_history set department_id = null ;
SQL> create table my_history as select * from hr.history ;
```

- A. All three statements are rolled back
- B. The first two statements are committed but the third is rolled back
- C. The first two statements are committed. The third statement creates the table but does not populate it with any data since that is a DML UPDATE operation.
- D. All three statements are committed

Chapter 10

1. Which of these statements about undo is true? Choose all that apply:

- A. One database can have many undo tablespaces
- B. One transaction can use many undo segments
- C. One undo segment can protect more than one transaction
- D. Rollback segments are preferrable to undo tablespaces for performance reasons
- E. Multiple undo tablespaces can be active as required by the transaction load in single- instance systems
- F. One undo segment can be sectioned among more than one datafile

Chapter 11

1. You need to audit updates to a table containing sensitive data. You only want to audit updates to the specific rows that actually contain the sensitive data. Which auditing method should you choose?

- A. Database auditing
- B. Database auditing with AUDIT_TRAIL set to ON
- C. SYSDBA auditing
- D. Value-based auditing with triggers
- E. Fine-Grained Auditing

Chapter 12

1. What is the best way to capture optimizer statistics? Choose the best answer:

- A. Manually run the Oracle package **DBMS_STATS** on each table individually to ensure complete statistics and optimal performance
- B. When you create the database using DBCA, take its default option to automate statistics gathering through its scheduled job
- C. Write shell scripts that run the SQL **ANALYZE** command against each table consecutively
- D. Schedule the MMON background process so that it will run at intervals you specify

Chapter 13

1. In an Oracle 11g database, what is the best way to manage memory? Choose the best answer:

   - A. Set the two configuration parameters **SGA_TARGET** and **PGA_AGGREGATE_TARGET** to split the available memory between the SGA and PGA most efficiently.
   - B. Configure individual SGA components: the Shared Pool, the Database Buffer Cache, and the Log Buffer. Let all other components default.
   - C. Set the configuration parameter **MEMORY_TARGET** to the available memory for Oracle and let Oracle manage that memory itself.
   - D. Set the configuration parameter **MEMORY_MAX_TARGET** to the available memory for Oracle and let Oracle manage that memory itself.

Chapter 14

1. A new administrator on staff somehow manages to use Windows **delete** or **erase** commands to delete several Oracle datafiles from the Windows filesystem. What kind of failure does this cause and how do you fix it? Choose the best answer:

   - A. This causes SQL statement failures against the tables that reside in those datafiles. Stop issuing the statements!
   - B. Oracle's instance recovery (so called **crash recovery**) will automatically handle this next time the database is started. PMON and SMON work together to fix this.
   - C. Since this is a user error the best fix is to issue flashback queries against the affected tables to recover their data.
   - D. This is equivalent to a media failure. You need to follow the procedures to restore and recover datafiles affected by media failure.

Chapter 15

1. Which file types does RMAN **not** back up? Choose all that apply:

   - A. Tablespace datafiles
   - B. PFILE
   - C. SPFILE
   - D. Online redo log files
   - E. Archived redo log files
   - F. Temporary tablespace datafiles
   - G. Password file
   - H. Oracle binaries (the ORACLE_HOME directory software)
   - I. Oracle network configuration files

Chapter 16

1. What is the usage sequence for **Data Recovery Advisor** or **DRA** to restore/recover non-critical data files in a Real Application Cluster or RAC environment? Choose the best answer:

   - A. LIST FAILURE, ADVISE FAILURE, REPAIR FAILURE
   - B. ADVISE FAILURE, LIST FAILURE, REPAIR FAILURE
   - C. DRA cannot help you restore/recover non-critical datafiles
   - D. DRA cannot help you restore/recover critical datafiles
   - E. DRA cannot help you restore/recover non-critical datafiles for RAC systems

Chapter 17

1. You need to load lots of data quickly into an Oracle database from a flat file dumped from a SQL Server database. What is your best approach? Choose the best answer:

    - ❍ A. Use a SQL INSERT statement to load the data
    - ❍ B. Use the CREATE TABLE statement to access the data as an EXTERNAL TABLE
    - ❍ C. Use SQL*Loader conventional path to load the data
    - ❍ D. Use SQL*Loader direct path to load the data
    - ❍ E. Write a program that reads data from SQL Server and INSERTs it into Oracle tables

Chapter 18

1. Match the kinds of patches with their proper descriptions below.

    - ❍ A. Patch set
    - ❍ B. Interim patch
    - ❍ C. CPU patch

__ This patch is designed to address a single specific problem.
__ This is a cumulative set of patches that increments the product release level.
__ This is a cumulative set of patches that does not increment the product release level.

Answers & Explanations

Chapter 1

1. ANSWER: A, D, F

Required SGA structures are the Log Buffer, Shared Pool, and the Database Buffer Cache. Answer B is incorrect because the Java Pool is an optional part of the SGA. Answer C is incorrect because the Large Pool is an optional part of the SGA. Answer F is incorrect because the Streams Pool is an optional part of the SGA. Answer G is incorrect because the Program Global Area is a separate memory structure and is not part of the SGA at all.

Chapter 2

1. ANSWER: D, C, B, E, A

Answer D **DISPLAY** sets a value for using a display monitor with the Oracle Universal Installer or OUI. It is required that you set this for Unix/Linux systems. Answer C **ORACLE_BASE** is the high-level directory for installing one or more Oracle products. Answer B **ORACLE_HOME** is the high-level directory within ORACLE_BASE for installing an individual Oracle product, such as a database. Answer E **LD_LIBRARY_PATH** is the location where Oracle can find dynamically-linked libraries. It is used for Unix/Linux systems. Answer A **ORACLE_SID** is the instance identifier. **SID** stands for *System Identifier*.

Chapter 3

1. ANSWERS: C, D, G, H

The **CREATE DATABASE** command creates the controlfile, the online redo log files, and the datafiles underlying the required **SYSAUX** and **SYSTEM** tablespaces. Answers A and B are incorrect because while DBCA can create parameter files, these are not created by the CREATE DATABASE command. Answer E is incorrect because the CREATE DATABASE command does not create a password file. Answer F is incorrect because the CREATE DATABASE command does not create any operating system files for user authentication.

Chapter 4

1. ANSWERS: B, D, E, F

B is correct because you can issue this command to generate a PFILE from an SPFILE:

```
CREATE PFILE [='pfilename'] FROM SPFILE [='spfilename'] ;
```

D is also correct. SPFILE's offer many benefits over PFILE's including the ability to change dynamic parameters while the database is up and running. Answer E is correct because when you change a parameter using ALTER SYSTEM the default is SCOPE=BOTH. Answer F is correct because if you start up an instance with a textual parameter file or PFILE, using ALTER SYSTEM… SCOPE=SPFILE will fail. Answer A is incorrect because the SPFILE is binary and you cannot edit it with a text editor. Answer C is incorrect because you cannot change every configuration parameter while the database is running, just the dynamic parameters.

Chapter 5

1.ANSWERS: B, C, E

B is correct because each dispatcher has its own response queue. C is correct because dispatchers communicate with user processes throughout their sessions to handle their queries. E is correct because one key motivation for using Shared Server Architecture is to optimize use of memory on the database server. Answer A is incorrect because there is a common shared input queue for the dispatchers. Answer D is incorrect because this describes Dedicated Server Architecture, where each user process has its own dedicated server process.

Chapter 6

1. ANSWER: E

You can place datafiles on all the listed device types. Raw devices are largely superceded by clustered filesystems and Oracle's **Automated Storage Management** or **ASM**, but are still supported. Local filesystems are the operating system filesystems supported by the computer on which you installed Oracle. Clustered filesystems provide for sharing datafiles between two or more computers in the cluster. They include Oracle's **Oracle Clustered File System** or **OCFS**. ASM was specifically designed for optimal performance with clustered systems running under Oracle's **Real Application Clusters** or **RAC**.

Chapter 7

1. ANSWER: B, A, D, C

Password file authentication uses a password file stored in the operating system's filesystem for authentication. This is also called **operating system authentication**. Password authentication uses Oracle's own Data Dictionary and the user account information it stores. External authentication uses external services for authentication. If you use Oracle's **Advanced Security Option** or **ASO** it could be a Kerberos or Radius server. Or it could be Windows native authentication service on Windows computers. Global authentication uses a **Lighweight Directory Access Directory** or **LDAP** server like Oracle's **Oracle Internet Directory** or **OID** for centralized authentication and administration.

Chapter 8

1. ANSWER: E

A standard B*tree index works most effectively with large tables having high data cardinality. **High data cardinality** means there will be a high percentage of unique values in the proposed index key. Answers A and B are incorrect because bitmap indexes work best with low-cardinality data. Answer C is incorrect because high data cardinality does not mean you would use a function-based index. Function-based indexes apply a function to one or more columns to provide the index key. This has nothing to do with data cardinality. Answer D is incorrect for the same reason as answers A and B. It refers to a bitmap index.

Chapter 9

1. ANSWER: D

Data Definition Language or **DDL** statements like **CREATE TABLE** are automatically committed as long as it arrives and runs at the database server, even if your session is later cut off. This commits all prior statements issued in the session, regardless of whether they are **Data Manipulation Language** or **DML**. Answer A is incorrect because the CREATE TABLE commits the three statements. Answer B is incorrect because the CREATE TABLE commits all three statements. Answer C is incorrect because a single SQL statement either succeeds or fails in its entirety. You can *never* have a situation in which one part of a SQL statement succeeds and another part of it fails.

Chapter 10

1. ANSWER: A, C, F

A is correct because a database can have many undo tablespaces, but only one of those undo tablespaces can be active at any one time. Answer C is correct because undo segments can protect more than one transaction, though Oracle is designed to limit the number of transactions a single undo segment will protect for performance reasons. Answer F is correct because a single undo segment will grow and can be divided among more than one datafile if necessary. Answer B is incorrect because a single transaction is always protected by just one undo segment. Answer D is incorrect because undo tablespaces replaced rollback segments, not the other way around. Undo is better than rollback segments for administrative ease. Answer E is incorrect because only one undo tablespace can ever be active at one time for a single-instance configuration. Remember that in multiple-instance or RAC systems, each instance will have its own active undo tablespace.

Chapter 11

1. ANSWER: D

Since you need to audit based on values within rows, triggers are required. They can fire per row, inspect the values involved, and cannot be circumvented. Answer A is incorrect because Database Auditing can track updates on particular objects but provides no facility to inspect values within rows. Answer B is incorrect for the same reason as answer A. Also answer B makes no sense, since by definiton **AUDIT_TRAIL** must be set to enable Database Auditing, and **ON** is not one of its allowable values. Answer C is incorrect because SYSDBA Auditing tracks statements issued by users connected as **SYSDBA**. It does not fulfill the stated requirements. Answer D is incorrect because Fine-Grained Auditing or **FGA** cannot distinguish between rows by inspecting column values.

Chapter 12

1. ANSWER: B

When you create a database using **Database Configuration Assistant** or **DBCA**, by default Oracle will create and schedule a job for automatic statistics collection. Answer A is incorrect because you could do this manually, but you will gain no more complete statistics or any better performance than if you had allowed Oracle to automate the process for you. Answer C is incorrect for the same reason as answer A. You could take this approach but it costs you extra effort for no apparent gain. Answer D is incorrect because MMON runs every 60 minutes by default. You don't need to schedule it yourself. Just ensure that configuration parameter **STATISTICS_LEVEL** is set to **TYPICAL** (the default) or **ALL** for this to occur. Should you set **STATISTICS_LEVEL** to **BASIC** you will *not* automatically collect optimizer statistics.

Chapter 13

1. ANSWER: C

In 11g, Oracle has finally completely automated memory management. All you have to do is figure out how much memory to give to Oracle, then set configuration parameter **MEMORY_TARGET** to that value. Answer A is incorrect because this is how you would set memory management for an Oracle 10g database. At that time you had to set two configuration parameters to allocate memory to the SGA and PGA separately. With 11g you no longer have to split memory like this, just allocate all memory by parameter MEMORY_TARGET. Answer B is incorrect because you could manually allocate memory like this if you wanted to, but it is both easier and more efficient to let Oracle do the job itself. Answer D is incorrect because you set parameter **MEMORY_TARGET** to the amount of memory to allocate to Oracle, not **MEMORY_MAX_TARGET**. MEMORY_MAX_TARGET is the maximum value you can dynamically alter MEMORY_TARGET up to.

Chapter 14

1. ANSWER: D

Whatever the reasons for the loss of the datafiles, when datafiles go missing it looks like a case of media failure to Oracle. Follow the procedures to restore and recover datafiles from media failure. Answer A is incorrect because while SQL statements will now fail against the missing table(s), you need to follow media failure procedures to restore and recover the affected datafile(s). Answer B is incorrect because Oracle's automatic instance recovery handles transaction failures due to disorderly shutdown, not loss of datafile(s). Answer C is incorrect because flashback query is not a fix for Oracle datafile(s) deleted from the operating system's filesystem. It is used when a user inadvertently deletes needed rows from tables from within Oracle, such as through a SQL **DELETE** statement.

Chapter 15

1. ANSWER: B, D, F, G, H, I

RMAN does not back up static parameter files (PFILE's), online redo log files, temporary tablespace datafiles, and the password file. The PFILE and password file are OS filesystem files. The online redo log files and temporary tablespace datafiles hold data not needed for restore/recovery. RMAN also does not back up the Oracle binaries (the Oracle software itself), residing in the **ORACLE_HOME** directory. Finally, RMAN does not back up the optional textual network configuration files. These include **TNSNAMES.ORA** and **SQLNET.ORA**. Answer A is incorrect because regular permanent tablespace datafiles are key to what RMAN backs up, since they are required for restore and recovery. Answer C is incorrect because RMAN backs up the SPFILE for restore and recovery. Answer E is incorrect because RMAN backs up and manages archived redo log files for restore and recovery.

Chapter 16

1. ANSWER: E

In 11g DRA only helps with single-instance systems. It cannot work with RAC clustered systems or with Data Guard standby databases. (Sometimes you can work around this limitation by mounting a RAC database in single-instance mode.) Answer A is incorrect because answer D is correct. These steps are correct for a single-instance system but the question asks for help with a RAC clustered system, and DRA cannot help with RAC systems. Answer B is incorrect for the same reason as answer A, plus the sequence of steps is incorrect (answer A has the proper sequence of steps for a single-instance system). Answers C and D are incorrect because DRA only works with single-instance systems, not the RAC cluster the question asks about.

Chapter 17

1. ANSWER: D

The quickest, best way to load this data into Oracle would be to use SQL*Loader and its direct path. Answer A is incorrect because the SQL INSERT statement is much slower than SQL*Loader direct path in loading data. Answer B is incorrect because it does not load the data into Oracle, as the question asks. Instead it makes the data accessible as a flat file external to the Oracle database. Answer C is incorrect because SQL*Loader conventional path uses INSERT statements to load the data. This is always slower than the direct path load. Answer E is incorrect because the program would use SQL INSERT statements. This will be slower than a bulk load using SQL*Loader direct path.

Chapter 18

1. ANSWER: B, A, C

Interim patches are one-off patches that address a specific problem or fix. **Patch Sets** are cumulative, integrated, and regression-tested, and they increment the product release level. **CPU** or **Critical Patch Updates** are cumulative, integrated, and regression-tested, but they do not increment the product release level.