KARPAGAM ACADEMY OF HIGHER EDUCATION

(Deemed to be University)
(Established Under Section 3 of UGC Act 1956)
Coimbatore - 641021.

DEPARTMENT OF CS, CA & IT

SYLLABUS

17CSP203	**ORACLE 10G ADMINISTRATION**	**4H – 4C**
Instruction Hours / week: L: 0 T: 0 P: 4	**Marks:** Int: **40** Ext: **60**	Total: **100**

SCOPE:
This course is designed to give a firm foundation in basic database administration.

OBJECTIVES:

- Install the Database.
- Back up and recover data.
- Administer users and manage data.
- Transport data between databases.
- Configure the network.
- Understand the Oracle database architecture and how its components work and interact with one another.
- Use performance monitoring database security user management and backup/recovery techniques.
- Create an Operational Database

UNIT-I
Oracle DBA'S**:** oracle DBA's role - DBA job classification - types of databases. Oracle database 10g architecture: database structures - processes - memory structures - database transaction. Creating an oracle databases**:** creating the database - creating the parameter file-creating a new database - using a server parameter file.

UNIT-II
Schema management**:** creating and managing table spaces – indexes - materialized views. Oracle transaction management: oracle transactions - transaction properties - transaction concurrency control - isolation levels - implementing oracle's concurrency control - using undo data to provide read consistency - transaction query - discrete transactions -autonomous transactions - resumable space allocation.

UNIT-III
Loading and transforming data**:** overview of extraction transformation and loading - using external tables to load data - transforming data. Using data pump export and import: introduction - performing exports and imports -monitoring - transportable table spaces.
Managing and monitoring operational databases: types of oracle performance statistics -server generated alerts - automatic workload repository - active session history - undo and MTTR advisors.

UNIT-IV

User management and database security: managing users - the database resource manager - controlling access to data - auditing database usage - authenticating users -enterprise user security - database security do's and don'ts. Backing up databases: backing up oracle databases - the recovery manager - backing up control file - oracle back up tool – user managed backups - database corruption detection - enhanced data protection for disaster recovery. Database recovery: types of database failures - oracle recovery processes - performing recovery with RMAN - media recovery scenarios.

UNIT-V

Improving database performance**:** SQL query optimization - approach to oracle performance tuning - optimizing oracle query processing - oracle optimization and oracle cost based optimizer - writing efficient SQL - DBA's role to improve SQL processing - SQL performance tuning tools - explain plan - SQL tuning advisor - simple approach to tuning SQL statement. Performance tuning: Tuning the instance - introduction to instance tuning - automatic performance tuning vs. dynamic performance views - tuning oracle memory - evaluating system performance - measuring IO performance - measuring instance performance - simple approach to instance tuning.

SUGGESTED READINGS

TEXT BOOK

T1. Sam, R. Alapati., & John Watson. (2007). Expert Oracle Database 10g Administration(1st ed.). New Delhi: Springer (India) Pvt Ltd.

REFERENCE BOOKS

R1. April Wells. (2006). Oracle DB Administration (1st ed.). New Delhi: Dream Tech Press
R2. Ivan Bayross. (2006). Oracle 10g DB with HTMLDB (1st ed.). New Delhi: BPB Publications.
R3. Jay Bayross. (2006). Oracle 10g Developer Suite (1st ed.). New Delhi: BPB Publications.

Web Sites

W1. download-uk.oracle.com
W2. cse.psu.edu
W3. dba-oracle.com
W4. otn.oracle.com
W5. oracle.com
W6. techonthenet.com/oracle
W7. forums.oracle.com

KARPAGAM ACADEMY OF HIGHER EDUCATION

(Deemed to be University)
(Established Under Section 3 of UGC Act 1956)
Coimbatore - 641021.

DEPARTMENT OF CS, CA & IT

LESSON PLAN

FACULTY NAME: K.BANUROOPA **BATCH: 2017-2019**
SUBJECT : ORACLE 10G ADMINISTRATION **SUB.CODE : 17CSP203**
CLASS : I - M.SC (CS) **SEMESTER: II**

S.NO	Lecture Duration	Topics to be Covered	Support Materials
		UNIT-1	
1.	1	**Oracle DBA'S:** oracle DBA's role	T1: 3-8, W1
2.	1	DBA job classification , types of databases	T1: 8-9
3.	1	**Oracle database 10g architecture:** database structures	T1: 99-112, W2
4.	1	Oracle Processes	T1: 113-119
5.	1	Memory structures , database transaction	T1: 120-130
6.	1	**Creating an oracle databases:** creating the database	T1: 329-333
7.	1	Creating the parameter file	T1: 333-339
8.	1	Creating the parameter file-control file	T1: 340-357
9.	1	Creating a new database	T1: 358-373
10.	1	Using a server parameter file.	T1: 374-377
11.	1	Recapitulation and Discussion of Possible Questions	
		Total Periods Planned for Unit I	**11**
		UNIT II	
1.	1	**Schema management:** creating and managing table spaces - Locally and Dictionary-Managed Tablespaces	T1:149-153
2.	1	- Creating Tablespaces	T1:154-159
3.	1	- Removing Tablespaces, - Adding Space to a Tablespace, - Renaming Tablespaces	T1:160-174 W3
4.	1	indexes	T1:196-202
5.	1	materialized views	T1: 209-214
6.	1	**Oracle transaction management**: oracle transactions transaction properties	T1:225-228 W4
7.	1	Transaction concurrency control , isolation levels implementing oracle's concurrency control	T1:229-236
8.	1	Using undo data to provide read consistency , transaction query	T1:243-256
9.	1	Discrete transactions	T1:267
10.	1	Autonomous transactions	T1:268
11.	1	Resumable space allocation	T1:269

12.	1	Recapitulation and Discussion of Possible Questions	
		Total Periods Planned for Unit II	**12**
		UNIT III	
1.	1	**Loading and transforming data:** Overview of extraction transformation and loading	T1:539-540,W4
2.	1	Using external tables to load data	T1:559-569
3.	1	transforming data	T1:570-583
4.	1	**Using data pump export and import**: Introduction	T1:589-598
5.	1	performing exports and imports	T1:599-620
6.	1	Monitoring , transportable table spaces	T1:621-624
7.	1	**Managing and monitoring operational databases:** types of oracle performance statistics	T1:823-827 W5
8.	1	Server generated alerts	T1:828-833
9.	1	Automatic workload repository	T1:834-844
10.	1	Active session history - undo and MTTR advisors.	T1:845-855
11.	1	Recapitulation and Discussion of Possible Questions	
		Total Periods Planned for Unit III	**11**
		UNIT IV	
1.	1	**User management and database security**: managing users	T1:421-430,W6
2.		the database resource manager	T1:431-441
3.	1	Controlling access to data,	T1:442-461
4.	1	auditing database usage	T1:461-471
5.	1	Authenticating users, enterprise user security	T1:471-482
6.	1	database security do's and don'ts	T1:482-490
7.	1	**Backing up databases**: backing up oracle databases	T1:631-640
8.	1	The recovery manager	T1:648-679
9.		backing up control file, oracle back up tool	T1:680-686
10.	1	User managed backups, database corruption detection, enhanced data protection for disaster recovery	T1:686-698 W6
11.	1	**Database recovery:** types of database failures , oracle recovery processes	T1:699-706 W7
12.	1	Performing recovery with RMAN, media recovery scenarios.	T1:707-711
13.	1	Recapitulation and Discussion of Possible Questions	
		Total Periods Planned for Unit IV	**13**
		UNIT V	
1	1	**Improving database performance:** SQL query optimization-approach to oracle performance tuning	T1:937-942 W2
2	1	optimizing oracle query processing, oracle optimization and oracle cost based optimizer	T1:943-956
3	1	writing efficient SQL, DBA's role to improve SQL processing, SQL performance tuning tools	T1:957-973 W2
4	1	Explain Plan, SQL tuning advisor, simple approach	T1:974-1000

		to tuning SQL statement	
5	1	**Performance tuning:** Tuning the instance, introduction to instance tuning	T1:1001-1002, W3
6	1	Automatic performance tuning vs. dynamic performance views, tuning oracle memory	T1:1003-1023
7	1	evaluating system performance, measuring IO performance	T1:1024-1031
8	1	Measuring instance performance, simple approach to instance tuning.	T1:1032-1067, W6
9	1	Recapitulation and Discussion of Possible Questions	
10	1	Discussion of Previous ESE Question Papers	
11	1	Discussion of Previous ESE Question Papers	
12	1	Discussion of Previous ESE Question Papers	
		Total Periods Planned for Unit V	**12**
		Total Periods	**60**

TEXT BOOK

T1. Sam, R. Alapati., & John Watson. (2007). Expert Oracle Database 10g Administration (1st Ed.). New Delhi: Springer (India) Pvt Ltd.

REFERENCE BOOKS

R1. April Wells. (2006). Oracle DB Administration (1st Ed.). New Delhi: Dream Tech Press
R2. Ivan Bayross. (2006). Oracle 10g DB with HTMLDB (1st Ed.). New Delhi: BPB Publications.
R3. Jay Bayross. (2006). Oracle 10g Developer Suite (1st Ed.). New Delhi: BPB Publications.

WEB SITES

W1. download-uk.oracle.com
W2. oracle-dba-online.com/
W3. dba-oracle.com
W4. otn.oracle.com
W5. oracle.com
W6. techonthenet.com/oracle
W7. forums.oracle.com

Unit I

SYLLABUS:

Oracle DBA'S**:** oracle DBA's role - DBA job classification - types of databases. Oracle database 10g architecture: database structures - processes - memory structures - database transaction. Creating an oracle databases**:** creating the database - creating the parameter file-creating a new database - using a server parameter file.

Introduction: Oracle 10g

Before going into the oracle 10g, first we will know the difference between Oracle 8i, 9i, 10g, 11g?

Here i stands for internet in oracle 9i.The *g* in Oracle Database 10*g* stands for "grid." The idea is to enable software to access spare processing power across networks (grids) of inexpensive servers.

8i -> 9i - Real Application Clusters, PL/SQL enhancements, XMLType enhancements
9i -> 10g - ASM space management, tracing and diagnostics, DML error logging, async commit, more PL/SQL changes, table/column encryption, restore points

Grid computing: In basic terms, grids are clusters of computers or servers that are linked together, enabling the pooling of compute resources. Oracle's grid infrastructure:

- Low cost
- High quality of service
- Easy to manage

These are the essential components of Oracle's grid-based systems:

• Real Application Clusters (RAC)
• Information sharing
• Easy server manageability
• The advisory framework
• Automatic performance tuning
• Automatic Storage Management (ASM)
• Automatic memory management
• Scheduling and resource management

The Oracle DBA's Role

The main responsibility of a DBA is to make corporate data available to the end users and the decision makers of an organization.

• *Security:* Ensuring that the data and access to the data are secure
• *Backup:* Ensuring that the database can be restored in the event of either human or systems failure
• *Performance:* Ensuring that the database and its subsystems are optimized for performance
• *Design:* Ensuring that the design of the database meets the needs of the organization
• *Implementation:* Ensuring proper implementation of new database systems and applications.

In a small organization a DBA could be managing the entire information technology (IT) infrastructure, including the databases, whereas in a large organization there could be a number of DBAs, each charged with managing a particular area of the system.

The DBA's Security Role

- Protecting the Database – The Oracle DBA is the person the information departments entrust with safeguarding the organization's data, and this involves preventing unauthorized use of and access to the database.
- Monitoring the System- The tasks involved in monitoring the system include the following:
  - Monitoring space in the database to ensure it is sufficient for the system
  - Checking to ensure that batch jobs are finishing as expected
  - Monitoring log files on a daily basis for evidence of unauthorized attempts to log in (something DBAs want to keep close tabs on)
- Creating and Managing Users - Every database has users, and it's the DBA's job to create them based on requests from the appropriate people.

The DBA's System Management Role

The following sections describe the various facets of the system management part of the Oracle DBA's job.

1. Troubleshooting
2. Ensuring Performance Tuning - all database tuning efforts can be grouped into two classes—proactive and reactive tuning.
3. Minimizing Downtime
4. Estimating Requirements – estimation of physical requirements, planning for future growth.
5. Developing Backup and Recovery Strategies
6. Loading Data
7. Overseeing Change Management

The DBA's Database Design Role

1. Designing the Database
2. Installing and Upgrading Software
3. Creating Databases
4. Creating Database Objects

DBA Job Classifications

A DBA's job description is not exactly the same in all organizations. There are several variations in the job's classification and duties across organizations.

***Production* DBA** refers to database administrators in charge of production databases. DBAs who are involved in the preproduction design and development of databases are usually called ***development* or *logical* DBAs.**

Types of Databases

Databases perform a variety of functions, but you can group all of those functions into two broad categories: online transaction processing (OLTP) and decision-support systems. Let's take a quick look at some of the basic classifications of Oracle databases.

1. **Online Transaction Processing and Decision-Support System Databases -** *Online transaction processing* (OLTP) databases includes order entry, billing, customer, supplier, and supply-chain databases. *Decision-support systems* (DSSs) range from small databases to large data warehouses. They can easily manage with regularly scheduled downtime and maintenance windows.

2. **Development, Test, and Production Databases-** *Development databases* are usually owned by the development team, which has full privileges to access and modify data and objects in those databases. The *test databases* are designed to simulate actual production databases and are used to test the functionality of code after it comes out of the development databases. No new code is usually implemented in the "real" *production databases* of the company unless it has been successfully tested in the test databases.

Oracle Database 10*g* Architecture

This topic explains about the fundamental structures of Oracle Database 10*g*. To understand how the Oracle database works, you need to understand several concepts, including transaction processing, backup and recovery, undo and redo data, the optimization of SQL queries, and the importance of the data dictionary.

Oracle Database Structures

In discussing the Oracle database architecture, you can make a distinction between the physical and logical structures.

The Logical Database Structures

Oracle databases use a set of logical database storage structures in order to manage the physical storage that is allocated in the form of operating system files. These logical structures, which primarily include tablespaces, segments, extents, and blocks, allow Oracle to control the use of the physical space allocated to the Oracle database.

***Data blocks*:** A data block consists of a number of bytes of disk space in the operating system's storage system. The smallest logical component of an Oracle database is the *data block.* Data blocks are defined in terms of bytes. For example, you can size an Oracle data block in units of 2KB, 4KB, 8KB, 16KB, or 32KB (or even larger chunks), and it is common to refer to the data blocks as *Oracle blocks.*

- ***Extents***: An *extent* is two or more contiguous Oracle data blocks, and this is the unit of space allocation.
- ***Segments*:** A *segment* is a set of extents that you allocate to a logical structure like a table or an index (or some other object).
- ***Tablespaces*:** A *tablespace* is a set of one or more data files, and usually consists of related segments.

Data Blocks Multiple Oracle Data Block Sizes

The DB_BLOCK_SIZE initialization parameter determines the standard block size in your Oracle database, and it can range from 2KB to 32KB. For example, you can have 2KB, 4KB, 8KB, 16KB, and 32KB block sizes all within the same database. Multiple block sizes are useful primarily when transporting tablespaces between databases with different database block sizes.

What's Inside a Data Block?

All data blocks can be divided into two main parts: the row data portion and the free space portion. The *row data* section of data blocks contains the data stored in the tables or their indexes. The *free space* section is the space left in the Oracle block for new data to be inserted or for existing rows in the block to be extended

The following five tablespaces are generally the default tablespaces that all databases must have,

- System tablespace
- Sysaux tablespace
- Undo tablespace
- Temporary tablespace
- Default permanent tablespace

The customization of the block size for a tablespace provides several benefits: *Optimal disk I/O*, *Optimal caching of data*, *Easier transport of tablespaces.*

Temporary Tablespaces

Users need a temporary location to perform certain activities, such as sorting, and if you don't provide a designated *temporary tablespace* for them, they end up using the System tablespace

Dictionary-Managed vs. Locally Managed Tablespaces

In the case of dictionary-managed tablespaces, every time a table or other object needs to grow, Oracle checks its data dictionary to ensure that there's free disk space to allocate to the object, and then updates its free-space information after allocating a new extent to the object. Locally managed tablespaces keep the space-management information in the data files themselves, and the tablespaces automatically track the free or used status of blocks in each data file.

Commonly Used Tablespaces

- Bigfile tablespaces are tablespaces with a single large data file, whose size can range from 8 to 128 terabytes, depending on the database block size.
- Smallfile tablespaces can contain multiple data files, but the files cannot be as large as a bigfile data file.
- Temporary tablespaces contain data that persists only for the duration of a user's session. Usually Oracle uses these tablespaces for sorting and similar activities for users.
- Permanent tablespaces include all the tablespaces that aren't designated as temporary tablespaces.
- Undo tablespaces contain undo records, which Oracle uses to roll back, or undo, changes to the database.
- Read-only tablespaces don't allow write operations on the data files in the tablespace.

Physical Database Structures

The Oracle database consists of the following three main types of files:

• *Data files*: These files store the table and index data. Oracle data files constitute most of a database's total space. When the database instance needs to read table or index data, it reads that from the data files on disk, unless that data is already cached in Oracle's memory
• *Control files*: These files record changes to all database structures. The control file contains the names and locations of the data files, redo log files, current log sequence numbers, backup set details, and the all-important *system change number* (SCN), which indicates the most recent version of committed changes in the database.
• *Redo log files*: These online files contain the changes made to table data. The set of redo log files that are currently being used to record the changes to the database are called *online redo log files*. These logs can be archived or copied to a different location before being reused, and the saved logs are called *archived redo logs*. Oracle writes all final changes made to data (committed data) first to the redo log files. Redo log files consist of *redo records,* which are groups of *change vectors,* each referring to a specific change made to a data block in the Oracle database.

The SPFILE

In the SPFILE, you specify the memory limits for the instance, the locations of the control files, whether and where the archived logs are saved, and other settings that determine the behavior of the Oracle database server. By default, the SPFILE (and the init.ora file) is placed in the ORACLE_HOME/dbs directory in UNIX systems and the ORACLE_HOME\database directory in Windows systems

The Password File

The *password* file is an optional file in which you can specify the names of database users who have been granted the special SYSDBA or SYSOPER administrative privileges

The Alert Log File

Every Oracle database has an *alert log* named alert*db_name*.log. The alert log captures major changes and events that occur during the running of the Oracle instance, including log switches, any Oracle-related errors, warnings, and other messages. Oracle puts the alert log in the location specified for the BACKGROUND_DUMP_DEST initialization
parameter. To find out where the alert log is located, issue the following command:

```
SQL> SHOW PARAMETER background dump
NAME          TYPE                VALUE
------------------------------------------
background_core_dump string partial
background_dump_dest string /u01/app/oracle/product/10.2.0/db_1/orcl/bdump
```

Trace Files

Oracle requires that you specify three different trace file directories in your initialization file: the background dump directory, the core dump directory, and the user dump directory

Oracle Managed Files

The OMF feature aims at relieving DBAs of their traditional file-management tasks. When you use the OMF feature, you don't have to worry about the names and locations of the physical files. The OMF-based files are ideal for test and small databases, but if you have a terabyte-sized database with a large number of archived logs and redo logs, you need flexibility, which the OMF file system can't provide.

Oracle Processes

A *process* is essentially a connection or thread to the operating system that performs a task or job. Oracle processes are divided into two general types both for efficiency and to keep client processes separate from the database server's tasks:

User processes: These processes are responsible for running the application that connects the user to the database instance.

Oracle processes: These processes perform the Oracle server's tasks, and you can divide them into two major categories: *server processes* and *background processes.* Together, these processes perform all the actual work of the database, from managing connections to writing to logs and data files to monitoring the user processes. The two types of Oracle processes—the server processes and the background processes.

The Server Process

The *server process* is the process that services an individual user process. Each user connected to the database has a separate server process created for the duration of the session. The server

process is created to service the user's process and is used by the user process to communicate with the Oracle database server. The most common configuration for the server process is to assign each user a *dedicated* server process. You can also configure shared server *connection pooling*. Connection pooling lets you reuse existing timed-out connections to service other active sessions.

The Background Processes

The *background processes* are the real workhorses of the Oracle instance—they enable large numbers of users to concurrently and efficiently use information stored in database files. Each of the Oracle background processes is in charge of a separate task, thus increasing the efficiency of the database instance. These processes are automatically created by Oracle when you start the database instance, and they terminate when the database is shut down.

Background Process	Function
Database writer	Writes modified data from the buffer cache to disk (data files)
Log writer	Writes redo log buffer contents to the online redo log files
Checkpoint	Updates the headers of all data files to record the checkpoint details
Process monitor	Cleans up after finished and failed processes
System monitor	Performs crash recovery and coalesces extents
Archiver	Archives filled online redo log files
Manageability Monitor	Performs database-manageability-related tasks
Manageability Monitor Light	Performs tasks like capturing session history and metrics
Memory manager	Coordinates the sizing of the SGA components
Job queue coordination process	Coordinates job queues to expedite job processes
Database writer	Writes modified data from the buffer cache to disk (data files)
Log writer	Writes redo log buffer contents to the online redo log files

Oracle Memory Structures

Oracle uses two kinds of memory structures, one shared and the other process-specific. The *system global area* (SGA) is the part of total memory that all server processes (including background processes) share. The process-specific part of the memory is known as the *program global area*(PGA), or *process-private memory*..

What is the SGA

The system global area (SGA) is just shared memory structures that are created at instance startup, hold information about the instance and control its behavior. The following table gives a brief synopsis of the particular components of the SGA, the variables that control the size of memory allocated, some of the areas of the Oracle server the particular component has an influence on, and then a very brief description.

Components of the SGA

SGA COMPONENT	SIZE CONTROLED BY	AREAS OF INFLUENCE	SIMPLE DESCRIPTONS
Shared Pool	SHARED_POOL_SIZE	Library Cache • Shared SQL areas • Private SQL areas • PL/SQL procedures and packages • Various control structures	Oracle needs to allocate & deallocate memory as SQL or procedural code is executed based on the individual needs of users' sessions and in accordance to the LRU algorithm.
		Dictionary Cache • Row cache • Library cache	Highly accessed memory structures that provide information on object structures to SQL statements being parsed.
Java Pool	JAVA_POOL_SIZE	• Run state • Methods • Classes • Session code • Data in JVM	Memory available for the Java memory manager to use for all things Java.
Streams Pool	STREAMS_POOL_SIZE	• Stream activity	New to Oracle 10g, memory available for stream processing.
Redo Log Buffer	LOG_BUFFER	• Redo entries	Holds changes made to data and allows for the reconstruction of data in the case of failure.
Database Buffer Cache	DB_2K_CACHE_SIZE DB_4K_CACHE_SIZE DB_8K_CACHE_SIZE DB_16K_CACHE_SIZE DB_32K_CACHE_SIZE DB_KEEP_CACHE_SIZE DB_RECYCLE_CACHE_SIZE	• Write list • LRU list	Holds copies of data requested by SQL and reduces requests to disk by having data in memory. You may have many different buffer caches that help segregate on usage patterns.
Large Pool	LARGE_POOL_SIZE	• Shared server • Oracle XA • I/O server processes • Backup & restore	For large memory allocations.

The Program Global Area (PGA)

A **program global area (PGA)** is a memory region that contains data and control information for a server process. It is a nonshared memory created by Oracle when a server process is started.

You can classify the PGA memory into the following types:

• Private SQL area: This area of memory holds SQL variable bind information and runtime memory structures. Each session that executes a SQL statement will have its own private SQL area.

• Runtime area: The runtime area is created for a user session when the session issues a SELECT, INSERT, UPDATE, or DELETE statement. After an INSERT, DELETE, or UPDATE statement is run, or after the output of a SELECT statement is fetched, the runtime area is freed by Oracle.

The size and content of the PGA depends on the Oracle-server options installed. This area consists of the following components:

- stack-space: the memory that holds the session's variables, arrays, and so on.
- session-information: unless using the multithreaded server, the instance stores its session-information in the PGA. (In a multithreaded server, the session-information goes in the SGA.)
- private SQL-area: an area in the PGA which holds information such as bind-variables and runtime-buffers.
- sorting area: an area in the PGA which holds information on sorts, hash-joins, etc.

The sum of all the PGA memory used by all sessions makes up the PGA used by the instance. Oracle recommends that you use *automatic PGA management*, which automates the allocation of PGA memory.

Database Transaction

A transaction is a logical unit of work in an Oracle database, and consists of one or more SQL statements. A transaction begins with the first executable SQL statement and terminates when you commit or roll back the transaction. Committing a transaction will make your changes permanent, and rolling back the changes will, of course, undo them. Once you commit the transaction, all other users' transactions that start subsequently will be able to see the changes made by your transactions.

Creating an Oracle Database

The Database Configuration Assistant (DBCA) an Oracle supplied tool that enables you to create an Oracle database, configure database options for an existing Oracle database, delete an Oracle database, or manage database templates. DBCA is launched automatically by the Oracle Universal Installer, but it can be invoked standalone from the Windows operating system start menu (under Configuration Assistants) or by entering the following on the UNIX command line:
dbca

Creating the File System for the Database

Planning your file systems is an important task, and you need to complete it before you start creating the database. You have to plan the location of the various database files, such as the redo log files and archive log files before you create the database.

Sizing the File System

The amount of file system space you need depends primarily upon the total space you need to allocate for Oracle data files. Your overall space estimate should include space for the following:

- *Tables and indexes*
- *Undo tablespace*
- *Temporary tablespace*
- *Default permanent tablespace*
- *System and Sysaux tablespaces*
- *Redo log files*:
- *Flash recovery area*

Ensuring Enough Memory Is Allocated

Setting the OS Environment Variables

In UNIX and Linux environments, you must set the following environment variables:

- ORACLE_SID: This is your database's name and same as the value of the DB_NAME initialization parameter.
- ORACLE_BASE: This is the top directory for the Oracle software. For this chapter's purposes, this is /u01/app/oracle.
- ORACLE_HOME: This is the directory in which you installed the Oracle software. Oracle recommends you use the following format for this variable: *$ORACLE_BASE/product/release/db_n*.
- PATH: This is the directory in which Oracle's executable files are located. Oracle's executables are always located in the $ORACLE_HOME/bin directory. You can add the location of Oracle's executable files to the existing PATH value in the following way:
- export PATH=$PATH:$ORACLE_HOME/bin LD_LIBRARY_PATH: This variable points out where the Oracle libraries are located. The usual location is the $ORACLE_HOME/lib directory.

Creating the Parameter File

Before you jump into the details of Oracle database creation, it's important to familiarize yourself with the important Oracle initialization parameters and how Oracle uses them.

Types of Database Parameter Files

Oracle uses a *parameter file* to store the initialization parameters and their settings for an instance. You can use either of the following two types of parameter files:

- *Server parameter file (SPFILE)*: A binary file that contains the initialization parameters
- *Initialization parameter file (pfile)*: A text file that contains a list of all initialization parameters

The key difference between these two types of files is that with an SPFILE, you have the option of making any changes you make to the initialization parameters while an instance is running persist across an instance shutdown.

The Initialization Parameter File

The interesting thing about the initialization parameter file is that it contains the configuration parameters for memory and some I/O parameters, but not for the database filenames or the tablespaces that the data files belong to.

Changing the Initialization Parameter Values

You have three ways to change the value of dynamic parameters: the ALTER SESSION, ALTER SYSTEM, and ALTER SYSTEM . . . DEFERRED commands.

Using the ALTER SESSION Command

The ALTER SESSION command enables you to change dynamic parameter values for the duration of the session that issues the command. The ALTER SESSION command is used only to change a parameter's value temporarily. Here is the general syntax for the command:
ALTER SESSION SET *parameter_name=value;*

Using the ALTER SYSTEM Statement

The ALTER SYSTEM statement changes the parameter's value for all sessions. However, these changes will be in force only for the duration of the instance; when the database is restarted, these changes will go away, unless you modify the init.ora file accordingly or you use the SPFILE. Here is the syntax for this command:
ALTER SYSTEM SET *parameter_name=value;*

Using the ALTER SYSTEM . . . DEFERRED Statement

The ALTER SYSTEM . . . DEFERRED statement will make the new values for a parameter effective for all sessions, but not immediately. Only sessions started after the statement is executed are affected. All currently open sessions will continue to use the old parameter values. Here is the syntax for this command:
ALTER SYSTEM SET *parameter_name* DEFERRED;

Important Oracle Database 10*g* Initialization Parameters

Audit-Related Parameters

AUDIT_TRAIL

The parameter is set as follows:
AUDIT_TRAIL = db
db: Oracle records the same type of auditing as with the os setting, but it directs all audit records to the database audit trail, which is the AUD$ table owned by SYS.

AUDIT_FILE_DEST

The AUDIT_FILE_DEST parameter specifies the directory in which the database will write the audit records, when you choose the operating system as the destination with the AUDIT_TRAIL parameter by specifying AUDIT_TRAIL=os.

AUDIT_SYS_OPERATIONS

This parameter, if set to a value of true, will audit all actions of the SYS user and any other user with a SYSDBA or SYSOPER role and will write the details to the operating system audit trail specified by the AUDIT_TRAIL parameter

Database Name and Other General Parameters

DB_NAME and DB_UNIQUE_NAME

The DB_NAME parameter sets the name of the database. This is a mandatory parameter and the value is the same as the database name you used to create the database. The DB_NAME value should be the same as the value of the ORACLE_SID environment variable.

DB_DOMAIN

The DB_DOMAIN parameter specifies a fully qualified name (in Internet dot notation) for the database, and this is typically the same as the name of the organization that owns the database. The DB_DOMAIN parameter specifies the logical location of the database within the network structure, and you should set this parameter if your database is part of a distributed system.

INSTANCE_NAME

The INSTANCE_NAME parameter will have the same value as the DB_NAME parameter in a single instance environment. You can associate multiple instances to a single database service (DB_NAME) in a Real Application Clusters environment.

File-Related Parameters

IFILE

You can use the IFILE parameter to embed another initialization file in it. For example, you can have a line in your init.ora file such as this:

ifile=config.ora

CONTROL_FILES

Control files are key files that hold data file names and locations and a lot of other important information.

CONTROL_FILE_RECORD_KEEP_TIME

The CONTROL_FILE_RECORD_KEEP_TIME parameter specifies how many days Oracle will retain records in the control file before overwriting them.

Viewing the Current Initialization Parameter Values

You can always use a file editor such as Windows Notepad to examine init.ora files, not only to view the settings for initialization parameters, but also (at your own risk) to change their values. However, there is a major drawback to doing this: you cannot see the default values of all the initialization parameters.

The V$PARAMETER View

You can run the following query to find out the values of all the parameters:

SQL> SELECT name, value, isdefault FROM v$parameter;

The isdefault column has a value of true if the parameter is the default value and yes if you actually set it to something other than the default value.

The SHOW PARAMETER Command

You can just type SHOW PARAMETER and you'll see all the initialization parameters with their values. For example, the keywords LOCKS, FILES, LOG, and many others can be passed along to the SHOW PARAMETER command to get the values of a related set of parameters.

Listing 1 shows an example of the use of the SHOW PARAMETER command. Here, the output shows all initialization parameters that contain the string "lock".

Listing 1 . ***Using the SHOW PARAMETER Command***

```
SQL> SHOW PARAMETER LOCK
NAME                                 TYPE                 VALUE
------------                         -----------------    ------- --------
db_block_buffers                      integer              0
db_block_checking                    boolean              FALSE
db_block_checksum                    boolean              TRUE
db_block_size                        integer              8192
db_file_multiblock_read_count        integer              8
ddl_wait_for_locks                   boolean              FALSE
distributed_lock_timeout             integer              60
```

Creating a New Database

In this section, I show you how to create a new database from scratch, using individual database creation statements.

Setting OS Variables

First, make sure ORACLE_HOME is set for the session you log into. The ORACLE_HOME environment variable in Oracle Database 10*g* databases is in the following format:
$ORACLE_BASE/product/10.2.0/db_1

You can thus set your ORACLE_HOME as in the following example:
$ export ORACLE_HOME=/u01/app/oracle/product/10.2.0/db_1

Ensuring You Have the Privileges to Create Databases

You can connect to the database as the super user SYS with the SYSDBA privilege, as shown here:
SQL> CONNECT sys AS sysdba

Creating the Database

The simplest database you can create will have a System tablespace to hold the data dictionary, a Sysaux tablespace, a pair of control files and redo log files, a default temporary tablespace, and an undo tablespace. Once you have this database going, you can add any number of new tablespaces to it. You can create your new database as shown in Listing 2

Listing 2. *The CREATE DATABASE Script*

```
SQL> create database nina
2 user sys identified by sys_password
3 user system identified by system_password
4 maxinstances 1
5 maxloghistory 1
6 maxlogfiles 5
7 maxlogmembers 5
8 character set US7ASCII
9 national character set AL16UTF16
10 datafile '/u02/app/oracle/oradata/nina/system01.dbf' size 500M
11 extent management LOCAL
12 SYSAUX datafile '/u02/app/oracle/oradata/nina/sysaux01.dbf' size 500M
13 DEFAULT TEMPORARY tablespace temp01
14 tempfile '/u02/app/oracle/oradata/nina/temp01_01.dbf' size 100M
15 UNDO tablespace undotbs_01
16 datafile '/u02/app/oracle/oradata/nina/undotbs01.dbf' size 200M
17 DEFAULT tablespace users
18 datafile '/u02/app/oracle/oradata/nina/users01.dbf' size 100M
19 LOGFILE group 1
20 ('/u02/app/oracle/oradata/nina/redo01.log') size 100M,
21 group 2
22 *('/u01/app/oracle/oradata/nina/redo02.log') size 100M;
Database created.
SQL>
```

A Simple Way to Create a Database

If you want to create a new Oracle database in a hurry, you can do so by following these steps:

1. Create a new init.ora file with just one parameter, DB_NAME.

2. Start up your new instance as follows:

```
SQL> STARTUP NOMOUNT
ORACLE instance started.
Total System Global Area 188743680 bytes
Fixed Size 1308048 bytes
Variable Size 116132464 bytes
Database Buffers 67108864 bytes
Redo Buffers 4194304 bytes
SQL>
```

3. Create your new database with the following simple statement:

```
SQL> CREATE DATABASE;
Database created.
SQL>
```

Creating Additional Tablespaces

Once you've decided on the tablespaces you need, use commands like the following to create the additional tablespaces (by default, Oracle will create a locally managed tablespace):

```
SQL> CREATE TABLESPACE sales01
DATAFILE '/u02/app/oracle/oradata/nina/sales01_01.dbf' size 500M
Tablespace created.
SQL>

SQL> CREATE TABLESPACE sales02
DATAFILE '/u02/app/oracle/oradata/nina/sales02_01.dbf' size 500M
Tablespace created.
SQL>
```

Changing the Passwords for the Default Users

For each of the default users, you must modify the default passwords, as shown in the following examples:

```
SQL> ALTER USER outln IDENTIFIED BY 'new_password';
SQL> ALTER USER dbsnmp IDENTIFIED BY 'new_password';
```

Changing the Archive Logging Mode

Oracle won't archive or save the redo logs it fills up. Instead, it overwrites them when it needs to write to a new log file. In archivelog mode, Oracle ensures that it first saves the filled-up redo log file before permitting it to be overwritten. Before you change anything, you should confirm the archivelog mode of the database. Here is one way of doing so:

```
SQL> SELECT log_mode FROM v$database;
LOG_MODE
------------------------
NOARCHIVELOG
1 row selected.
```

Here's the database shutdown command:

```
SQL> SHUTDOWN IMMEDIATE
```

Database closed.
Database dismounted.
ORACLE instance shut down.

Using a Server Parameter File (SPFILE)

A server parameter file is basically a repository for initialization parameters. Initialization parameters stored in an SPFILE are persistent, meaning any parameter changes made while an instance is running can persist across instance shutdown and startup. A server parameter file is initially built from the traditional text initialization parameter file, using the create SPFILE statement. It is a binary file that cannot be browsed or edited with a text editor.

Creating a Server Parameter File- The server parameter file is initially created from a text initialization parameter file (init.ora). It must be created prior to its use in the STARTUP command. The create SPFILE statement is used to create a server parameter file. The following example creates a server parameter file from an initialization parameter file:

```
CREATE SPFILE FROM PFILE='/u01/oracle/product/920/dbs/initRAC1.ora';
```

Below is another example that illustrates creating a server parameter file and supplying a name:

```
CREATE SPFILE='/u01/oracle/product/920/dbs/racdb_spfile.ora'
FROM PFILE='/u01/oracle/product/920/dbs/init.ora';
```

The following example creates a text initialization parameter file from the server parameter file. Since no paths are specified, the files will be located in $ORACLE_HOME/database or its equivalent:

```
CREATE PFILE FROM SPFILE;
```

The example below creates a text initialization parameter file from a server parameter file, where the names of the files are specified:

```
CREATE PFILE='/u01/oracle/product/920/dbs/racdb_init.ora'
FROM SPFILE='/u01/oracle/product/dbs/racdb_spfile.ora';
```

Setting the Scope of Dynamic Parameter Changes

Once you create an SPFILE, you can use a special SCOPE clause as part of all your ALTER SYSTEM commands that will determine whether the changes persist or not. The SCOPE clause can take the following three values:

- SPFILE
- MEMORY
- BOTH

When the SCOPE clause is set to MEMORY, changes are merely temporary and they go away after the database is restarted. When the SCOPE clause is set to BOTH, all dynamic changes get recorded in the SPFILE and are operational in the instance immediately. When the SCOPE clause is set to SPFILE, changes aren't applied immediately but only get recorded in the SPFILE; dynamic and static configuration parameters become effective only after the next startup of the database.

Here are some examples:

```
SQL> ALTER SYSTEM SET
log_archive_dest_2='location=/test02/app/oracle/oradata/arch'
SCOPE=SPFILE;

SQL> ALTER SYSTEM SET log_checkpoint_interval=600
SCOPE=MEMORY;
```

POSSIBLE QUESTIONS:

PART-B:

1. Elucidate the role of DBA and their responsibilities in managing a database.
2. Explain the Logical Database Structures of Oracle 10g
3. Explain the physical Database Structures of Oracle 10g
4. What are the different types of processes in Oracle? Explain.
5. List out the components of SGA and explain their usage.
6. What is a PGA? Explain its classification.
7. List out the types of parameter files in Oracle and explain their use.
8. Explain the sequence of actions needed to create a database in Oracle.
9. Describe the steps involved in creating and using a Server Parameter File.
10. Elucidate the different types of databases.

Karpagam Academy of Higher Education
Department of CS, CA & IT
Subject: Oracle 10g Administration (17CSP203)

Batch : 2017-2019 **Class: I M.Sc CS**

Objective Type Questions

UNIT I

S.NO	QUESTIONS	OPTION 1	OPTION 2	OPTION 3	OPTION 4	KEY
1	DBAs who are involved in the preproduction design and development of database is known as _____.	production	development	implement	excution	development
2	Decision support sytem is also known as _____.	OLTP	OLAP	OLDP	OLTM	OLAP
3	_____database are characterized by heavy transaction volume.	OLTP	OLAP	OLDP	OLTM	OLTP
4	_____ range from small database to large database.	OLTP	DAM	DSS	OLTM	DSS
5	The _____ are designed to simulate actual production databases and are used to test the functionality of code.	production database	test databases	implement database	execution database	test databases
6	An _____ consists of files, both data files and Oracle system files.	production database	test databases	Oracle database	execution database	Oracle database
7	A _____consists of a number of bytes of disk space in the operating system's storage system.	extends	data block	segments	tablespaces	data block
8	An _____ is two or more contiguous Oracle data blocks.	extends	data block	segments	tablespaces	extends
9	A _____ is a set of extents that you allocate to a logical structure like a table or an index.	extends	data block	segments	tablespaces	segments
10	A _____ is a set of one or more data files, and usually consists of related segments.	extends	data block	segments	tablespaces	tablespaces
11	The smallest logical component of an Oracle database is the _____.	extends	data block	segments	tablespaces	data block
12	The _____ of data blocks contains the data stored in the tables or their indexes.	free space section	row data section	database section	segment section	free space section

13	The _____ is a purely logical construct and is the primary logical storage structure of an Oracle database.	extends	data block	segments	tablespaces	tablespaces
14	_____ tablespaces are the default in Oracle Database 10g.	locally managed	dictionary managed	database managed	uniform managed	locally managed
15	_____ tablespaces are tablespaces with a single large data file.	Bigfile	smallfile	temporary	permanent	Bigfile
16	_____tablespaces can contain multiple data files.	Bigfile	smallfile	temporary	permanent	smallfile
17	_____ files store the table and index data.	Bigfile	Data files	temporary	permanent	Data files
18	_____files record changes to all database structures.	Bigfile	Data files	Control files	permanent	Control files
19	_____online files contain the changes made to table	Bigfile	Data files	Control files	Redo log files	Redo log files
20	The _____ is an optional file in which you can specify the names of database users who have been granted the special SYSDBA or SYSOPER administrative privileges.	Bigfile	Data files	password file	Redo log files	password file
21	_____ are responsible for running the application that connects the user to the database instance.	oracle processes	server processes	User processes	database processes	User processes
22	The ____ process is the process that services an individual user process.	server	oracle	background	database	server
23	_____ process Cleans up after finished and failed processes	system monitor	database writer	Process monitor	memory manager	Process monitor
24	The job of the _____ process is to transfer the contents of the redo log buffer to disk.	system monitor	database writer	Process monitor	log writer	log writer
25	The_____ process performs system-monitoring tasks for the Oracle instance.	database writer	Process monitor	system monitor	log writer	system monitor
26	The _____ process collects several types of statistics to help the database manage itself.	database writer	manageability monitor	system monitor	log writer	manageability monitor
27	The _____ purpose is to speed up query performance and to enable a high amount of concurrent database	SGA	PGA	MMA	MMON	SGA
28	_____ are data buffers that are currently in active use by user sessions.	shared buffer	Pinned buffers	dirty buffers	free buffers	Pinned buffers

29	The _____ process coordinates the sizing of the memory components.	SGA	MMAN	PGA	MMON	MMAN
30	The _____ process coordinates disk rebalancing activity when you use an automatic storage management storage system.	rebalance master	ASM rebalance	MMON	ASM background	rebalance master
31	The _____ parameter turns auditing on or off for the database.	AUDIT_SET	AUDIT_FILE	AUDIT_TRAIL	AUDIT_DEST	AUDIT_TRAIL
32	The _____ parameter sets the name of the database.	DB_DOMAIN	DB_INSTANCE	DB_NAME	DB_UNIQUE_NAME	DB_NAME
33	The _____ parameter allows you to use the latest oracle database release.	SERVICE_NAME	INSTANCE_NAME	INSTANCE_TYPE	COMPATIBLE	COMPATIBLE
34	You can use the _____ parameter to embed another initialization file in it.	IFILE	CONTROL_FILES	DB_FILES	USER_FILE	IFILE
35	The _____ parameter specifies the maximum number of database files that can be opened for a database.	IFILE	CONTROL_FILES	DB_FILES	USER_FILE	DB_FILES
36	The _____ parameter specifies the size of the redo log buffer.	LOG_POOL_SIZE	LARGE_POOL_SIZE	LOG_BUFFER	BUFFER_SIZE	LOG_BUFFER
37	_____ parameter specifies the default filename format for the archieved redo log files.	LOG_ARCHIVE_FORMAT	LOG_FORMAT	LOG_ARCHIVE	LOG_BUFFER	LOG_ARCHIVE_FORMAT
38	_____ parameter specifies the maximum number of users you can create in your database	LICENSE_USERS	LICENSE_DB	LICENSE_MAX_USERS	LICENSE_LEVEL	LICENSE_MAX_USERS
39	The file is called _____ file because it is always maintained on the machine where the oracle database server is located.	oracle	server	init	orcl	server
40	Which of the following is a package?	put_line	serveroutput	dbms_output	dbms_input	dbms_output

41	You can create a new init.ora file from the SPFILE in the default location by using the following command:	create init.ora from spfile	create pfile from spfile;	create init from spfile	create pfile from spfile;	create pfile from spfile;
42	_____ is the command to suspend the database.	alter system quiesce restricted	alter database quiesce restricted	alter database suspend;	Alter system suspend	Alter system suspend
43	____ is the continuation character in SQL.	hypen	comma	semicolon	dot	hypen
44	_____ commands are executed locally and are not sent to the server.	server executed	SQL*Plus	Local	Create table	Local
45	____ is used to see the cureently logged in user name.	show user	show sga	show recyclebin	show errors	show user
46	____ parameter specifies both the username and the password of the user in the database .	userid	userpassword	userpw	userlogin	userid
47	_____ are buffers that do not contain any useful data.	shared buffer	Pinned buffers	dirty buffers	free buffers	free buffers
48	_____ contain data that was read from disk and then modified, but hasn't yet been written to the data files on	shared buffer	Pinned buffers	dirty buffers	free buffers	dirty buffers
49	_____ keeps the state of java program execution.	shared pool	redo log buffer	java pool	large pool	java pool
50	____ stores large memory allocations, such as RMAN backup buffers.	shared pool	redo log buffer	java pool	large pool	large pool
51	_____ holds copies of data blocks read from data files.	shared pool	database buffer cache	java pool	large pool	database buffer cache
52	_____ file is an alternative to the init.ora file.	SPFILE	PFILE	IFILE	TFILE	SPFILE
53	_____parameter provides a name for the database service.	DATABASE_ SERVICE	SERVICE_NA ME	DATA_SERV ICE	SERVICE_U SERS	SERVICE_NAME

UNIT-II

SYLLABUS

Schema management: creating and managing table spaces – indexes - materialized views. Oracle transaction management: oracle transactions - transaction properties – transaction concurrency control - isolation levels - implementing oracle's concurrency control – using undo data to provide read consistency - transaction query - discrete transactions –autonomous transactions - resumable space allocation.

Schema management

In Oracle, a *schema* is defined as a collection of logical structures of data, or schema objects, although it is used mostly as a synonym for the database user (specifically, the application owner) that owns the schema pertaining to a specific application. Although the DBA can use the CREATE SCHEMA statement to create a specific schema, more often the application owner creates the database objects and is referred to as the *schema owner*.

Creating and Managing Tablespaces

Tablespaces are logical entities—each of an application's tables and indexes are stored as a segment, and the segments are stored in the data files that are parts of tablespaces. A tablespace is thus a logical allocation of space for Oracle schema objects. The following are the important types of Oracle tablespaces:

• Temporary tablespaces are used to store objects for the duration of a user's session only. You use temp files to create a temporary tablespace, instead of data files.

• Undo tablespaces are a type of permanent tablespace that are used to store undo data, which is used to undo changes to data.

You can create two basic types of tablespaces in an Oracle database, which differ by how they manage the database extents: *locally managed* and *dictionary-managed* tablespaces.

Locally and Dictionary-Managed Tablespaces

The basic unit of space allocation in Oracle databases, and dictionary managed tablespaces store extent information in the data dictionary. Locally managed tablespaces, on the other hand, manage extents by referring to the bitmaps kept in each physical data file header for all the blocks within that data. Locally managed tablespaces have several advantages over the traditional dictionary-managed tablespaces. Dictionary-managed tablespaces have to constantly check the data dictionary during the course of extent management—whenever an extent is allocated to an object or reclaimed from an object, Oracle will update the relevant tables in the data dictionary.

Allocating the Extent Size: Autoallocate vs.Uniform

You create a locally managed tablespace by specifying LOCAL in the EXTENT MANAGEMENT clause of the CREATE TABLESPACE statement. This is the default for new permanent tablespaces, but you must specify if it you want to specify the management of the locally managed tablespace. You can have the database manage extents for you automatically with the AUTOALLOCATE clause (the default), or you can specify that the tablespace is managed with uniform extents of a specific size (UNIFORM). The following statement creates a locally managed tablespace named lmtbsb and specifies AUTOALLOCATE:

```
CREATE TABLESPACE lmtbsb DATAFILE '/u02/oracle/data/lmtbsb01.dbf' SIZE 50M
```

EXTENT MANAGEMENT LOCAL AUTOALLOCATE;

AUTOALLOCATE causes the tablespace to be system managed with a minimum extent size of 64K. In contrast, dictionary-managed tablespaces have a minimum extent size of two database blocks. Therefore, in systems with block size smaller than 32K, autoallocated locally managed tablespace will be larger initially.

The alternative to AUTOALLOCATE is UNIFORM which specifies that the tablespace is managed with extents of uniform size. You can specify that size in the SIZE clause of UNIFORM. If you omit SIZE, then the default size is 1M.

The following example creates a tablespace with uniform 128K extents. (In a database with 2K blocks, each extent would be equivalent to 64 database blocks). Each 128K extent is represented by a bit in the extent bitmap for this file.

```
CREATE TABLESPACE lmtbsb DATAFILE '/u02/oracle/data/lmtbsb01.dbf' SIZE 50M
  EXTENT MANAGEMENT LOCAL UNIFORM SIZE 128K;
```

You cannot specify the DEFAULT storage clause, MINIMUM EXTENT, or TEMPORARY when you explicitly specify EXTENT MANAGEMENT LOCAL. If you want to create a temporary locally managed tablespace, use the CREATE TEMPORARY TABLESPACE statement.

Automatic vs. Manual Segment Space Management

When you create a locally managed tablespace using the CREATE TABLESPACE statement, the SEGMENT SPACE MANAGEMENT clause lets you specify how free and used space within a segment is to be managed. You can choose either manual or automatic segment-space management.

- MANUAL: Manual segment-space management uses free lists to manage free space within segments. Free lists are lists of data blocks that have space available for inserting rows. With this form of segment-space management, you must specify and tune the PCTUSED, FREELISTS, and FREELIST GROUPS storage parameters for schema objects created in the tablespace. MANUAL is the default.
- AUTO: Automatic segment-space management uses bitmaps to manage the free space within segments. The bitmap describes the status of each data block within a segment with respect to the amount of space in the block available for inserting rows. As more or less space becomes available in a data block, its new state is reflected in the bitmap. These bitmaps allow the database to manage free space automatically.

Automatic segment-space management delivers better space utilization than manual segment-space management. The following statement creates tablespace lmtbsb with automatic segment-space management:

```
CREATE TABLESPACE lmtbsb
DATAFILE '/u02/oracle/data/lmtbsb01.dbf' SIZE 50M
EXTENT MANAGEMENT LOCAL
SEGMENT SPACE MANAGEMENT AUTO;
```

Creating Tablespaces

Oracle strongly recommends the use of locally managed tablespaces and will eventually stop supporting dictionary-managed tablespaces. In Oracle Database 10*g*, locally managed tablespaces are the default for new permanent tablespaces.

Data Files and Tablespaces

A tablespace can have one or more data files, and a data file can belong to only one tablespace. Oracle creates a data file for a tablespace by specifying the keyword DATAFILE during tablespace creation. As a segment grows in size, Oracle allocates extents to it from the free space in its data files. When the tablespace starts to fill up, you can either add new data files to it or extend the size of the existing data files by using the RESIZE commandFor a temporary tablespace, you must use the clause TEMPFILE instead.

```
SQL> CREATE TABLESPACE test01
2 DATAFILE '/pasx02/oradata/pasx/test01.dbf'
3* SIZE 500M;
Tablespace created.
```

Now, let's execute the following query to determine the Oracle Database 10*g* Release 2 defaults for extent management, extent allocation type, and segment space management:

```
SQL> SELECT extent_management,
2 allocation_type,
3 segment_space_management
4 FROM dba_tablespaces
5* WHERE tablespace_name='TEST01';
EXTENT_MAN ALLOCATIO SEGMEN
---------- --------- ------
LOCAL SYSTEM AUTO
```

Note the defaults in Oracle Database 10*g* Release 2 carefully:

• Extent management: LOCAL

• Allocation of extent sizes: AUTOALLOCATE (shows up as SYSTEM in the preceding output)

• Segment space management: AUTO

Extent Allocation and Deallocation

Each Oracle data block corresponds to a specific number of bytes of disk space. Each of your database tables and indexes is called a *segment*, which is a set of extents allocated for a specific data structure Once Oracle allocates space to a segment by allocating a certain number of extents to it, that space will remain with the extent unless you make an effort to deallocate it. If you truncate a table with the DROP STORAGE option (TRUNCATE TABLE *table_name* DROP STORAGE), for example, Oracle deallocates the allocated extents. You can also manually deallocate unused extents using the following command:

```
SQL> ALTER TABLE table_name DEALLOCATE UNUSED;
```

When Oracle frees extents, it automatically modifies the bitmap in the data file where the extents are located, to indicate that they are free and available again.

Removing Tablespaces

You can drop a tablespace and its contents (the segments contained in the tablespace) from the database if the tablespace and its contents are no longer required. The following statement drops the users tablespace, including the segments in the tablespace:

```
DROP TABLESPACE users INCLUDING CONTENTS;
```

To delete the data files associated with a tablespace at the same time that the tablespace is dropped, use the INCLUDING CONTENTS AND DATAFILES clause. The following statement drops the user's tablespace and its associated datafiles:

```
DROP TABLESPACE users INCLUDING CONTENTS AND DATAFILES;
```

Adding Space to a Tablespace

When your tablespace is filling up with table and index data, you need to expand its size. You do this by adding more physical file space with the ALTER TABLESPACE command:

```
SQL> ALTER TABLESPACE test01
ADD DATAFILE '/finance10/app/oracle/finance/test01.dbf'
SIZE 1000M;
```

You can also increase or decrease the size of the tablespace by increasing or decreasing the size of the tablespace's data files with the RESIZE option. Note that you need to use the ALTER DATABASE command, not the ALTER TABLESPACE command, to resize a data file.

```
SQL> ALTER DATABASE DATAFILE '/finance10/oradata/data_09.dbf'
RESIZE 500m;
```

You can use the AUTOEXTEND provision when you create a tablespace or when you add data files to a tablespace to tell Oracle to automatically extend the size of the data files in the tablespace to a specified maximum. Here's the syntax for using the AUTOEXTEND feature:

```
SQL> ALTER TABLESPACE data01
ADD DATAFILE '/finance10/oradata/data01.dbf' SIZE 200M
AUTOEXTEND ON
NEXT 10M
MAXSIZE 1000M;
```

Proactive Tablespace Space Alerts

If a segment needs to be extended to accommodate the insertion of new data, there must be free space available in the tablespace that the segment belongs to. If not, the new data can't be inserted, and you'll get an Oracle error indicating that the operation failed due to the lack of space in the tablespace. You can write scripts to alert you that a tablespace is about to run out of space, but in Oracle Database 10*g* the database itself sends you proactive space alerts for all locally managed tablespaces, including the undo tablespace. The database will send out two types of tablespace out-of-space alerts: a warning alert and a critical alert. The warning alert cautions you that a tablespace's free space is running low, and the critical alert tells you that you should immediately take care of the free space problem so the database doesn't issue "out of space" errors. Both of these alerts are based on threshold values called warning and critical thresholds, which you can modify. There are two ways to set alert thresholds: you can specify that the database alert be based on the *percent of space used* or on the *number of free bytes left in the tablespace.*

Renaming Tablespaces

Oracle Database 10*g* lets you rename tablespaces by using the ALTER TABLESPACE command, as shown here:

```
SQL> ALTER TABLESPACE test01 RENAME TO test02;
Tablespace altered.
```

You can rename both permanent and temporary tablespaces, but there are a few restrictions:

• You can't rename the System and Sysaux tablespaces.
• The tablespace being renamed must have all its data files online.
• If the tablespace is read-only, renaming it doesn't update the file headers of its data files

Sometimes, you may need to rename a data file. The process for this is straightforward:

1. Take the data file offline by taking its tablespace offline. Use the following command:

```
SQL> ALTER TABLESPACE test01 OFFLINE NORMAL;
Tablespace altered.
SQL>
```

2. Rename the file using an operating system utility such as cp or mv in UNIX, or copy in Windows.

```
$ cp /u01/app/oracle/test01.dbf /u02/app/oracle/test01.dbf
```

3. Rename the data file in the database by using the following command:

```
SQL> ALTER TABLESPACE test01
2 RENAME DATAFILE
3 '/u01/app/oracle/test01.dbf'
4 TO
5* '/u02/app/oracle/test01.dbf';
Tablespace altered.
SQL>
```

Read-Only Tablespaces

All tablespaces are initially created as read/write. Use the READ ONLY clause in the ALTER TABLESPACE statement to change a tablespace to read-only. You must have the ALTER TABLESPACE or MANAGE TABLESPACE system privilege.

Before you can make a tablespace read-only, the following conditions must be met.

- The tablespace must be online. This is necessary to ensure that there is no undo information that needs to be applied to the tablespace.
- The tablespace cannot be the active undo tablespace or SYSTEM tablespace.
- The tablespace must not currently be involved in an online backup, because the end of a backup updates the header file of all datafiles in the tablespace

The following statement makes the flights tablespace read-only:

```
ALTER TABLESPACE flights READ ONLY;
```

Use the READ WRITE keywords in the ALTER TABLESPACE statement to change a tablespace to allow write operationsThe following statement makes the flights tablespace writable:

```
ALTER TABLESPACE flights READ WRITE;
```

Taking Tablespaces Offline

Except for the System tablespace, you can take any or all of the tablespaces *offline*—that is, you can make them temporarily unavailable to users. Four modes of offlining are possible with Oracle tablespaces: *normal*, *temporary*, *immediate*, and *for recovery*you can take any tablespace offline with no harm by using the following command:

SQL> ALTER TABLESPACE index_01 OFFLINE NORMAL;

To bring the tablespace online, use the following command:

SQL> ALTER TABLESPACE index_01 ONLINE;

ORACLE INDEXES

Oracle indexes provide speedy access to table rows by storing sorted values of specified columns. Indexes enable you to find a row with a certain column value without your having to look at more than a small fraction of the total rows in the table. An index is a tree structure that allows direct access to a row in a table. Indexes can be classified based on their logical design or physical implementation. For example, some classifications are:

- *Single column and composite indexes*. Composite index is built on multiple columns in a table whereas single column index has only one column . The maximum number of columns in a composite key is 32.
- *Unique and non-unique key*. A unique index guarantees no two rows of a table have duplicate values.
- *Function-based indexes*. Function-based indexes can be created as either a B-tree or a bitmap index. B-tree is the default index created by Oracle and bitmap indexes can be used when a table has millions of rows and there are few distinct values for the column to be indexed. For example, bitmap indexes may be more preferable for the gender and martial status column of a passport table.

Oracle Index Schemes

Oracle Database provides several indexing schemes that provide complementary performance functionality. These are:

- B-tree indexes: the default and the most common
- B-tree cluster indexes: defined specifically for cluster
- Hash cluster indexes: defined specifically for a hash cluster
- Global and local indexes: relate to partitioned tables and indexes
- Reverse key indexes: most useful for Oracle Real Application Clusters applications
- Bitmap indexes: compact; work best for columns with a small set of values
- Function-based indexes: contain the precomputed value of a function/expression
- Domain indexes: specific to an application or cartridge.

Creating Index

The syntax of creating a B-tree index is

CREATE [UNIQUE] INDEX <index name> ON <table name> (<column name>, [column name]...) ...

The syntax of creating a bitmap index is

CREATE BITMAP INDEX <index name> ON <table name> (<column name>, [column name]...) ...

There are other optional clauses after the ON clause which specify the detailed storage characteristics of an index. They are omitted here for simplicity.

Examples:

SQL> CREATE INDEX info ON employee (empno);

Bitmap Indexes

To create a bitmap index, you use the CREATE INDEX statement with the BITMAP keyword added to it:

SQL> CREATE BITMAP INDEX gender_idx ON employee(gender)
TABLESPACE emp_index_05;

Reverse-Key Indexes

The biggest advantage to using reverse-key indexes is that they tend to avoid hot spots when you do sequential insertion of values into the index. Here's how to create one:

SQL> CREATE INDEX reverse_idx ON employee(emp_id) REVERSE;

Function-Based Indexes

Function-based indexes precompute functions on a given column and store the results in an index. When WHERE clauses include functions, function-based indexes are an ideal way to index the column. Here's how to create a function-based index, using the LOWER function:

SQL> CREATE INDEX lastname_idx ON employee(LOWER(l_name));

Partitioned Indexes

Partitioned indexes are used to index partitioned tables. Oracle provides two types of indexes for partitioned tables: *local* and *global*

Global Indexes

Global indexes on a partitioned table can be either partitioned or nonpartitioned. The globally nonpartitioned indexes are similar to the regular Oracle indexes for nonpartitioned tables. You just use the regular CREATE INDEX syntax to create these globally nonpartitioned indexes. Here's an example of a global index on the ticket_sales table:

SQL> CREATE INDEX ticketsales_idx ON ticket_sales(month)
GLOBAL PARTITION BY range(month)
(PARTITION ticketsales1_idx VALUES LESS THAN (3)
PARTITION ticketsales1_idx VALUES LESS THAN (6)
PARTITION ticketsales2_idx VALUES LESS THAN (9)
PARTITION ticketsales3_idx VALUES LESS THAN (MAXVALUE);

Local Indexes

Locally partitioned indexes, unlike globally partitioned indexes, have a one-to-one correspondence with the table partitions. You can create locally partitioned indexes to match partitions or even subpartitions. Here is a simple example of creating a locally partitioned index on a partitioned table:

SQL> CREATE INDEX ticket_no_idx ON
ticket_sales(ticket__no) LOCAL
TABLESPACE localidx_01;

Monitoring Index Usage

Oracle Database provides a means of monitoring indexes to determine whether they are being used. If an index is not being used, then it can be dropped, eliminating unnecessary statement overhead. To start monitoring the usage of an index, issue this statement:

ALTER INDEX *index* MONITORING USAGE; Later, issue the following statement to stop the monitoring:

ALTER INDEX *index* NOMONITORING USAGE;

Rebuilding an Existing Index

You can use the REBUILD command on a periodic basis to reorganize indexes
to make them more compact and thus more efficient. You can also use the REBUILD command to alter the storage parameters you set during the initial creation of the index. Here's an example:

```
SQL> ALTER INDEX sales_idx REBUILD;
Index altered
```

USING MATERIALIZED VIEWS

A materialized view is a database object that contains the results of a query. The FROM clause of the query can name tables, views, and other materialized views. Collectively these objects are called **master tables** (a replication term) or **detail tables** (a data warehousing term). This reference uses "master tables" for consistency. The databases containing the master tables are called the **master databases**. You can do the following with a materialized view:

- Create indexes on a materialized view
- Create a materialized view on partitioned tables
- Partition a materialized view

You can use various types of aggregations like SUM, COUNT(*), AVG, MIN, and MAX in a materialized view. You can also use multiple table joins in the materialized view definition.

Query Rewriting

The materialized views are completely transparent to users. If users write qu ries using the underlying table, Oracle will automatically rewrite those queries to use the materialized views—this query-optimization technique is known as *query rewrite.*

The automatic query rewrite optimization technique is at the heart of materialized view usage. The QUERY_REWRITE_ENABLED initialization parameter determines whether Oracle will rewrite a query or not. When you enable query rewriting by setting QUERY_REWRITE_ENABLED = TRUE in your initialization parameter file, query rewriting is enabled system-wide, for the entire database.

The REWRITE_OR_ERROR Hint

If the queries take too long to complete without the materialized view, you can force Oracle to stop executing the query without the materialized view. You can use a hint
to tell Oracle to issue an error instead of executing the unrewritten query. The hint is called the REWRITE_OR_ERROR hint, and here's how you use it:

```
SQL> SELECT /*+ REWRITE_OR_ERROR */
prod_id SUM(quantity_sold) AS sum_sales_qty FROM sales_data
GROUP BY prod_id
```

Refreshing Materialized View Data

The following sections present the materialized view refresh options.

Refresh Mode You can choose between the ON COMMIT and ON DEMAND modes of data refresh.

• ON COMMIT: In this mode, whenever a data change in one of the master tables is committed, the materialized view is refreshed automatically to reflect the change.

• ON DEMAND: In this mode, you must execute a procedure like DBMS_MVIEW.REFRESH to update the materialized view.

Refresh Type

You can choose from the following four refresh types:

• COMPLETE: This refresh option will completely recalculate the query underlying the materialized view.

FAST: Under the fast refresh mechanism, Oracle will use a *materialized view log* to log all changes to the master tables.

FORCE: If you choose this option, Oracle will try to use the FAST refresh mechanism.

NEVER: This refresh option never refreshes a materialized view.

Creating Materialized Views

There are three steps required to get the materialized views going, although the creation itself is simple:

1. Grant the necessary privileges.

2. Create the materialized view log (assuming you're using the FAST refresh option).

3. Create the materialized view.

Granting the Necessary Privileges

You must first grant the necessary privileges to the user who is creating the materialized views. The main privileges are those that enable the user to create a materialized view. In addition, you must grant the QUERY REWRITE privilege to the user, either by using the GLOBAL QUERY REWRITE privilege or specific QUERY REWRITE privileges on each object that is not part of the user's schema.

Here are the GRANT statements that enable a user to create a materialized view in the user's schema:

```
SQL> GRANT CREATE MATERIALIZED VIEW TO salapati;
SQL> GRANT QUERY REWRITE TO salapati;
```

Creating the Materialized View Log

Here's how you create the materialized view log:

```
SQL> CREATE MATERIALIZED VIEW LOG ON products;
Materialized view log created.
SQL> CREATE MATERIALIZED VIEW LOG ON sales;
Materialized view log created.
```

Creating the Materialized View

Now you are ready to create your materialized view. The example, shown in Listing 4 uses the REFRESH COMPLETE clause, to specify the COMPLETE refresh option.

Listing 4. *Creating a Materialized View*

```
SQL> CREATE MATERIALIZED VIEW test_mv
2 BUILD IMMEDIATE
3 REFRESH FAST ON COMMIT
4 ENABLE QUERY REWRITE
5 AS
6 SELECT sh.products.prod_category,
7 SUM(sh.sales.quantity_sold),
8 COUNT(sh.sales.quantity_sold), count(*)
10 FROM sh.sales, sh.products
11 WHERE sh.products.prod_id = sh.sales.prod_id
12 AND sh.products.prod_category <= 'Women'
13 AND sh.products.prod_category >= 'Boys'
14 GROUP BY sh.products.prod_category;
Materialized view created.
```

Oracle Transaction Management

A *transaction* is a logical unit of work consisting of one or more SQL statements. A transaction may perform one operation or an entire series of operations on the database objects, either interactively or as part of a program.

Oracle Transactions

The effects of all the SQL statements in a transaction can be either all **committed** (applied to the database) or all **rolled back** (undone from the database).A transaction begins with the first executable SQL statement. A transaction ends when it is committed or rolled back, either explicitly with a COMMIT or ROLLBACK statement or implicitly when a DDL statement is issued. A transaction starts implicitly when the first executable SQL statement begins, and it continues as the following SQL statements are processed until one of the following events occurs:

• COMMIT: If a transaction encounters a COMMIT statement, all the changes to that point are made permanent in the database.

• ROLLBACK: If a transaction encounters a ROLLBACK statement, all changes made up to that point are cancelled.

• *DDL statement*: If a user issues a DDL statement, such as CREATE, DROP, RENAME, or ALTER, Oracle first commits any current DML statements that are part of the transaction, before executing and committing the results of the DDL statement. This is called an *implicit commit*, since the committing of the DML statements immediately preceding the DDL statements isn't explicitly done by the user.

• *Normal program conclusion*: If a program ends without errors, all changes are implicitly committed by the database.

• *Abnormal program failure*: If the program crashes or is terminated, all changes made by it are implicitly rolled back by the database.

COMMIT Statement

The COMMIT statement ends a transaction successfully. All changes made by all SQL statements since the transaction began are recorded permanently in the database.

You can commit a transaction by using either of the following statements, which make the changes permanent:

SQL> COMMIT;

SQL> COMMIT WORK;

When an Oracle transaction is committed, the following three things happen:

1. The transaction tables in the redo records are tagged with the unique system change number (SCN) of the committed transaction.

2. The log writer writes the redo log information for the transaction from the redo log buffer to the redo log files on disk, along with the transaction's SCN. This is the point at which a commit is considered complete in Oracle.

3. Any locks that Oracle holds are released, and Oracle marks the transaction as complete. The default behavior for the COMMIT statement, which is generally the only type you'll encounter, is to use the IMMEDIATE and WAIT options:

• IMMEDIATE vs. BATCH: With the IMMEDIATE option, the log writer writes the redo log records for the committing transaction immediately to disk. If you'd rather the log writer write the redo records by buffering them in memory until it's convenient to write them, you can use the alternative BATCH option.

• WAIT vs. NOWAIT: With the WAIT option, the COMMIT statement doesn't return as successful until the redo records are successfully written to the redo logs. If you'd rather have the COMMIT statement return without waiting for the writing of the redo records, you can use the NOWAIT option. You can modify this default behavior by using the COMMIT_WRITE initialization parameter at either the system or the session level. To specify the BATCH and NOWAIT options by default, you can use the COMMIT_WRITE initialization parameter in the following way:

COMMIT_WRITE = BATCH, NOWAIT

ROLLBACK Statement

The ROLLBACK statement undoes, or rolls back, the changes made by SQL statements within a transaction, so long as you didn't already commit the transaction. Once you issue the ROLLBACK statement, none of the changes made to the tables by SQL statements since the transaction began are recorded to the database permanently.

SQL> ROLLBACK;

You can also partially roll back the effects of a transaction by using *save points* in the transaction. Using a save point, you can roll back to the last SAVEPOINT command in the transaction, as follows:

SQL> ROLLBACK TO SAVEPOINT POINT A;

Transaction Properties

Transactions in RDBMSs must possess four important properties, symbolized by the ACID acronym, which stands for *atomicity*, *consistency*, *isolation*, and *durability* of transactions. Let's look at the transaction properties

- *Atomicity*: Either a transaction should be performed entirely or none of it should be performed.
- *Consistency*: The database is supposed to ensure that it's always in a consistent state.
- *Isolation*: Isolation means that although there's concurrent access to the database by multiple transactions, each transaction must appear to be executing in isolation. The isolation property of transactions ensures that a transaction is kept from viewing changes made by another transaction before the first transaction commits
- *Durability*: The last ACID property, durability, ensures that the database saves commit transactions permanently.

Transaction Concurrency Control

Transaction concurrency is achieved by managing various users' simultaneous transactions without permitting any interference among them

One solution to concurrency control is to lock the entire table for the duration of each operation, so one user's transactions do not impact another's. Oracle *does use* locking mechanisms to keep the data consistent, but the locking is done in the least restrictive fashion, with the goal of maintaining the maximum amount of concurrency.

Concurrency Problems

Some of the most important problems potentially encountered in concurrent transaction processing are dirty reads, phantom reads, lost updates, and nonrepeatable reads.

- **The Dirty-Read Problem:** A *dirty read* occurs when a transaction reads data that has been updated by an ongoing transaction but has not been committed permanently to the database.
- **The Phantom-Read Proble:** Phantom-read problems are caused by the appearance of new data in between two database operations in a transaction.
- **The Lost-Update Problem:** The *lost-update* problem is caused by transactions trying to read data while it is being updated by other transactions.
- **The Nonrepeatable-Read (Fuzzy-Read) Problem:** When a transaction finds that data it has read previously has been modified by some other transaction, you have a *nonrepeatable-read* (or fuzzy-read) problem.

Schedules and Serializability

If the database permits concurrent access, then you need to consider the cumulative effect of all the transactions on database consistency. To do this, the database uses a *schedule*, which is a sequence of operations from one or more transactions. If all the transactions executed serially, one after another, the schedule would also be *serial.* If the database can produce a schedule that is equivalent in its effect to a serial schedule, even though it may be derived from a set of concurrent transactions, it is called a *serializable schedule.*

Isolation Levels

Isolation of transactions keeps concurrently executing database transactions from viewing incomplete results of other transactions. The main isolation levels are the serializable, repeatable

read, read-uncommitted, and read-committed isolation levels. Here's what the different levels of transaction isolation levels mean:

Serializable: Under the serializable level of isolation, the transaction will lock all the tables it is accessing, thereby preventing other transactions from updating any of the tables underneath it until it has completed its transaction by using a COMMIT or ROLLBACK command.

• *Repeatable read*: The repeatable-read isolation level guarantees read consistency—a transaction that reads the data twice from a table at two different points in time will find the same values each time. You avoid both the dirty-read problem and the nonrepeatable-read probe-

• *Read uncommitted*: The read-uncommitted level, which allows a transaction to read another transaction's intermediate values before it commits, will result in the occurrence of all the problems of concurrent usage.

• *Read committed*: Oracle's default isolation level is the read-committed level of isolation at the statement level. Oracle queries see only the data that was committed at the beginning of the query. Because the isolation level is at the statement level, each statement is allowed to see only the data that was committed before the commencement of that statement. The read-committed level of isolation guarantees that the row data won't change while you're accessing a particular row in an Oracle table.

Transaction- and Statement-Level Consistency

Oracle automatically provides *statement-level* read consistency by default. That is, all data that a query sees comes from a single point in time. This means that a query will see consistent data when it begins. The query sees only data committed before it starts, and no data committed during the course of the query is visible to it.

Changing the Default Isolation Level

You can change the isolation level from the default level of read-committed to a serializable isolation level using the following statement:

SQL> ALTER SESSION SET ISOLATION LEVEL SERIALIZABLE;

A serializable level of isolation is suited for databases where multiple consistent queries need to be issued during an update transaction. However, serialization isn't a simple choice, because it seriously reduces your concurrency.

Implementing Oracle's Concurrency Control

A database may use one or more methods to implement concurrency of use. These include locking mechanisms to guarantee exclusive use of a table by a transaction, time-stamping methods that enable serialization of transactions, and the validation-based scheduling of transactions. Oracle uses a combination of the available methods.

Here are some important features of Oracle locking:

• Oracle implements locks by setting a bit in the data item being locked. The locking information is stored in the data block where the row lives.

• Locks are held for the entire length of a transaction and are released when a COMMIT or a ROLLBACK statement is issued.

• Oracle doesn't use lock escalation. Oracle doesn't need to escalate locks, as it stores the locking information in the individual data blocks. Lock escalation—for example, an escalation from the row level to the table level—reduces concurrency.

• Oracle does use *lock conversion,* which involves changing the restrictiveness of a lock while keeping the granularity of the lock the same.

Oracle Locking Methods

Oracle uses locks to control access to two broad types of objects: user objects, which include tables, and system objects, which may include shared memory structures and data dictionary objects. Oracle follows a pessimistic locking approach, which anticipates potential conflicts and will block some transactions from interfering with others in order to avoid conflicts between concurrent transactions.

Oracle Lock Types

Oracle locks can be broadly divided into the following types, according to the type of object that is locked: DML locks, DDL locks, latches, internal locks, and distributed locks. These lock types are described in the following sections.

DML Locks

DML locks are locks placed by Oracle to protect data in tables and indexes. Whenever a DML statement seeks to modify data in a table, Oracle automatically places a row-level lock on the rows in the table that are being modified. Any Oracle lock mode will permit queries on the table. A query will never block an update, delete, or insert, and vice versa. An *exclusive lock* only permits queries on a table, and prevents users from performing any other activity on it, like updating or deleting data. A *row exclusive lock*,on the other hand, allows concurrent access to a table for updating, deleting, and inserting data, but prevents any user from locking the entire table for exclusive use.

Table 3 summarizes the row-level and table-level DML locks that are acquired for the most common database operations.

Table 3. *DML Row- and Table-Level Locks Held for Common Operations*

Operation	**Row-Level Lock**	**Table-Level Lock**
SELECT . . . FROM *table*	None	None
INSERT INTO *table*	Exclusive	Row exclusive
UPDATE *table*	Exclusive	Rowexclusive
INSERT INTO *table*	Exclusive	Row exclusive
DELETE FROM *table*	Exclusive	Row exclusive

DDL Locks

- Protects the definition of an object while being used by a DDL operation. Recall that a DDL statement implicitly commits.
- Create Procedure will automatically acquire DDL locks for all schema objects referenced in the procedure definition. The DDL locks prevent objects referenced in the procedure from being altered/dropped before the compile is complete.

- Cannot explicitly request DDL locks. Individual schema objects that are modified or referenced are locked during DDL operations; the whole data dictionary is never locked.
- Three categories: exclusive DDL locks, share DDL locks, and breakable parse locks.

Latches, Internal Locks, and Distributed Locks

Latches are internal mechanisms that protect shared data structures in the SGA. *Data dictionary locks* are used by Oracle whenever the dictionary objects are being modified.

Distributed locks are specialized locking mechanisms used in a distributed database system or in the Oracle Real Application Clusters (RAC) environment. *Internal locks* are used by Oracle to protect access to structures such as data files, tablespaces, and rollback segments.

Explicit locking in oracle

Oracle provides explicit locking features to override the implicit locks placed by Oracle on behalf of transactions. You can override Oracle's default (implicit) locking mechanism at the transaction level or the session level.

Blocking Locks

A blocking lock occurs when a lock placed on an object by a user prevents or blocks other users from accessing the same object or objects. The DBA_BLOCKERS table is useful in getting this information— it tells you which sessions are currently holding locks on objects for which some other object is presently waiting.

Here is the SQL statement:

```
SQL> SELECT a.username, a.program, a.sid, a.serial#
2 FROM v$session a, dba_blockers b
3 WHERE a.sid = b.holding_session;
```

Deadlocks

Deadlocks occur in any RDBMS when two sessions block each other while each waits for a resource that the other session is holding. When Oracle encounters a deadlock between transactions, it records in the trace file the session IDs involved, the SQL statements issued in the transactions, and the specific object name and the rows on which locks are held in each session involved in the deadlock. You can avoid deadlocks by paying attention in the design phase and ensuring the proper locking order of the objects.

Managing Oracle Locks

You can use either a script-based approach or the Oracle Enterprise Manager to analyze locks in your instance.

Using SQL to Analyze Locks

Oracle provides a script called utllockt.sql that gives you a lock wait-for graph in a tree-structured format showing sessions that are holding locks that are affecting other sessions. Here's a sample execution of the
utllockt.sql script:

```
SQL> @$ORACLE_HOME/rdbmsa/admin/utllockt.sql
```

Waiting session	Type	Mode requested	Mode Held	Lock Id1
682	None	None	None	0
363	TX	Share (S)	Exclusive (X)	

Using undo data to provide read consistence

Oracle uses special structures called undo records to help provide automatic statement level read consistency. Only data committed when a query begins will be seen by the query. If a transaction is modifying data, oracle will write a before image of the table data to its undo records. SCN – identifies the order in which the transactions occurred in the database.

Purposes

- Providing read consistency for sql queries
- Rolling back unwanted active transactions
- Recovering terminated transactions
- Analyzing older data by using flashback query
- Recovering from logical corruptions using flashback features

Automatic undo management-AUM

- More efficient and easy
- Create adequate sized tablespace for storing undo information
- Database will automatically allocate and deallocate undo segments to match the transaction

Advantages

- Includes flashback query features
- Avoids many of the errors associated with the older technique of managing undo data by using rollback segments
- Eliminates most of the undo block and consistent read contention
- Undo segments use space much more efficiently by exchanging space dynamically with other segments
- When undo segments are not needed, it will reclaim the space used by the segments
- It manages all undo space allocation automatically behind the scenes

Setting up AUM

Three ways

1. UNDO_MANAGEMENT
2. UNDO_TABLESPACE
3. UNDO_RETENTION

Undo classification can be classified in two broad types:

1. If a transaction that has gnerated the undo data is still active, the undo data is said to be active(uncommitted)
2. If the transaction that generated the undo data is inactive(committed) - committed
3. You can set the undo rentention size by specifying it in the initialization file as follows: UNDO_RENTENTION = 1800 /*(30 minutes)

The default value is 900 seconds.

Undo Retention

- Committed undo information normally is lost when its undo space is overwritten by a newer transaction. However, for consistent read purposes, long-running queries sometimes require old undo information for undoing changes and producing older images of data blocks. The success of several Flashback features can also depend upon older undo information

- You enable the guarantee option by specifying the RETENTION GUARANTEE clause for the undo tablespace when it is created by either the CREATE DATABASE or CREATE UNDO TABLESPACE statement. Or, you can later specify this clause in an ALTER TABLESPACE statement. You *do not* guarantee that unexpired undo is preserved if you specify the RETENTION NOGUARANTEE clause.
- You can use the DBA_TABLESPACES view to determine the RETENTION setting for the undo tablespace. A column named RETENTION will contain a value on GUARANTEE, NOGUARANTEE, or NOT APPLY (used for tablespaces other than the undo tablespace).

Sizing the Undo Tablespace

- Using Auto-Extensible Tablespaces
- Sizing Fixed-Size Undo Tablespaces - If you have decided on a fixed-size undo tablespace, the Undo Advisor can help you estimate needed capacity, and you can then calculate the amount of retention your system will need. You can access the Undo Advisor through Enterprise Manager or through the DBMS_ADVISOR PL/SQL package
- The Undo Advisor PL/SQL Interface- Oracle Database provides an Undo Advisor that provides advice on and helps automate the establishment of your undo environment. You activate the Undo Advisor by creating an undo advisor task through the advisor framework

Monitoring the Undo Tablespace

View	Description
V$UNDOSTAT	Contains statistics for monitoring and tuning undo space. Use this view to help estimate the amount of undo space required for the current workload. The database also uses this information to help tune undo usage in the system. This view is meaningful only in automatic undo management mode.
V$ROLLSTAT	For automatic undo management mode, information reflects behavior of the undo segments in the undo tablespace
V$TRANSACTION	Contains undo segment information
DBA_UNDO_EXTENTS	Shows the status and size of each extent in the undo tablespace
WRH$_UNDOSTAT	Contains statistical snapshots of V$UNDOSTAT information
WRH$_ROLLSTAT	Contains statistical snapshots of V$ROLLSTAT information.

TRANSACTION QUERY

One of the important new features of Oracle Database 10g is the Flashback Transaction Query. It is a diagnostic tool to view the changes made to the database at the transaction level. This feature will help diagnose problems, analyze and audit transactions, and recover from user or application errors. The undo SQL generated by the flashback transaction query can be used to rollback the changes made by a transaction. Flashback Transaction Query can be used to reconstruct the SQL statements used to make changes in the database, and those that can be used to undo the change.

Flashback Transaction Query uses an indexed access path to get to the undo data. It is faster than the LogMiner, which mines the redo log files to obtain the undo information.

The FLASHBACK_TRANSACTION_QUERY view lets you identify which transaction or transactions were responsible for a certain change in a table's data during a specified interval. You need the SELECT ANY TRANSACTION system privilege to query the FLASHBACK_TRANSACTION_ QUERY view. This view contains columns that let you identify a transaction's time stamp, the identity of the user who made the transaction, the type of operations done during the transaction, and the undo statements necessary to retrieve the original row.

The FLASHBACK_TRANSACTION_QUERY view contains the following columns:

• START_SCN and START_TIMESTAMP identify when a certain row was created.

• COMMIT_SCN and COMMIT_TIMESTAMP tell you when a transaction was committed.

• XID, ROW_ID, and UNDO_CHANGE# identify the transaction, the row, and the undo change number, respectively.

• OPERATION tells you whether the DML operation was an insert, update, or delete operation.

The following query will display all transactions, both committed and active, in all the undo segments:

```
SQL> SELECT operation, undo_sql, table_name
FROM flashback_transaction_query;
```

Using Flashback Version Query

You use a Flashback Version Query to retrieve the different versions of specific rows that existed during a given time interval. A new row version is created whenever a COMMIT statement is executed.

The Flashback Version Query returns a table with a *row for each version* of the row that existed at any time during the time interval you specify. Each row in the table includes pseudocolumns of metadata about the row version. The pseudocolumns available are

VERSIONS_XID : Identifier of the transaction that created the row version

VERSIONS_OPERATION : Operation Performed. I for Insert, U for Update, D for Delete

VERSIONS_STARTSCN : Starting System Change Number when the row version was created

VERSIONS_STARTTIME : Starting System Change Time when the row version was created

VERSIONS_ENDSCN : SCN when the row version expired.

VERSIONS_ENDTIME : Timestamp when the row version expired

The Flashback Table Feature

You can use the Flashback Table feature to roll back changes to a previous point in time defined by either a time stamp or an SCN. Flashback Table uses undo information to restore data rows in changed blocks of tables with DML statements like INSERT, UPDATE, and DELETE. These DML operations change the row IDs of the affected rows, so you must ensure that you have enabled row movement in the tables you are using for the Flashback Table feature, as shown here:

```
SQL> ALTER TABLE emp ENABLE ROW MOVEMENT;
Table altered.
```

Before you use the Flashback Table feature, note its complete syntax:

SQL> FLASHBACK TABLE
[*schema.*]*table*
[,[*schema*.]*table]* . . .
TO {{SCN|TIMESTAMP} *expr*
[{ENABLE|DISABLE}TRIGGERS]
|BEFORE DROP[RENAME TO *table*]
};

Discrete Transactions

- When you specify a transaction as a discrete transaction, oracle skips certain routine processing overhead such as writing the undo records.
- It doesn't modify the data blocks until the transaction commits.
- Use BEGIN_DISCRETE_TRANSACTION to implement the discrete transaction strategy.
- The data blocks aren't modified until the discrete transaction commits.

Autonomous transactions

- A transaction can run as part of another transaction.
- The parent transaction is the main transaction
- Independent child transaction is the autonomous transaction, it can be called from another transaction
- Packages, procedures, functions and triggers – autonomous transactions
- Autonomous transactions have its own ROLLBACK and COMMIT.
- The main transaction can be paused and autonomous transactions can be executed.
- it provides develoeprs with thte ability to create more fine-grained transactions
- Can have nested autonomous transactions
- Provides lot of flexibility

Example:

```
CREATE TABLE at_test (
  id          NUMBER      NOT NULL,
  description  VARCHAR2(50) NOT NULL
);
INSERT INTO at_test (id, description) VALUES (1, 'Description for 1');
INSERT INTO at_test (id, description) VALUES (2, 'Description for 2');
SELECT * FROM at_test;
       ID DESCRIPTION
---------- --------------------------------------------------
         1 Description for 1
         2 Description for 2
2 rows selected.
```

Next, we insert another 8 rows using an anonymous block declared as an autonomous transaction, which contains a commit statement.

```
DECLARE
  PRAGMA AUTONOMOUS_TRANSACTION;
```

```
BEGIN
  FOR i IN 3 .. 10 LOOP
    INSERT INTO at_test (id, description)
    VALUES (i, 'Description for ' || i);
  END LOOP;
  COMMIT;
END;
/
PL/SQL procedure successfully completed.
SELECT * FROM at_test;
        ID DESCRIPTION
---------- --------------------------------------------------
         1 Description for 1
         2 Description for 2
         3 Description for 3
         4 Description for 4
         5 Description for 5
         6 Description for 6
         7 Description for 7
         8 Description for 8
         9 Description for 9
        10 Description for 10
10 rows selected.
```

As expected, we now have 10 rows in the table. If we now issue a rollback statement we get the following result.

```
ROLLBACK;
SELECT * FROM at_test;
        ID DESCRIPTION
---------- --------------------------------------------------
         3 Description for 3
         4 Description for 4
         5 Description for 5
         6 Description for 6
         7 Description for 7
         8 Description for 8
         9 Description for 9
        10 Description for 10
8 rows selected.
SQL>
```

Resumable Space Allocation

Resumable space allocation, is for all tablespaces at the session level. Database operations are suspended when an out-of-space condition is encountered. These suspended operations automatically resume when the error condition disappears.

Resumable Operations

The following types of database operations are resumable:

• *Queries*: These operations can always be resumed after they run out of temporary sorting space.

• *DML operations*: Insert, update, and delete operations can be resumed after an error is issued.

• *DDL operations*: Index operations involving creating, rebuilding, and altering are resumable, as are CREATE TABLE AS SELECT operations and several other DDL operations.

• *Import and export operations*: SQL*Loader data load jobs that run out of space are resumable.

You must use the RESUMABLE parameter when you specify the SQL*Loader job, to make the operation resumable. Two other resumable operation parameters, RESUMABLE_TIMEOUT and RESUMABLE_NAME, can be set only if you set the RESUMABLE parameter.

Common Resumable Errors

- *Out of space errors*:
- *Maximum extents errors*:
- *User's space quota errors*:

Using the Resumable Space Allocation Feature

To use the Resumable Space Allocation feature, a user must have the appropriate privileges:

```
SQL> GRANT RESUMABLE TO salapati;
Grant succeeded.
SQL>
```

When you wish to revoke the privilege, use the following command:

```
SQL> REVOKE RESUMABLE FROM salapati;
Revoke succeeded.
```

Resumable Mode

Operations can be made resumable by explicitly switching the session mode using:

```
ALTER SESSION ENABLE RESUMABLE;
ALTER SESSION DISABLE RESUMABLE;
```

Timeout Period

Operations will remain in a suspended state until the timeout period, 2 hours by default, is reached. The timeout period can be modified using any of the following commands:

```
ALTER SESSION ENABLE RESUMABLE TIMEOUT 3600;
EXECUTE Dbms_Resumable.Set_Timeout(3600);
ALTER SESSION ENABLE RESUMABLE TIMEOUT 3600 NAME 'insert into table';
```

Alter session enable resumable

The ALTER SESSION ENABLE RESUMABLE statement is used to activate resumable space allocation for a given session. A new parameter, called RESUMABLE, is used to enable resumable space allocation for export, import and load utilities.

You can also set the timeout interval using the DBMS_RESUMABLE package, as follows:

```
SQL> EXECUTE DBMS_RESUMABLE.set_session_timeout(4349,18000);
PL/SQL procedure successfully completed.
```

POSSIBLE QUESTIONS:

PART-B: 6 MARKS

1. Describe in detail creating and managing tablespaces.
2. What is an Index? Explain the types of indexes in detail.
3. Distinguish between a view and materialized view and give the syntax of creating a materialized view
4. What are the different types of isolation levels and locks available in Oracle?
5. How transaction concurrency control mechanism is implemented in Oracle? Explain.
6. Write briefly about discrete and autonomous transactions.
7. How resumable space allocation is used in transaction management?
8. Write a brief note on Flashback Transaction Query.
9. Explain in detail using undo data to provide read consistency.
10. Explain the usage of COMMIT and ROLLBACK statements with examples.

Karpagam Academy of Higher Education
Department of CS, CA & IT
Subject: Oracle 10g Administration (17CSP203)

Batch : 2017-2019 **Class: I M.Sc CS**

Objective Type Questions
UNIT II

S.NO	QUESTIONS	OPTION 1	OPTION 2	OPTION 3	OPTION 4	KEY
1	_____tablespaces are used to store objects for the duration of a user's session only	undo	temporary	permanent	locally	temporary
2	_____ tablespaces are a type of permanent tablespace that are used to store undo data.	undo	temporary	permanent	locally	undo
3	_____ tablespace have to constantly check the data dictionary during the course of extend management	temporary	permanent	locally managed	dictionary managed	dictionary managed
4	Oracle keep track of how much free space is in its data blocks by maintaining _____	buffers	freelists	freespace	freeorder	freelists
5	Every table and index maintains a list of all its data blocks with freespace greater than	PCTFREE	PCTUSED	PCTNULL	PCTSPACE	PCTUSED
6	The _____ parameter lets you reserve a percentage of space in each data block for future updates to existing data	PCTFREE	PCTUSED	PCTNULL	PCTSPACE	PCTFREE
7	_____ is the default for extend management.	ALLOCATE	EXTEND	AUTOALLOCATE	RESIZE	AUTOALLOCATE
8	To remove the tablepace from the database use the command_____	REMOVE	DROP	DELETE	ALTER	DROP
9	You can use the new _____ data dictionary view to manage the temporary tablespace groups in your database	DBA_TABLESPACE_GROUPS	DBA_TABLESPACE_VIEW	DBA_TABLES	DBA_GROUPS	DBA_TABLESPACE_GROUPS
10	_____ provides speedy access to table rows by stroing sorted values of specified columns	oracle processes	oracle files	oracle indexes	oracle tables	oracle indexes
11	_____ indexes are the unique indexes in a table that must always posses a value	secondary	nonunique	primary	composite	primary
12	_____ indexes are indexes that contain two or more columns from the same table.	secondary	nonunique	primary	composite	composite

13	_____ indexes are also known as concatenated indexes	secondary	nonunique	primary	composite	composite
14	_____ indexes are structured in the form of an inverse tree.	primary	composite	B-tree	h-tree	B-tree
15	_____ is a virtual table.	view	index	functions	tables	view
16	_____ are static objects that derive their data from the underlying base tables.	view	materialized views	index	functions	materialized views
17	Oracle will automatically rewrite the queries using the techniques called _____	query writable	query rearrange	query rewrite	query change	query rewrite
18	Materialized view is automatically refreshed when _____ mode is used	ON DEMAND	ON END	ON COMMIT	ON EXIT	ON COMMIT
19	Under the _____ mechanism, oracle will use a materialized view log to log all changes to the master table.	COMPLETE	FAST	FORCE	NEVER	FAST
20	A _____ is a logical unit of work consisting of one or more SQL statements	order	transaction	recovery	backup	transaction
21	The majority of _____ statements retrive data from the database.	DML	DDL	Functional	indexed	DML
22	If a transaction encounters a _____ statement, all the changes to that point are made permanent in the database.	ROLLBACK	COMMIT	RECOVERY	BACKUP	COMMIT
23	If a transaction encounters a _____ statement, all changes made up to that point are cancelled	ROLLBACK	COMMIT	RECOVERY	BACKUP	ROLLBACK
24	With the _____ option, in the commit statement, the log wirter writes the redo log records for the committing transaction immediately to disk.	IMMEDIATE	BATCH	WAIT	NOWAIT	IMMEDIATE
25	You can partially rollback the effects of a transaction by using _____	SAVEPOINT	BACK	WAIT	NOWAIT	SAVEPOINT
26	A _____ occurs when a transaction reads data that has been updated by an ongoing transaction but has not been committed permanently to the database	dirty read	phantom read	lost update	fuzzy read	dirty read
27	The _____ level, allows a transaction to read another transaction's intermediate values before it commits	serializable	repeatable read	read uncommited	read committed	read uncommited
28	Oracle uses _____ granularity to lock objects	row level	table level	column level	serial level	row level

29	An _____ only permits queries on a table, and prevents users from perfroming any other activity on it.	exclusive lock	inclusive lock	import lock	export lock	exclusive lock
30	_____ are internal mechanisms that protect shared data structures in the SGA.	internal lock	distributed lock	latches	DDL locks	latches
31	_____ locks are used by oracle whenever the dictionary objects are being modified.	internal lock	distributed lock	latches	data dictionary	data dictionary
32	_____ are used by oracle to protect access to structure such as data files, tablespaces and rollback segments	internal lock	distributed lock	latches	data dictionary	internal lock
33	_____ locks are used in oracle real application clusters	internal lock	distributed lock	latches	data dictionary	distributed lock
34	Oracles uses _____ to provide automatic statement level read consistency	undo records	undo read	undo locks	undo buffer	undo records
35	The _____ parameter lets you control the reuse of the committed undospace.	expired	unexpired	undo_retention	undo_time	undo_retention
36	The default retention time is _____ seconds	1000	1200	900	800	900
37	_____ retrives data from a past point in time	flashback query	flashback data	flashback table	flashback statements	flashback query
38	The independent child transaction is called as _____ transaction	parent	autonomous	discrete	non autonomous	autonomous
39	_____keyword indicates the percentage of space reserved in the index blocks	PCTFREE	PCTTHRESHOLD	THRESHOLD	PCTFREE THRESHOLD	PCTTHRESHOLD
40	In Indexed Organised structure the data is stored in _____index structure.	B-tree Index Structure	Hybrid Structure	Logical Structure	Network structure	B-tree Index Structure
41	_____is used to store two or more tables together	Cluster	Group	Joins	Composite	Cluster
42	_____package is used to estimate the size of a new index.	DBMS_SPACE	INDEX_COST1	DBMS_SIZE	INDEX_COST1	DBMS_SPACE
43	Views can be deleted by using _____command	Delete	Remove	Replace	Drop	Drop
44	_____view takes up physical space in our database.	Materialized view	Updatable view	Cursor view	Physical View	Materialized view

45	_____problems are caused by the appearance of new data in between two database operations in a transaction.	Lost-Update problem	Dirt-Read Problem	Phantom-Read Problem	Nonrepeatable-Read Problem	Phantom-Read Problem
46	_____isolation level will lock all the tables	Repeatable read	Serializable	Read uncommitted	Read Commited	Serializable
47	_____isolation level guarantees that the row data won't change while we are accessing a particular row in an Oracle table.	Repeatable read	Serializable	Read uncommitted	Read Commited	Read Commited
48	Serializable transactions are more prone to_____.	deadlocks	locks	throughput	Phanthom	deadlocks
49	____ specifies the name and the location of the bad file.	log	bad	undo	dirty	bad
50	_____ is used for the current date.	date	sysdate	today	timestamp	sysdate
51	Direct path loading method doesn't use the _____ statement to put data into the tables.	SQL insert	delete	drop	update	SQL insert
52	_____parameter is used to show a value of LOCAL or DICTIONARY managed tablespace.	Initial Extent	Next Extent	Extent Management	Segment Space Management	Extent Management
53	_____parameter is used to set the limits of the tablespaces	LIMIT	MAXSIZE	UNLIMITED	AUTOEXTEND	MAXSIZE
54	_____tablespace contains only one very large data file.	Bigfile tablespace	Large file tablespace	Voulme tablespace	Default tablespace	Bigfile tablespace
55	Which clause is used to drop the constraints permanently?	DELETE CONSTRAINT	REMOVE CONSTRAINT	CASCADE CONSTRAINT	PURGE CONSTRAINT	CASCADE CONSTRAINT
56	_____keyword indicates the percentage of space reserved in the index blocks	PCTFREE	PCTTHRESHOLD	THRESHOLD	PCTFREE THRESHOLD	PCTTHRESHOLD
57	You can't drop the _____tablespace by using DROP command.	permanent	temporary	sysaux	local	sysaux

UNIT-III

SYLLABUS:

Loading and transforming data**:** overview of extraction transformation and loading - using external tables to load data - transforming data. Using data pump export and import: introduction - performing exports and imports -monitoring - transportable table spaces.
Managing and monitoring operational databases: types of oracle performance statistics - server generated alerts - automatic workload repository - active session history - undo and MTTR advisors.

An Overview of Extraction, Transformation, and Loading

Before you can run your application on an Oracle database, you need to populate your database. Most warehouse data goes through three major steps before you can analyze the data *extraction, transformation, and loading* (ETL). These steps are defined as follows:

• ***Extraction*** is the identification and extraction of raw data, possibly in multiple formats, from several sources, not all of which may be relational databases.
• ***Transformation*** of data is the most challenging and time-consuming of the three processes. Transformation of data may involve the application of complex rules to data. It may also include performing operations such as data aggregation and the application of functions to the raw data.
• ***Loading*** is the process of placing the data in the database tables. This may also include the task of maintaining indexes and constraints on the tables.

Traditionally, organizations have used two different methods of performing the ETL process: the *transform-then-load* method and the *load-then-transform* method.
Oracle's ETL solution includes the following components:
• **External tables**: External tables provide a way to merge the loading and transformation processes. Using external tables will enable you to eliminate cumbersome and time-consuming intermediate staging tables during data loading.
• **Multitable inserts**: Using the multitable insert feature, you can insert data into more than one table at the same time, using different criteria for the various tables. This capability eliminates the additional step of first dividing data into separate groupings and then performing data loading.
• **Upserts:** This is simply a made-up name indicating the technique by which you can either insert data into a table or just update the rows with a single SQL statement: MERGE. The MERGE statement will insert new data and update data if the rows already exist in the table. This simplifies your loading process because you don't need to worry about whether a table already contains the data.
• **Table functions**: Table functions produce a set of rows as output. Table functions return a collection type instance (nested table and VARRAY data types). Table functions are similar to views, but, instead of defining the transform declaratively in SQL, you define it procedurally in PL/SQL. Table functions are a great help when you're doing large and

complex transformations, because you can perform the transformations before loading data into a data warehouse.

• **Transportable tablespaces**: These tablespaces provide you with an efficient and speedy way to move data from one database to another. For example, you can migrate data between an OLTP database and a data warehouse using transportable tablespaces.

Using External Tables to Load Data

- The external table descriptor is also called the external table layer. This external table layer, along with the access driver, maps the data in the external file to the external table definition.
- External tables are created using the SQL CREATE TABLE...ORGANIZATION EXTERNAL statement. When you create an external table, you specify the following attributes:
- TYPE - specifies the type of external table. The two available types are the ORACLE_LOADER type and the ORACLE_DATAPUMP type. Each type of external table is supported by its own access driver.
- The ORACLE_LOADER access driver is the default. It can perform only data loads, and the data must come from text datafiles. Loads from external tables to internal tables are done by reading from the external tables' text-only datafiles.
- The ORACLE_DATAPUMP access driver can perform both loads and unloads. The data must come from binary dump files. Loads to internal tables from external tables are done by fetching from the binary dump files. Unloads from internal tables to external tables are done by populating the external tables' binary dump files.
- DEFAULT DIRECTORY - specifies the default location of files that are read or written by external tables. The location is specified with a directory object, not a directory path
- ACCESS PARAMETERS - describe the external data source and implements the type of external table that was specified. Each type of external table has its own access driver that provides access parameters unique to that type of external table
- LOCATION - specifies the location of the external data. The location is specified as a list of directory objects and filenames. If the directory object is not specified, then the default directory object is used as the file location

The following example shows the use of each of these attributes:

- CREATE TABLE emp_load (employee_number CHAR(5), employee_last_name CHAR(20), employee_first_name CHAR(15), employee_middle_name CHAR(15)) ORGANIZATION EXTERNAL (TYPE ORACLE_LOADER DEFAULT DIRECTORY ext_tab_dir ACCESS PARAMETERS (RECORDS FIXED 62 FIELDS (employee_number CHAR(2), employee_dob CHAR(20), employee_last_name CHAR(18), employee_first_name CHAR(11), employee_middle_name CHAR(11))) LOCATION ('info.dat'));

Example: Creating and Loading an External Table Using ORACLE_LOADER

The steps in this section show an example of using the ORACLE_LOADER access driver to create and load an external table. A traditional table named emp is defined along with an external table named emp_load. The external data is then loaded into an internal table.

1. Assume your .dat file looks as follows:

```
56    november, 15, 1980        baker     mary     alice
87    december, 20, 1970        roper     lisa     marie
```

1. Execute the following SQL statements to set up a default directory (which contains the data source) and to grant access to it:

```
CREATE DIRECTORY ext_tab_dir AS'/usr/apps/datafiles';
GRANT READ ON DIRECTORY ext_tab_dir TO SCOTT;
```

2. Create a traditional table named emp:

```
CREATE TABLE emp (emp_no CHAR(6), last_name CHAR(25), first_name CHAR(20), middle_initial CHAR(1));
```

3. Create an external table named emp_load:

```
CREATE TABLE emp_load (employee_number CHAR(5), employee_last_name CHAR(20), employee_first_name CHAR(15), employee_middle_name CHAR(15))
ORGANIZATION EXTERNAL
(TYPE ORACLE_LOADER DEFAULT DIRECTORY ext_tab_dir ACCESS PARAMETERS (RECORDS DELIMITED BY NEWLINE FIELDs (employee_number CHAR(2), employee_dob CHAR(20),employee_last_name CHAR(18),employee_first_name CHAR(11), employee_middle_name CHAR(11)))
LOCATION ('info.dat'));
```

4. Load the data from the external table emp_load into the table emp:

```
INSERT INTO emp (emp_no, first_name, middle_initial, last_name)
(SELECT employee_number, employee_first_name,
    substr(employee_middle_name, 1, 1),
     employee_last_name
FROM emp_load);
```

5. Perform the following select operation to verify that the information in the .dat file was loaded into the emp table:

```
SQL> SELECT * FROM emp;

EMP_NO LAST_NAME   FIRST_NAME     M
------ ----------- -------------------------  -----
56              baker       mary                a
87              roper       lisa                m
```

Important among them are

1. RECORD_FORMAT_INFO – optional
2. FIXED – specify fixed length
3. VARIABLE- indicates that each record is of different size

4. DELIMITED BY – indicates the character that terminates each record. The common one is , or |

5. LOAD WHEN – indicates the conditions that must be satisfied before a record can be loaded into a table.

LOAD WHEN (job !=MANAGER)

6. LOG FILE, BAD FILE AND DISCARD FILE – OPTIONAL

7. CONDITION – IT COMPARES ALL OR PART OF A FIELD AGAINST GIVEN CONIDITION

The main differences between SQL*Loader and External Tables are:

- When there are several input datafiles SQL*Loader will generate a bad file and a discard file for each datafile.
- The *CONTINUEIF* and *CONCATENATE* keywords are not supported by External Tables.
- The *GRAPHIC*, *GRAPHIC EXTERNAL*, and *VARGRAPHIC* are not supported for External Tables.
- *LONG*, nested tables, *VARRAY*, *REF*, primary key *REF*, and *SID* are not supported.
- For fields in External Tables the character set, decimal separator, date mask and other locale settings are determined by the database NLS settings.
- The use of the backslash character is allowed for SQL*Loader, but for External Tables this would raise an error. External Tables must use quotation marks instead.

For example:
SQL*Loader
FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"'
External Tables
TERMINATED BY ',' ENCLOSED BY '"'

- A second driver is available, the *ORACLE_DATAPUMP* access driver, which uses the Data Pump technology to read the table and unload data to an External Table. This driver allows the user to perform a logical backup that can later be read back to the database without actually loading the data. The *ORACLE_DATAPUMP* access driver utilizes a proprietary binary format for the external file, so it is not possible to view it as a flat file.
- Access Driver – ensures that the external data processing matches the description of the external table.

2 types of access drivers –

1. ORACLE_LOADER(only load data)
2. ORACLE_DATAPUMP(both load and unload data)

Populating External tables

Loading data – means reading data from an external table and loading it into a regular oracle table

Unloading data – means reading data from a regular oracle data and putting it into an external table.

When you load an oracle table from an external table, you use the INSERT INTO … SELECT clause

Eg:

```
Sql> INSERT INTO costs
(sale_date, produdt_id, unit_cost,unit_price)
SELECT sale_Date, product_id,
Sum(unit_cost), sum(unit_price)
 FROM sales_ext GROUP BY time_id, prod_id;
```

When you populate an external table using oracle tabledata(data unloading), use

CREATE TABLE … AS SELECT

Eg:

```
Sql> CREATE TABLE test_xt
ORGANIZATION EXTERNAL(TYPE ORACLE_DATAPUMP
DEFAULT DIRECTORY ext_data_dir
 LOCATION('text_xt.dmp'))
AS SELECT * FROM scott.dept;
```

Unloading Data Using the ORACLE_DATAPUMP Access Driver

- To unload data, you use the ORACLE_DATAPUMP access driver. The data stream that is unloaded is in a proprietary format and contains all the column data for every row being unloaded.
- An unload operation also creates a metadata stream that describes the contents of the data stream. The information in the metadata stream is required for loading the data stream. Therefore, the metadata stream is written to the datafile and placed before the data stream.

Parallel Access with ORACLE_LOADER

- The ORACLE_LOADER access driver attempts to divide large datafiles into chunks that can be processed separately.
- The following file, record, and data characteristics make it impossible for a file to be processed in parallel:
- Sequential data sources (such as a tape drive or pipe)
- Data in any multibyte character set whose character boundaries cannot be determined starting at an arbitrary byte in the middle of a string
- This restriction does not apply to any datafile with a fixed number of bytes per record.
- Records with the VAR format
- Specifying a PARALLEL clause is of value only when large amounts of data are involved.

Parallel Access with ORACLE_DATAPUMP

- When you use the ORACLE_DATAPUMP access driver to unload data, the data is unloaded in parallel when the PARALLEL clause or parallel hint has been specified and when multiple locations have been specified for the external table.
- Each parallel process writes to its own file. Therefore, the LOCATION clause should specify as many files as there are degrees of parallelism

External Table Restrictions

- An external table does not describe any data that is stored in the database.
- An external table does not describe how data is stored in the external source. This is the function of the access parameters
- An external table cannot load data into a LONG column.
- When identifiers (for example, column or table names) are specified in the external table access parameters, certain values are considered to be reserved words by the access parameter parser. If a reserved word is used as an identifier, it must be enclosed in double quotation marks.

Transforming Data

In most cases, especially in data warehousing environments, the data you're loading needs to be transformed to make it more meaningful for analysis. Oracle10g helps to efficiently transform the data within the database itself.The techniques used are:

1. Derive the tables from existing tables – use join or aggregations
2. Use SQL to transform data – MERGE, table functions, multiple table inserts.
3. Use oracle database 10g's – MODEL perform highly expressive computations using sets of interrelated formulas)

Deriving the data from Existing tables

2 methods

- 1. creating newly – use CTAS
- 2. table exists already – use **INSERT /*APPEND */ INTO … SELECT** method

Using CTAS method means u can create a new table from an existing table.

Example showing the use of CTAS method

```
Sql> CREATE TABLE new_employees
        AS  SELECT e.empno,INITCAP(e.ename), e.sal*1.1,
e.mgr,d.deptno,d.loc,d.dname
FROM emp e, dept d
WHERE e.deptno =d.deptno;
```

Table created.

Example showing how to load data into an existing table from another table.

```
Sql> INSERT/*APPEND NOLOGGING PARALLEL*/
INTO sales_data
SELECT product_id, customer_id, TRUNC(sales_date),
Discount_rate, sales_quantity, sale_price FROM sales_history;
```

NOLOGGING and PARALLEL makes the bulk run extremely fast.

You must use the following statement so any DML statements you issue can be considered for parallel execution:

```
Sql> ALTER SESSION ENABLE PARALLEL DML;
Session altered.
```

Using SQL to transform data

Common ways – MERGE statement, multitable inserts and table functions

Using MERGE statement

- It is a powerful means of transforming data because it provides the functionality of checking data to see if an update is indeed required for a given row.
- It is actually an UPDATE-ELSE-INSERT operation performed by a single SQL statement
- In the first pass, you update all rows(that have the matching) and in second pass, you insert all rows (that don't have the matching)

Eg:

```
Sql> UPDATE catalog c SET
(catalog_name, catalog_desc, catalog_category, catalog_price) =
SELECT (catalog_name, catalog_desc, catalog_category, catalog_price)
FROM catalog_data d
WHERE c.catalog_id=d.catalog_id;
```

Second, the insert

```
Sql> INSERT INTO  catalog c
SELECT * FROM catalog_data d
WHERE c.catalog_id NOT IN
(select catalog_id from catalog_data);
```

Syntax:

```
MERGE INTO <table/view>
USING <table/view/subquery> ON (condition)
--update
WHEN MATCHED THEN
UPDATE SET column = (value/expr),..
WHERE (condition) | DELETE WHERE (condition)
--insert
WHEN NOT MATCHED THEN
INSERT (column,..) VALUES (value/expr,..)
WHERE (condition)
```

Example: Create a target table called TEST1 and inset some test data

```
CREATE TABLE test1 (
co1 VARCHAR2(3),
co2 VARCHAR2(3),
co3 VARCHAR2(3),
co4 NUMBER );
ALTER TABLE test1 ADD CONSTRAINT test1_pk PRIMARY KEY(co1);
INSERT INTO test1 VALUES('1', 'val', 'sd1', 100);
INSERT INTO test1 VALUES('2', 'va2', 'sd2', 100);
INSERT INTO test1 VALUES('3', 'va3', 'sd3', 100);
INSERT INTO test1 VALUES('4', 'va4', 'sd4', NULL);
INSERT INTO test1 VALUES('5', 'va5', 'sd5', 100);
commit;
```

Example: Create a target table called TEST2 and inset some test data

```
CREATE TABLE test2 (
a1 VARCHAR2(3),
a2 NUMBER );

ALTER TABLE test2 ADD CONSTRAINT test2_pk PRIMARY
KEY(a1);
INSERT INTO test2 VALUES('1', 10);
INSERT INTO test2 VALUES('3', 20);
INSERT INTO test2 VALUES('6', 30);
INSERT INTO test2 VALUES('10', 40);
INSERT INTO test2 VALUES('15', 50);
INSERT INTO test2 VALUES('2', -10);
commit;
```

Now by using the MERGE statement we are going to update/delete or either insert records to the target table (test1) by using the source (test2).

```
1. MERGE INTO test1 a
2. USING (select a1, a2
3. from test2 ) b
4. ON (a.co1 = b.a1)
5. WHEN MATCHED THEN
6. update set a.co4 = a.co4 * b.a2,
7. a.co2 = b.a2
8. where co3 = \'sd3\'
9. delete where co3 = \'sd3\'
10. WHEN NOT MATCHED THEN
11. insert (a.co1, a.co2, a.co3, a.co4) values
12. (b.a1, null, null, b.a2);
```

Example

```
MERGE Students AS T USING NewYearRoster AS S
ON S.LastName = T. LastName and   S.FirstName = T.FirstName
WHEN MATCHED and T.Address <> S.Address THEN
  UPDATE SET T.Address = S.Address,  T.Age = S.Age
WHEN NOT MATCHED THEN
  INSERT (LastName,  FirstName,  Address,  Age)
        VALUES (S.LastName,  S.FirstName,  S.Address,  S.Age)
WHEN NOT MATCHED BY SOURCE THEN  DELETE;
```

{or}

You can use DELETE WHERE clause with any conditions also

Multitable INSERT in Oracle

Most INSERT statements are the single-table variety, but Oracle also supports a multiple-table INSERT statement. With a multitable insert, you can make a single pass through the source data and load the data into more than one table.

- *[ALL | FIRST]*
  WHEN condition THEN insert_into_clause [values_clause]
  [insert_into_clause [values_clause]]...
  [WHEN condition THEN insert_into_clause [values_clause]
  [insert_into_clause [values_clause]]...
  ]...
  [ELSE insert_into_clause [values_clause]
  [insert_into_clause [values_clause]]...
  ]

Simple multi-table insert

To begin, we will unconditionally INSERT ALL the source data into every target table. The source records and target tables are all of the same structure so we will omit the VALUES clause from each INSERT.

```
SQL> SELECT COUNT(*) FROM all_objects;
  COUNT(*)
----------
     28981
1 row selected.
SQL> INSERT ALL
  2    INTO t1
  3    INTO t2
  4    INTO t3
  5    INTO t4
  6  SELECT owner
  7  ,     object_type
  8  ,     object_name
  9  ,     object_id
```

```
 10 ,     created
 11  FROM   all_objects;

115924 rows created.
SQL> SELECT COUNT(*) FROM t1;
  COUNT(*)
----------
    28981
1 row selected.
```

Using table functions for data transformation

It produces a collection of transformed rows that can be queried just like a regular table's data. Table functions can take a set of rows as input and return a transformed set of rows.

Three features make table functions a powerful means of performing fast transformation of data sets

1. streaming – direct transmission of results from one process to the other without any intermediate steps.
2. parallel execution – concurrent execution of the functions on multiprocessor systems
3. pipelining – lets you see the results of a query iteratively, instead of waiting for the entire result set to be batched and returned.

Using Data Pump Export and Import

Introduction

Oracle Database 10*g* offers the new Data Pump technology, a server-side infrastructure for fast data movement between Oracle databases. The Data Pump technology enables DBAs to transfer large amounts of data and metadata at very high speeds compared with the older export/import technology. Data Pump manages multiple parallel streams of data to achieve maximum throughput. Data Pump is a superset of the original export and import utilities, offering several new capabilities. Data Pump lets you estimate job times, perform fine-grained object selection, monitor jobs effectively, and directly load a database from a remote instance via the network. Oracle Data Pump technology consists of two components: the Data Pump Export utility, to unload data objects from a database, and the Data Pump Import utility, to load data objects into a database.

Here's how you invoke the two utilities:

$ expdp *username*/*password* (*various parameters here*)

$ impdp *username/password* (*various parameters here*)

Benefits of Data Pump Technology

Advantages using data pump are

- ability to estimate jobs times
- ability to restart failed jobs
- perform fine-grained object selection
- monitor running jobs

- directly load a database from a remote instance via the network
- remapping capabilities
- improved performance using parallel executions

Data Pump Uses

You can use data pump for the following

- migrating databases
- copying databases
- transferring oracle databases between different operating systems
- backing up important tables before you change them
- moving database objects from one tablespace to another
- transporting tablespace's between databases
- reorganizing fragmented table data
- extracting the DDL for tables and other objects such as stored procedures and packages

Data Pump components

Data pump technology consists of three major components

- *dbms_datapump* - the main engine for driving data dictionary metadata loading and unloading
- *dbms_metadata* - used to extract the appropriate metadata
- *command-line* - expdp and impdp are the import/export equivalents

Data Access methods

Data pump has two methods for loading data, direct path or external table path you as a dba have no control with what data pump uses, normally simple structures such as heap tables without triggers will use direct path more complex tables will use the external path, oracle will always try and use the direct-path method.

Direct Path	bypasses the database buffer cache and writes beyond the high water mark when finished adjusts the high water mark, No undo is generated and can switch off redo as well, minimal impact to users as does not use SGA. Must disable triggers on tables before use.
External Path	Uses the database buffer cache acts as a SELECT statement into a dump file, during import reconstructs statements into INSERT statements, so whole process is like a normal SELECT/INSERT job. Both undo and redo are generated and uses a normal COMMIT just like a DML statement would.

In the following cases oracle will use the external path if any of the below are in use

- clustered tables
- active triggers in the table
- a single partition in a table with a global index
- referential integrity constraints
- domain indexes on LOB columns
- tables with fine-grained access control enabled in the insert mode
- tables with BFILE or opaque type columns

Data Pump files

You will use three types's of files when using data pump, all files will be created on the server.

- *dump files* - holds the data and metadata
- *log files* - the resulting output from the data pump command
- *sql files* - contain the DDL statements describing the objects included in the job but can contain data
- *Master data pump tables* - when using datapump it will create tables within the schema, this is used for controlling the datapump job, the table is removed when finished.

Data Pump privileges

In order to advance features of data pump you need *exp_full_database* and *imp_full_database* privileges.

How Data Pump works

The Master Control Process (MCP), has the process name DMnn, only one master job runs per job which controls the whole Data Pump job, it performs the following

- create jobs and controls them
- creates and manages the worker processes
- monitors the jobs and logs the process
- maintains the job state and restart information in the master table (create in the users schema running the job)
- manages the necessary files including the dump file set

The master process creates a master table which contains job details (state, restart info), this table is created in the users schema who is running the Data Pump job. Once the job has finished it dumps the table contents into the data pump file and deletes the table. When you import the data pump file it re-creates the table and reads it to verify the correct sequence in which the it should import the various database objects.

The worker process is named DWnn and is the process that actually performs the work, you can have a number of worker process running on the same job (parallelism). The work process updates the master table with the various job status.

The shadow process is created when the client logs in to the oracle server it services data pump API requests, it creates the job consisting of the master table and the master process.

The client processes are the expdp and impdp commands.

Performing Data Pump Exports and Imports

- The data pump export utility loads row data from database tables, as well as object metadata, into dump file sets in a proprietary format that only the dat pump import utility can read.
- Dump files usually refer to a single file such as the default export dump file exptdat.dmp

Data pump methods

1. Using the command line

eg: $ expdp system/manager DIRECTORY=dpump_dir1 DUMPFILE=expdat1.dmp

2. Using Parameter file

eg: SCHEMAS =HR
DIRECTORY=dpump_dir1
DUMPFILE=system1.dmp
SCHEMAS=hr

Once you create the parameter file, all you need to do in order to export the HR schema is invoke expdp with the PARFILE parameter

$ expdp PARFILE=myfile.txt

3. Using interactive datapump export- can use interactive mode for one purpose : to change some export parameters midstream while the job is still running. 1st way to perform is (Ctrl + c, which interrupts the running job and displays export prompt, you can insert the command). 2nd way is (using ATTACH command and parameter)

Data pump export modes

- Full export modes – export the entire database in one export session
- Schema modes – used only for a single user's data or objects only
- Tablespace mode – can export all the tables in one or more tablespaces
- Table mode – can export one or more tables

Data pump export parameters

1. **File and directory related parameters** – DIRECTORY, DUMPFILE, DILESIZE, PARFILE, LOGFILE, NOLOGFILE, NOLOGFILE, COMPRESSION

2. **Export mode related parameters** – FULL, SCHEMAS, TABLES, TABLESPACES, TRANSPORT_TABLESPACES, TRANSPORT_FULL_CHECK

3. **Export filtering parameters** –

(i) CONTENT – ALL, DATA_ONLY, METADATA_ONLY

(ii) EXCLUDE & INCLUDE – can perform metadata filtering, helps to omit/include specific database object types from an export or import operation.

- EXCLUDE =object_type(:name_clause)
- INCLUDE=object_type[:name_clause)
- EXCLUDE=TABLE: "LIKE 'EMP%' "

(iii) QUERY- it lets to selectively export table row data with the help of a SQL statement. SAMPLE parameter can alos be used. (lets you to specify a percentage value ranging from 0.000001 to 100)

Syntax:

SAMPLE =[[schema_name.]table_name:]sample_percent

Eg: SAMPLE="HR"."EMPLOYEE":50

4. **Estimation Parameters**- estimates how much physical space your export job will consume:

(i) ESTIMATE – tells you how much space ur new export job is going to consume. It is estimated in terms of bytes.

Syntax:

ESTIMATE ={BLOCKS | STATISTICS}

(ii) ESTIAMTE_ONLY – the estimate parameter is operative only during an actual export job, you can use the ESTIMATE_ONLY parameter without starting an export job.

5. **The Network Link parameter** – provides a way to initiate a network export. You can initiate an export job from your server and have data pump export data from a remote database to dump files located on the instance from which you initiate the Data pump export job - **ENCRYPTION parameter**.

6. **Job related parameters**

JOB_NAME,STATUS,FLASHBACK_SCN,FLASHBACK_TIME,
PARALLEL,ATTACH.

JOB_NAME is used to give an explicit name to the export job.

STATUS is useful while you are running the long jobs

FLASHBACK_SCN specifies the SCN that Datapump export wil use to enable the flashback utility

FLASHBACK_TIME –gives the time limit

PARALLEL – lets you specify more than a single active executive thread for your export job

ATTACH – attaches your data pump client session to a running job and palces you in an interactive mode.

Data pump import types and modes

You must have the IMPORT_FULL_DATABASE role in order to perform one of the following:

1. full database import
2. import of a schema other than your own
3. import of a table that you don't own.

Data pump import parameters

1. file and directory related parameters
2. filtering parameters
3. job related parameters
4. import mode related parameters
5. remapping parameters
6. the network link parameters
7. the transform parameters
8. the flashback parameters

1. File and directory related parameters

- it uses the PARFILE, DIRECTORY , DUMPFILE, LOGFILE AND NOLOGFILE and SQLFILE. The SQLFILE parameter is similar to the old import INDEXFILE parameter. When you perform a DATA pump Import job, you may sometimes wish to extract the DDL from the export dump file. The SQLFILE parameter enables you to do this easily.

2. Filtering parameters

- use CONTENT parameter
- CONTENT=DATA_ONLY
-CONTENT=ALL
-CONTENT=METADATA_ONLY

Use INCLUDE parameters to list the objects that you wish to import.Use the EXCLUDE parameters to list the objects you don't want to import

3. **Job related parameters** – JOB_NAME, STATUS PARALLEL

4. **Import mode related parameters**- can perform datapump import in various modes using TABLE, SCHEMAS, TABLESPACES, FULL, TRANSPORT_FULL_CHECK. TRANSPORT_DATAFILES import parameter is used during a transportable tablespaces operation to specify the list of data fiels the job should import into the target databases.

5. **Remapping parameters** – it remaps the objects during the data import process. The parameters are REMAP_SCHEMA, REMAP_DATAFILE and REMAP_TABLESPACE.

REMAP_SCHEMA – you can move objects from one schema to another.

$impdp system/manager DUMPFILE=newdump.dmp REMAP_SCHEMA=hr:oe

REMAP_DATAFILE – helps to change the filenames, when the databases are moved from one platform to another platform.

REMAP_TABLESPACE – enables you to move objects from one tablespace into a different tablespace into different tablespace during an import statement.

REMAP_DATAFILE – when you are moving databses between two different paltforms, each with a separate file-nameing convention, the REMAP_DATFILE parameter comes

6. **Transform parameter**- lets you specify that your datapup import job should not import certain stroage and other attributes. Using this, you can exclude the STORAGE and TABLESPACE clauses. Syntax:

TRANSFORM = transform_name:value[:object_type]

7. **Network Link Parameters** – enables the Datapump import utility to connect directly to the source database and transfer data to the target database.

Eg: 1. create a database link in the remote database

sql>create database link remote connect to system identified by sammy1 using 'remote.world';

2. If there isn't one already create a datapump directory object

Sql> create directory remote_dir1 AS '/u01/app/oracle/dp_dir';

3. set the new directory as your default directory, by exporting the directory value:

$export DATA_PUMP_DIR=remote=dir1

4. perform the network import from the database named remote, using the following data pump import command

[local]$impdp system/sammy1 SCHEMAS =scott NETWORK_LINK=remote.

8. **Flashback parameters** – enables you to import data at specified time. The FLASHBACK_SCN parameter is similar to the FLASHBACK_TIME parameter, except that you directly specify the SCN.

Monitoring a Data Pump Job

Two new views – DBA_DATAPUMP_JOBS and DBA_DATAPUMP_SESSIONS are crucial for monitoring data pump jobs.

Viewing Data Pump jobs – The DBA_DATAPUMP_JOBS view shows summary information of all currently running datapump jobs.

Eg:

```
Sql> select * from dba_datapump_jobs;
OWNER NAME      JOBNAME     OPERATION          JOB_MODE   STATE
        DEGREE              ATTACHED_SESSIONS
SYSTEM          SYS_EXPORT_FULL_01   EXPORT     FULL EXECUTING  1
                            1
```

It shows only the active jobs, a query on this view will reveal the value of the important JOB_NAME column for any job that is running right now.

The JOB_MODE – tale the values FULL, TABLE, SCHEMA or TABLESPACE,reflecting the mode of the currently executing export or the import job.

The STATE column can take the values UNDEFINED, DEFINING , EXECUTING and NOT RUNNING depending on which stage of the export or import you execute you query.

Viewing Data Pump Sessions – DBA_DATAPUMP_SESSIONS view identifies the user sessions currently attached to a Data Pump Export or import job.

Viewing Datapump job progress- In the V$SESSION_LONGOPS view, you can use the following four columns to monitor the progress of an export or import job:

(i) TOTAL WORK- shows the total estimated number of megabytes in the job.

(ii) SOFAR – shows the megabytes transferred thus far in the job

(iii) UNITS – stands for megabytes

(iv) OPNAME – shows the data pump job name

Eg: select opname, target_desc, sofar, totalwork from v$session_longops;

Transportable Tablespaces

Oracle's transportable tablespaces feature offers you an easy way to move large amounts of data between databases efficiently by simply moving data files from one database to the other. Transporting tablespaces involves copying all the data files belonging to the source database to the target database and importing the data dictionary information about the tablespaces from the source database to the target database.

Uses for Transportable Tablespaces

The following are some of the important uses of the transportable tablespaces feature:

- Moving data from a source database (usually OLTP) to a data warehouse
- Moving data from a staging database into a data warehouse
- Moving data from a data warehouse to a data mart
- Performing tablespace point-in-time recovery (PITR)
- Archiving historical data

Transporting a Tablespace

Transporting a tablespace between two databases involves the following main steps:

1. Select the tablespace to be transported (and make sure there are no dependencies with objects in other tablespaces).

2. Generate the transportable tablespace set.

3. Perform the tablespace import. This involves copying data files to the target server and importing related metadata into the target database.

Selecting the Tablespaces to Be Transported

The primary condition you must meet for transporting tablespaces is that the set of candidate tablespaces must be *self-contained.* For example, if the tables in the tablespaces have any indexes, they should be contained in one of the tablespaces in the set you're transporting. One way to verify that your set of tablespaces meets the self-contained criteria is by using the DBMS_TTS package, as follows:

SQL> EXECUTE sys.dbms_tts.transport_set_check('sales01,sales02',true);
PL/SQL procedure successfully completed.
You must have the EXECUTE_CATALOG_ROLE role to execute the TRANSPORT_SET_CHECK procedure.
You can further confirm this by querying the transport_set_violation table, which table lists all the partially contained tables in a tablespace and any references between objects belonging to different tablespaces.
SQL> SELECT * FROM sys.transport_set_violation
no rows selected

Generating the Transportable Tablespace Set

Before you can transport your tablespaces to the target database, you must generate a *transportable tablespace set.* The transportable tablespace set consists of all the data files in the tablespaces plus the export dump file, which contains the structural data dictionary information about the tablespaces. You can then transport this new tablespace to a different database.
SQL> ALTER TABLESPACE sales01 READ ONLY;
Tablespace altered.
SQL> ALTER TABLESPACE sales02 READ ONLY;
Tablespace altered.

Exporting the Dictionary Information (Metadata) for the Tablespaces

The first step in creating the transportable tablespaces set is to export the metadata that describes the objects that are part of the tablespaces you want to export. Listing 7 shows the export of the metadata for the pair of tablespaces.

Listing 7. *Exporting the Dictionary Metadata for the Tablespaces*
[finance] $ expdp oe/oe DIRECTORY=dpump_dir1 DUMPFILE=sales.dmp
TRANSPORT_TABLESPACES=sales01,sales02 INCLUDE=triggers,constraint,grant
Export: Release 10.2.0.0.0 - 64bit on Sunday, 29 May, 2005 14:34
Copyright (c) 2003, Oracle. All rights reserved.
Connected to: Oracle Database 10g Enterprise Edition Release 10.2.0.0.0 –
64bit Beta With the Partitioning, OLAP and Data Mining options
Starting "oe"."SYS_EXPORT_TRANSPORTABLE_01": oe/********
transport_tablespaces=sales01,sales02
include=triggers,constraint,grant directory=dpump_dir1 dumpfile=sales.dmp

```
Processing object type TRANSPORTABLE_EXPORT/TYPE/GRANT/OBJECT_GRANT
Master table "OE"."SYS_EXPORT_TRANSPORTABLE_01" successfully loaded/unloaded
******************************************************************************
***
Dump file set for OE.SYS_EXPORT_TRANSPORTABLE_01 is:
/u01/app/oracle/dba/sales.dmp
Job "OE"."SYS_EXPORT_TRANSPORTABLE_01" successfully completed at 14:36
oracle@finance.netbsa.org [/u01/app/oracle]
[finance] $
```

Copying the Export File and the Tablespace Files to the Target

The next step in generating the transportable tablespaces set is the physical copying of the data files contained in the tablespaces and the export dump file containing the metadata about the tablespaces to the target location.

Performing the Tablespace Import

Next, run the Data Pump Import utility (in the target database), which will plug in the tablespaces and incorporate information about them in the data dictionary of the target database. Because the export dump file doesn't have any data, all you'll be importing is the metadata about the objects.

Transporting Tablespaces across Platforms with Different Endian Formats

Endian format refers to the byte ordering of file systems. Endian format could be one of two types: big or little. If the endian formats of the source and target database are identical, everything you've seen up to now is all you'll need to do to transport the tablespaces. However, if the endian formats are different, you must convert the endian format of the source data files, either before or after transporting the data files to the target server.

Here are the steps:

1. Ensure the tablespaces are self-contained.
2. Make the tablespaces read-only.
3. Export the metadata using Data Pump Export.
4. Convert the data files to match the endian format.
5. Copy the files to the target system.
6. Use the Data Pump Import utility to import the metadata.

Managing and Monitoring the Operational Database

Types of Oracle Performance Statistics

Cumulative Statistics

Cumulative statistics are the accumulated total value of a particular statistic since instance startup.

Database Metrics

Database metrics are the statistics that measure the rate of change in a cumulative performance statistic.The background process MMON (Manageability Monitor) updates metric data on a minute-by-minute basis, after collecting the necessary fresh base statistics.

Sample Data

The new Automatic Session History (ASH) feature now automatically collects session sample data, which represents a sample of the current state of the active sessions.

Baseline Data

The statistics from the period where the database performed well are called baseline data.

Database Metrics

Database *metrics,* statistics that measure the *rate of change* in a cumulative performance statistic, are also important Oracle performance statistics. You can consider statistics such as the number of user transactions and the number of physical reads in the system as the *base statistics* from which database metrics are derived. The MMON (Manageability Monitor) background process updates metric data on a minute-by-minute basis
after collecting the necessary base statistics.
Oracle Database 10*g* uses several *metric groups*, with each group representing items like a wait event, service, or session. Table 4 lists the basic metric groups in Oracle Database 10*g*.

Table 4 *Oracle Database 10gMetric Groups*

Metric	**Description**
Event class metrics	Metrics collected at the wait event class level, such as DB_TIME_WAITING
Event metrics	Metrics collected on various wait events
File metrics long duration	Metrics collected at the file level, such as AVERAGE_FILE_WRITE_TIME
Service metrics	Metrics collected at the service level, such as CPU_TIME_PER_CALL
Session metrics short duration	Metrics collected at the session level, such as BLOCKED_USERS
System metrics long duration	Metrics collected at the system level
Tablespace metrics long duration	Metrics collected at the tablespace level, such as TABLESPACE_PCT_FULL

In-Memory Metrics

The MMON background process collects database metrics and saves them in the SGA for one hour. You can view system-related metrics by using views like V$SYSMETRIC_HISTORY and V$SYSMETRIC.

Here are some of the system metrics maintained in the V$SYSMETRIC view:

- Buffer cache hit ratio
- CPU usage per second
- Database CPU time ratio
- Database wait time ratio
- Disk sort per second
- Hard parse count per second
- Host CPU utilization percent
- Library cache hit ratio
- SQL service response time
- Shared pool free percent

Saved Metrics

Using the AWR snapshots, Oracle saves the metric information that is being continuously placed in the SGA by the MMON process. After saving performance metrics in memory for an hour, the MMON process flushes metric data from the SGA to disk, where they are stored permanently in the DBA_HIST_* views, such as DBA_HIST_SUMMARY_HISTORY, DBA_HIST_SYSMETRIC_HISTORY,
and DBA_HIST_METRICNAME. Each of these views actually represents snapshots of the corresponding V$ view, with, for example, the DBA_HIST_SYSMETRIC_HISTORY view containing snapshots of the V$SYSMETRIC_HISTORY view.

Server Generated Alerts

- Server generated alerts automatically alert you when a problem conditions occur.
- The database generates alerts based on the occurrence of specific events, or when certain database metrics exceed their threshold values.
- Oracle calls the threshold based alerts stateful alerts, and they can be set off at either a warning threshold or a critical threshold.

Examples for stateless alerts:

- 1. recovery area space usage exceeded
- 2. resumable session suspended
- Snapshot too old
- There are 3 situations when a database can send an alert
- 1. a metric crosses a critical threshold value
- A metric crosses a warning threshold values
- A nonthreshold type of alert occurs
- Default server generated alerts

The server generated default alerts could be either threshold based or problem alerts. Some of them are

1. Snapshot too old
2. Tablespace space usage
3. Resumable session suspended
4. Recovery session running out of free space

In addition to default alerts, u can choose to use other alerts, and u can also change the thresholds for the default alerts. When the database issues an alerts, you can see it in the database control alerts table, which is located at the bottom of the database control homepage. The alert data is, by default updated every 60 seconds. To get the details of an alert, click on the alert message in the message column of the alerts table.

Managing alerts

- best way to manage database alerts is to use the OEM database control.
- Oracle automatically sends an alert message to a persistent queue named ALERT_QUE and OEM reads this queue and sends out notification about the outstanding server alerts.
- It is very easy to set your own warning and critical thresholds for any database metric.
- To set alert thresholds go to database control home page and click the manage metrics link which will find under the related links groups.

For each metric on the edit thresholds page, you can set the following:

- 1. warning and critical threshold
- 2. Response action
- There are notification rules which enable you to control the conditions under which you want to receive a message from the OEM. (Preferences Link)

Using the DBMS_SERVER_ALERT package to manage alerts

- It provides an easy way to manage database alerts.
- It has two main procedures : GET_THRESHOLD
- SET_THRESHOLD-define threshold settings for a database metric.

Eg: SQL>DESC DBMS_SERVER_ALERT.SET_THRESHOLD
PROCEDURE dbms_server_alert.set_threshold

- DBMS_AQ & DBMS_AQADM packages can also be used to directly access and read alert messages.
- DBMS_AQADM packages lets you subscribe to the alert queue, set threshold and display alert modifications using various procedures.
- The DBMS_AQ packages lets you manage alert notifications.

Proactive tablespace alerts

- 10g tablespaces have built-in alerts that will notify you if there is free space drops below a set threshold.
- Two default thresholds are critical and warning
- The MMON background process monitors, the free space in each tablespace and sends out the alerts.
- Oracle will send alert with a warning when your tablespace is at 85% capacity and will send a critical alert when the tablespace is at 97% capacity.
- To view information on your thresholds, see the DBA_THRESHOLDS view.

Example:

1. create a small tablespace to use for testing oracle alert mechanism:

SQL>create tablespace test datafile 'test01.dbf' size 10M
Extend management local uniform size 3M;
Tablespace created

Data dictionary view related to metrics and alerts

- V$METRICNAME – show mapping of metric names to metric IDs
- V$ALERT_TYPES – dispalys information about server alert types
- DBA_HIST_SYSMETRIC_HISTORY – contains snapshots of V$SYSMETRIC_HISTORY
- DBS_ALERT_HISTORY – provides a history of alerts of alerts that are no longer outstanding that is, all alerts that you have already resolved
- DBA_THRESHOLDS – shows the names as well as the critical and warning values for all thresholds in the databases.

V$ALERT TYPES

- It provides information about all system alert types.

1. STATE – stateful or stateless alerts. Stateful alerts are those alerts that clear automatically when the alert threshold that prompted the alert is cleared. It first appears in the DBA_OUTSTANDING_ALERTS view and then goes to DBA_ALERT_HISTORY Whereas the stateless alert goes straight to DBA_ALERT_HISTORY.
2. SCOPE – classifies alerts into database-wide and instance –wide
3.GROUP_NAME – oracle aggregates the various database alerts into some common groups:space, performance and configuration.

DBA_THRESHOLDS

- it provides the current threshold settings for all alerts.. It is useful when we want to find out the current threshold settings for any alert.

SQL> SELECT metric_name, warning_value, critical_value, consecutive_occurences from DBA_THRESHOLDS where metrics_name LIKE '%CPU Time%'

The Automatic Workload Repository (AWR)

AWR automatically collects and stores database performance statistics relating to problem detection and tuning, and it lies at the heart of the new database self-tuning mechanism. It generates snapshots of key performance data, such as system and session statistics, segment-usage statistics, time-model static's, and high load SQL statistics and it stores the snapshots in the sysaux tablespace.

It provides performance statistic in two formats:

- 1. A temporary in memory collection of statistics in the SGA
- 2. A persistent type of performance data in the form of regular AWR snapshots.

AWR Features

The AWR is used to collect performance statistics including:

- Wait events used to identify performance problems.
- Time model statistics indicating the amount of DB time associated with a process from the V$SESS_TIME_MODEL and V$SYS_TIME_MODEL views. Active Session History (ASH) statistics from the V$ACTIVE_SESSION_HISTORY view.
- Some system and session statistics from the V$SYSSTAT and V$SESSTAT views.
- Object usage statistics.
- Resource intensive SQL statements.

AWR Data Handling

It is important to understand that the AWR isn't a permanent repository for oracle performance statistics.

The space used by the AWR depends on the following:

1. Data retention period- longer period, more space
2. Snapshots interval – more frequently used, more space
3. Number of active sessions – higher number of user sessions, the more data is collected.

By default, AWR saves the data for a period of seven days, but you can modify this period.

Managing AWR

Snapshots provide you values for key performance statistics at a given point in time. The default interval for snapshot collection is 60 minutes, and the minimum interval is 10 minutes. You can manage the AWR snapshots either with the help of the OEM database control or with the oracle supplied DBMS_WORKLOAD_REPOSITORY package.

Snapshots

To create snapshot manually, use the CREATE_SNAPSHOT procedure, as follows:

```
SQL> BEGIN
```

```
dbms_workload_repository.create_snapshot();
        END;
```

By default snapshots of the relevant data are taken every hour and retained for 7 days. The default values for these settings can be altered using:

```
SQL>BEGIN  DBMS_WORKLOAD_REPOSITORY.modify_snapshot_settings(  retention
=> 43200, -- Minutes (= 30 Days). Current value                retained if NULL.
    interval => 30); -- Minutes. Current value retained if NULL.
 END;
```

Extra snapshots can be taken and existing snapshots can be removed using:

```
EXEC DBMS_WORKLOAD_REPOSITORY.create_snapshot;
BEGIN
DBMS_WORKLOAD_REPOSITORY.drop_snapshot_range  (  low_snap_id  =>  22,
high_snap_id => 32);
END; /
```

Snapshot information can be queried from the DBA_HIST_SNAPSHOT view

Using database control to manage AWR snapshots

Click Admin -> automatic workload repository link.
There are 2 sections:

1. General section and manage snapshots
2. Preserved snapshot sets sections

In the general section you can edit

- Snapshots retention intervals
- Snapshots collection intervals
- Snapshot collection level

From manage snapshots page, you can do the following:

- Create a snapshot spontaneously
- View a list of the snapshots collected over a specific period
- Establish a range of snapshots to use as a baseline
- Delete a defined range of snapshots from the list of snapshots collected over a period of time
-

Creating and deleting AWR snapshot baseline

The purpose of using snapshot baselines is to have a valid measuring stick for acceptable database performance.

A baseline is a pair of snapshots that represents a specific period of usage. Once baselines are defined they can be used to compare current performance against similar periods in the past. You may wish to create baseline to represent a period of batch processing like:

```
BEGIN    DBMS_WORKLOAD_REPOSITORY.create_baseline ( start_snap_id => 210,
   end_snap_id => 220,
   baseline_name => 'batch baseline');
 END; /
```

The pair of snapshots associated with a baseline are retained until the baseline is explicitly deleted:

```
BEGIN
  DBMS_WORKLOAD_REPOSITORY.drop_baseline (
        baseline_name => 'batch baseline',
              cascade => FALSE); -- Deletes associated
        snapshots if TRUE.
END; /
```

Baseline information can be queried from the DBA_HIST_BASELINE view.

Workload Repository Views

The following workload repository views are available:

- V$ACTIVE_SESSION_HISTORY - Displays the active session history (ASH) sampled every second.
- V$METRIC - Displays metric information.
- V$METRICNAME - Displays the metrics associated with each metric group.
- V$METRIC_HISTORY - Displays historical metrics.
- V$METRICGROUP - Displays all metrics groups
- DBA_HIST_ACTIVE_SESS_HISTORY - Displays the history contents of the active session history.
- DBA_HIST_BASELINE - Displays baseline information.
- DBA_HIST_DATABASE_INSTANCE - Displays database environment information.
- DBA_HIST_SNAPSHOT - Displays snapshot information.
- DBA_HIST_SQL_PLAN - Displays SQL execution plans.
- DBA_HIST_WR_CONTROL - Displays AWR settings.

Workload Repository Reports

Oracle provides a script named awrrpt.sql to generate summary reports about the statistics collected by the AWR facility. when you run the awrrpt.sql script, you'll need to make the following choices:

- Choose between an HTML or plain text report
- Specify the beginning and ending snap id's

The two reports give essential the same output but the awrrpti.sql allows you to select a single instance. The reports can be generated as follows:
@$ORACLE_HOME/rdbms/admin/awrrpt.sql
@$ORACLE_HOME/rdbms/admin/awrrpti.sql

The AWR reports include voluminous information, including the following:

- Load profile
- Top five timed events
- Wait events and latch activity
- Time-model statistics
- Operating system statistics
- Operating system statistics
- SQL ordered by elapsed time
- Tablespace and file I/O statistics
- Buffer pool and PGA statistics and advisors

Active Session History

It is a new source of Oracle database performance data in 10*g*.

- An *active session* is one which is in a user call
  - Parse
  - Execute
  - Fetch
  - On the CPU
- Provides historical information about recently *sampled* "active" sessions
- ASH = V$SESSION_WAIT++ with History
  - Note: In 10*g* V$SESSION_WAIT is integrated with V$SESSION
- It facilitates *spot analysis* of both foreground and background sessions

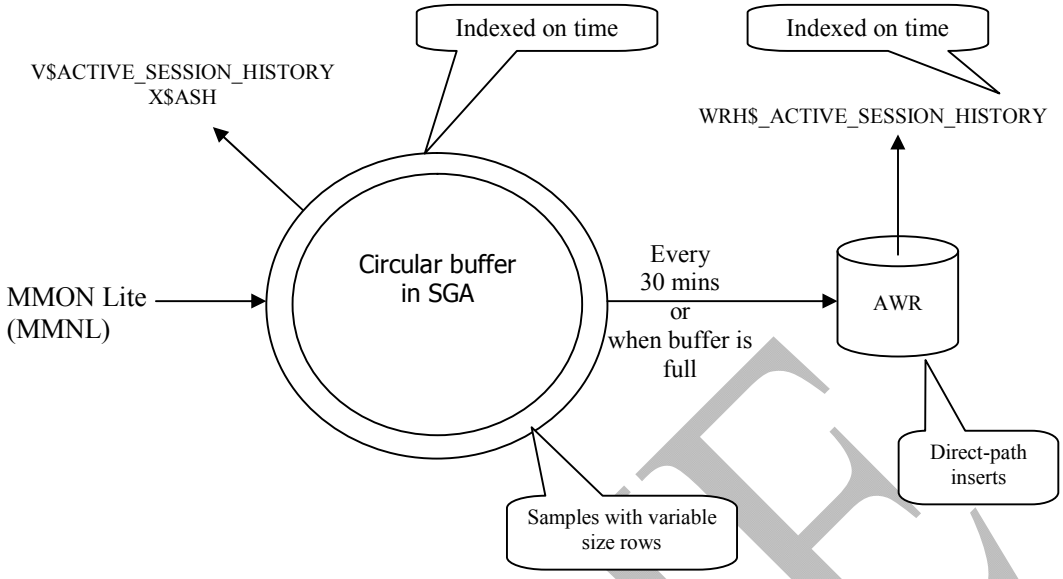
Why should you use ASH?

- Great for performance diagnostics
  - Logs wait events along with SQL details and session-specific context in a circular buffer in memory
- Fixed session sampling algorithm uses < 0.1% of 1 CPU
  - Can be modified by the use of an _ parameter
- Primary data provider for the Automatic Database Diagnostic Monitor (ADDM)
  - ADDM supports proactive performance diagnostics within the Oracle Kernel
  -

Components of ASH

- Memory buffers in the fixed areas
- New Oracle Background Process
  - MMNL – MMON Lite
- V$ACTIVE_SESSION_HISTORY
- X$ASH
- DBA_HIST_ACTIVE_SESS_HISTORY
  - Based on WRH$_ACTIVE_SESSION_HISTORY

ASH Architecture



ASH Details - General

- No installation or setup required
- Intended 30-min circular buffer in the SGA
- In memory ASH contains as much history as it can store.
  - Circular buffer not cleared when written to disk
- ASH on Disk (1 of 10 in memory samples)
- Init.ora
  - STATISTICS_LEVEL = TYPICAL (Default)
- Master Switch
  - _ACTIVE_SESSION_HISTORY = TRUE (Default)
- 30-minute circular buffer in the SGA - GOAL
  - May scale down to smaller duration on large systems
- Circular Buffer Sizing Formula:
  Max(Min (# of CPUs * 2MB, 5% of SHARED_POOL_SIZE, 30MB), 1MB)
- If SHARED_POOL_SIZE is not explicitly set
  - Formula changes to 2% of SGA target

- Assumptions for MAX Size - 30MB
  - 100 active sessions
  - Sampled at once per second (60 samples in 1 minute)
  - Assume 17 minutes of non-stop collection
  - Assume 300 bytes per sample
  - Size = 100*60*17*300 bytes ~ 29.18MB
  - Fudge Factor of 0.82 MB
- History flushed to Automatic Workload Repository (AWR) every 30 minutes

- Part of the AWR snapshot
  - Database metrics
  - Session Wait Information
- Sampling done every second
  - Can support sub-second sampling
  - _ash_sampling_interval = 1000 (milliseconds by default)
- Can dump to process trace (if required)
- Estimated 2500 CPU Instructions per active session per sample
  - 400 active sessions on a 1 Ghz processor consumes < 1 millisecond
- The sampler (MMNL) does not take any latches
- It supports dirty reads
- Can write to the in-memory buffer without any issues

ASH Details – View Describe

SQL> desc v$active_session_history

Name	Null?	Type
SAMPLE_ID		NUMBER
SAMPLE_TIME		TIMESTAMP(3)
SESSION_ID		NUMBER
SESSION_SERIAL#		NUMBER
USER_ID		NUMBER
SQL_ID		VARCHAR2(13)
SQL_CHILD_NUMBER		NUMBER
SQL_PLAN_HASH_VALUE		NUMBER
SQL_OPCODE		NUMBER
SERVICE_HASH		NUMBER

ASH reporting

You can use the ashrpt.sql script, located in the $ORACLE_HOME/rdbms/admin/directory to get an ASH report.

The script used is similar to AWR.

To run the ashrpt.sql script to get an ASH report:

$ ORACLE_HOME/rdbms/admin/ashrpt.sql

Working with the Undo and the MTTR Advisors

Using the Undo Advisor

You can get to the Undo Advisor by following these steps:

1. From the Database Control home page, click on the Advisor Central link.

2. On the Advisor Central page, click on the Undo Management link.

3. Click the Undo Advisor button at the top of the page.

Using the Undo Advisor, you can do the following:

• Set the low threshold value for undo retention
• Figure out the size of the undo tablespace size you'll need for a new undo retention setting
• Use different analysis time periods representing different levels of system activity to get recommendations, in the form of a graph, about the right undo tablespace size for varying undo retention length

The MTTR Advisor

To control database recovery time, you use the FAST_START_MTTR_TARGET initialization parameter to set the mean time to recover (MTTR) from a crash.
You can access the MTTR Advisor through the Database Control, as follows:
1. From the Database Control home page, click on the Advisor Central link under the Related Links section.
2. Click on the MTTR Advisor link under the Advisors group.

You can do the following things with the help of the MTTR Advisor:
• Look up the tradeoff between a certain MTTR and total I/O in the Instance Recovery section.
• Turn archive logging on and off and enable and disable the automatic archiving of the redo logs, both through the Media Recovery section.
• Manage the flash recovery area, including its location and size, and enable and disable Flashback Database logging for fast database point-in-time recovery. You do all this from the Flash Recovery Area section.

POSSIBLE QUESTIONS:

PART B: 6 MARKS

1. Elucidate the process involved in loading data using external tables.
2. Describe in detail about transforming data.
3. Discuss in detail the usage of data pumps for export and imports.
4. Write a note on Transportable Tablespaces.
5. Discuss the usage of automatic workload repository
6. List out the types of Oracle Performance Statistics
7. What is Active Session History? Explain.
8. Write short notes on undo and MTTR advisors.
9. What are server generated alerts? Give example.
10. How do you monitor a data pump? Explain.

Karpagam Academy of Higher Education
Department of CS, CA & IT
Subject: Oracle 10g Administration (17CSP203)

Batch : 2017-2019 **Class: I M.Sc CS**

Objective Type Questions
UNIT III

S.NO	QUESTIONS	OPTION 1	OPTION 2	OPTION 3	OPTION 4	KEY
1	_____ is the process of placing the data in the database tables.	loading	extraction	transformation	insertion	loading
2	_____ provides a way to merge the loading and transformation processes	multitable inserts	external tables	upserts	table functions	external tables
3	_____ produce a set of rows as output.	multitable inserts	external tables	upserts	table functions	table functions
4	The term _____ implies that a given table structure is mapped to a data file that's located in an operating system file	multitable inserts	external tables	upserts	table functions	external tables
5	Using _____ driver, you can only load data into a table from an external table.	ORACLE_LOADER	ORACLE_DATAPUMP	ORACLE_PUMP	ORACLE_OBJECT	ORACLE_LOADER
6	The _____ access driver lets you both load and unload data using external dump files	ORACLE_LOADER	ORACLE_DATAPUMP	ORACLE_PUMP	ORACLE_OBJECT	ORACLE_DATAPUMP
7	_____ data means reading data from an external table and loading it into a regualr oracle table.	loading	unloading	copying	inserting	loading
8	Using _____ statement , it can perform highly expressive computations using sets of interrelated formulas	MODEL	EXTRACT	SQL	DML	MODEL
9	_____ refers to the direct transmission of results from one process to the other without any intermediate steps.	parallel execution	pipelining	streaming	loading	streaming
10	You can use the DBMS_ADVISOR package to create an ADDM report by using the ___ procedure.	create_report	create_task	create_work	create_load	create_report
11	Data Pump Export is invoked with the_____ command	impdp	expdp	exp	export	expdp
12	Data Pump Import is invoked with the____ command	impdp	import	impdp	import dump	impdp

13	The_____ technology enables very high transfer of data from one database to another	data pump	data load	loader	directory object	data pump
14	Data pump is a_____ technology	client side	server side	complier	interface	server side
15	_____ the main engine for driving data dictionary metadata loading and unloading	dbms_pump	dbms_datapump	dbms_data	dbms_load	dbms_datapump -
16	_____used to extract the appropriate metadata	dbms_pump	dbms_datapump	dbms_metadata	dbms_load	dbms_metadata
17	_____lets oracle read data from and write data to operating system files that lie outside the database	direct path	external path	external loader	internal path	external path
18	_____ holds the data and metadata	dump files	sql files	log files	redo files	dump files
19	_____contain the DDL statements describing the objects included in the job but can contain data	dump files	sql files	log files	redo files	sql files
20	A _____maps a name to a directory path on the file system.	index	command	directory object	execute	directory object
21	The _____ Process , creates jobs and monitors them.	Master Control	slave	shadow	client	Master Control
22	The _____ process updates the master table with the various job status	Master Control	slave	shadow	work	work
23	The _____processes are the expdp and impdp commands	Master Control	slave	client	work	client
24	Interactive datapump export can be performed by using _____ command	ATTACH	ALTER	ALIGN	ARRANGE	ATTACH
25	In _____ modes datapump exports the entire database in one export session	schema modes	Full export modes	table mode	tablespace modes	Full export modes
26	_____ mode is used only for a single user's data or objects only	schema modes	Full export modes	table mode	tablespace modes	schema modes
27	_____ parameter lets you specify more than a single active executive thread for your export job	parallel	orderby	sortby	linear	parallel
28	Use _____parameters to list the objects that you wish to import	EXCLUDE	IMPORT	INCLUDE	INTER	INCLUDE
29	The_____ view shows summary information of all currently running datapump jobs.	DBA_DATAPUMP	DBA_DATAPUMP_JOBS	DBA_DATAPUMP_FILES	DBA_DATAPUMP_TABLES	DBA_DATAPUMP_JOBS

30	_____ view identifies the user sessions currently attached to a Data Pump Export or import job	DATAPUMP_SESSIONS	DBA_SESSIONS	DBA_ACTIVE_Sessions	DBA_DATAPUMP_SESSIONS	DBA_DATAPUMP_SESSIONS
31	_____ facilitates spot analysis of both foreground and background sessions	ASH	AWR	ARR	ARW	ASH
32	The _____ package lets you transfer operating system files directly through the database	DBMS_TRANSFER	DBMS_FILE_TRNSFER	DBMS_FULL_FILE	DBMS_TRANSFER_FILE	DBMS_FILE_TRNSFER
33	_____ are the accumulated total value of particular statistics since the start of an oracle isntance	sample data	baseline data	cumulative statistics	database metrics	cumulative statistics
34	The ASH automatically collects the _____	session sample data	baseline data	cumulative statistics	database metrics	session sample data
35	The statistics from the period when the database performed well are called _____	session sample data	baseline data	cumulative statistics	database metrics	baseline data
36	_____ measures the rate of change in a cumulative performance statistics	session sample data	baseline data	cumulative statistics	database metrics	database metrics
37	_____ procedure is used to define threshold settings for a database metric	SET_THRESHOLG	GET_THRESHOLD	GIVE_THRESHOLD	PASS_THRESHOLD	SET_THRESHOLG
38	The _____ package lets you manage alert notifications	DBMS_QUERY	DBMS_AQADM	DBMS_ASH	DBMS_ALERT	DBMS_AQADM
39	_____ shows the mapping of metric names to metric IDs	V$ALERT_TYPES	V$METRICNAME	V$DATABASE	V$DBNAME	V$METRICNAME
40	The _____ view provides information about all system alert types.	V$ALERT_TYPES	V$METRICNAME	V$DATABASE	V$DBNAME	V$ALERT_TYPES
41	The _____ automatically collects and stores database performance statistics	ASH	AWR	ATH	SGA	AWR
42	Expand MMNL	manageability monitor light	manageability monitor link	manageability movement link	manageability monitor loader	manageability monitor light
43	The _____ view is where the database stores a sample of all active session data	V$ALERT_TYPES	V$METRICNAME	V$ACTIVE_SESSION_HISTORY	V$DBNAME	V$ACTIVE_SESSION_HISTORY
44	_____ script is used to produce a ASH report	ashrpt.sql	ashrrt.sql	ashreport.sql	ashret.sql	ashrpt.sql

45	The _____ ensures that the external data processing matches the description of the external table.	access driver	directory objects	tables	functional mode	access driver
46	The older export/import technology was ____ .	client-based	server-based	DBMS_STATS	terminate	client-based
47	___ is / are the components of data pump technology. .	dbms_datapump package	dbms_metadata package	cluster synchronization service	terminate	terminate
48	Data pump operations uses _____ files.	dump files	log	abort immediate	drop diskgroup	drop diskgroup
49	Data pump job creates all its dump files on the _____ .	server machine	client machine	abort immediate	remote machine	server machine
50	_____ command causes all the open Oracle connections to terminate automatically.	abort immediate	terminate	shutdown abort	close imemdiate	shutdown abort
51	_____ are suitable for large data loads that may have a onetime use in your database	external tables	local tables	global tables	unique tables	external tables
52	_____ data means reading data from a regular oracle table and putting it into an external table.	loading	unloading	moving	replacing	unloading
53	Expand MCP	Master Control Process	Main Control Process	Memory Control Process	Management Control Process	Master Control Process
54	_____mode can export one or more tables	full mode	table mode	tablespaces mode	schema mode	table mode
55	_____format refers to the byte ordering of file systems.	ASCII	Endian	Excess 3 Code	Unary	Endian
56	Expand ASH.	Automatic Space History	Automatic Session Parameters	Automatic Session History	Automatic Space Parameters	Automatic Session Parameters

UNIT IV

SYLLABUS:

User management and database security: managing users - the database resource manager - controlling access to data - auditing database usage - authenticating users -enterprise user security - database security do's and don'ts. Backing up databases: backing up oracle databases - the recovery manager - backing up control file - oracle back up tool – user managed backups - database corruption detection - enhanced data protection for disaster recovery. Database recovery: types of database failures - oracle recovery processes - performing recovery with RMAN - media recovery scenarios.

User Management and Database Security

These are the main aspects of Oracle database security management:

- Controlling access to data (authorization)
- Restricting access to legitimate users (authentication)
- Ensuring accountability on part of the users (auditing)
- Safeguarding key data in the database (encryption)
- Managing the security of the entire organizational information structure (enterprise security)

Managing Users

User management is a pretty complex topic because not only does it deal with authorizing users to use the database, but it also touches on vital topics such as security and resource management. When you create a new database, the only users at first will be the application or schema owners. Later on, you'll create the actual end users who will be using the database on a day-to-day basis.

Temporary and Default Tablespaces

All users need a *temporary tablespace* where they can perform work such as sorting data during SQL execution. Users also need to have a *default tablespace*, where their objects will be created if they don't explicitly assign a different tablespace during object creation.

In Oracle Database 10*g*, you can create a default temporary tablespace and a default permanent tablespace for all users during the database-creation process.

Creating a New User

Database USER account is created with CREATE USER statement. You must assign a unique username and authentication method. Additional attributes (Optional) can be set to the user account with the CREATE USER statement.For example, to create a password authenticated user named rajesh with a password of welcome, you execute the following:

CREATE USER rajesh IDENTIFIED BY welcome;

The keywords IDENTIFIED BY password (in this case, password is welcome) tell the database that this user account is a password authenticated account.

Assigning a Default Tablespace

If you execute a CREATE TABLE statement and do not explicitly specify a tablespace, the database uses your default tablespace. If you do not explicitly assign a default tablespace to a user at the time you create the user, the database assigns the database's default tablespace to the new user account. Use the keywords DEFAULT TABLESPACE tablespace_name to assign a default tablespace to either a new user via a CREATE USER statement or an existing user, like this:

CREATE USER rajesh IDENTIFIED BY welcome DEFAULT TABLESPACE users;

Or via an ALTER USER statement:

ALTER USER rajesh DEFAULT TABLESPACE users;

To change the database default tablespace (the value that users inherit if no default tablespace is provided), use the ALTER DATABASE statement, like this:

ALTER DATABASE DEFAULT TABLESPACE users;

Assigning a Temporary Tablespace

If you do not explicitly assign a temporary tablespace at user creation time, the database assigns the database default temporary tablespace to the new user account. Use the keywords TEMPORARY TABLESPACE tablespace_name to assign a temporary tablespace either to a new user via the CREATE USER statement:

CREATE USER rajesh IDENTIFIED BY welcome DEFAULT TABLESPACE users TEMPORARY TABLESPACE temp;

Or to an existing user via an ALTER USER statement: ALTER USER rajesh TEMPORARY TABLESPACE temp;

Removing a User from the Database

You use the DROP USER statement to remove a user from the database. You can optionally include the keyword CASCADE to tell the database to recursively drop all objects owned by that user. To drop both user rajesh and all objects he owns, execute the following:

DROP USER rajesh CASCADE;

Assigning a Profile to a User

You can set the individual resource limits in Oracle by using what are known as *profiles.* A profile is a collection of resource-usage and password-related attributes that you can assign to a user. Multiple users can share the same profile, and you can have an unlimited number of profiles in an Oracle database.

Here, for example, is a profile called "miser" (because it limits resource usage to a minimum):

```
SQL> CREATE PROFILE miser
2 LIMIT
3 connect_time 120
4 failed_login_attempts 3
5 idle_time 60
6* sessions_per_user 2;
```

Profile created.

Profile Parameters and Limits

The following sections provide brief explanations of these parameters, which can be divided into two broad types: *resource parameters,* which are concerned purely with limiting resource usage, and *password parameters,* used for enforcing password-related security policies.

Resource Parameters

The main reason for using resource parameters is to ensure that a single user or a set of users doesn't monopolize the database and server resources. Here are the most important resource parameters that you can set within an Oracle Database 10*g* database:

• CONNECT_TIME: Specifies the total time (in seconds) a session may remain connected to the database.

• CPU_PER_CALL: Limits the CPU time used per each call within a transaction (for the parse, execute, and fetch operations).

• CPU_PER_SESSION: Limits the total CPU time used during a session.

• SESSIONS_PER_USER: Specifies the maximum number of concurrent sessions that can be opened by the user.

• IDLE_TIME: Limits the amount of time a session is idle (which is when nothing is running on its behalf).

Password Parameters

Oracle provides you with a wide variety of parameters to manage user passwords. You can set the following password-related profile parameters to enforce your security policies:

• FAILED_LOGIN_ATTEMPTS: Specifies the number of times a user can attempt to log in before being locked out.

• PASSWORD_LIFE_TIME: Sets the time limit for using a particular password. If you don't change the password within this specified time, the password expires.

PASSWORD_GRACE_TIME: Sets the time period during which you'll be warned that your password has expired. After the grace period is exhausted, you can't connect to the database with that password.

• PASSWORD_LOCK_TIME: Specifies the number of days a user will be locked out after reaching the maximum number of unsuccessful login attempts.

• PASSWORD_REUSE_TIME: Specifies the number of days that must pass before you can reuse the same password.

The Default Profile

If you create a user and don't explicitly assign any profile to the
user, the user will inherit the *default* profile, as shown here:

```
SQL> SELECT profile FROM dba_users
WHERE username = 'SALAPATI'
PROFILE
-----------
DEFAULT
```

Assigning a User Profile

You can assign a user a profile when you create the user. Here's an example:

```
SQL> CREATE USER salapati IDENTIFIED BY sammyy1
TEMPORARY TABLESPACE TEMPTBS01
DEFAULT TABLESPACE USERS
GRANT QUOTA 500M ON USERS;
```

PROFILE 'prod_user';
User created.

You can also assign a profile to a user any time by using the ALTER USER statement, as shown here:

```
SQL> ALTER USER salapati
2 PROFILE test;
User altered.
```

Altering a User Profile

You can alter a profile by using the ALTER PROFILE statement, as follows:

```
SQL> ALTER PROFILE test
2 LIMIT
3 sessions_per_user 4
4* failed_login_attempts 4;
Profile altered.
```

Dropping a User Profile

Dropping a profile is straightforward. Here's how you would drop the test profile:

```
SQL> DROP PROFILE test CASCADE;
Profile dropped.
```

Managing Resources

With large numbers of database users, resource management becomes an important issue. Server resources are ultimately limited, and you must have some means of apportioning the scarce resources among the users. Oracle provides a powerful tool, the Database Resource Manager, which allows you to control database resource usage in a sophisticated manner. User profiles are effective in controlling the resource usage of individual users, but Oracle prefers that you use profiles mainly for password management. Oracle recommends using the Database Resource Manager to control resource usage.

The Database Resource Manager

The Oracle Database Resource Manager is the answer—it allows you to create resource plans, which specify how much of your resources should go to the various consumer groups. You can group users based on their resource requirements, and you can have the Database Resource Manager allocate a preset amount of resources to these groups. Using the Database Resource Manager, it's possible for you to ensure that your critical user groups (formally referred to as *resource consumer groups*) are always guaranteed enough resources to perform their tasks. The Database Resource Manager also enables you to limit the length of time a user session can stay idle and to automatically terminate long-running SQL statements and user sessions.

The following four elements are integral to the Database Resource Manager:

• *Resource consumer group*: A resource consumer group is used to group together similar users based on their resource needs.

• *Resource plan*: The resource plan lays out how resource consumer groups are allocated resources.

• *Resource allocation method*: The resource allocation method dictates the specific method you choose to use to allocate resources like the CPU. These are the available methods of allocating database resources:

• *CPU method*: Oracle uses multiple levels of CPU allocation to prioritize and allocate CPU usage among the competing user sessions.

• *Idle time*: You can direct that a user's session be terminated after it has been idle for a specified period of time. You can also specify that only idle sessions blocking other sessions be terminated.

• *Execution time limit*: You can control resource usage by setting a limit on the maximum execution time of an operation.

• *Undo pool*: By setting an undo pool directive, you can limit the total amount of undos that can be generated by a consumer resource group.

• *Active session pool*: You can set a maximum allowable number of concurrent sessions within any consumer resource group. All sessions that are beyond the maximum limit are queued for execution after the freeing up of current active sessions.

• *Automatic consumer group switching*: Using this method, you can specify that a user session be automatically switched to a different group after it runs more than a specified number of seconds. The group the session should switch to is called the switch group, and the time limit is the switch time. The session can revert to its original consumer group after the end of the top call, which is defined as an entire PL/SQL block or a separate SQL statement.

• *Canceling SQL and terminating sessions*: By using CANCEL_SQL or KILL_SESSION as the switch group, you can direct a long-running SQL statement or even an entire session to be canceled or terminated.

• *Parallel degree limit*: You can use this method to specify the limit of the degree of parallelism for an operation.

• *Resource plan directive*: The resource plan directive links a resource plan to a specific resource consumer group.

Using the Database Resource Manager

DBMS_RESOURCE_MANAGER package.

```
SQL> EXEC DBMS_RESOURCE_MANAGER_PRIVS.GRANT_SYSTEM_PRIVILEGE -
(GRANTEE_NAME => 'scott', PRIVILEGE_NAME => '
ADMINISTER_RESOURCE_MANAGER');
```

Here are the steps you need to follow to start using the Database Resource Manager:

1. Create a pending area. This is the work area where you create and validate resource consumer groups, resource plans, and plan directives.

2. Create a resource consumer group. This is a grouping of users who will receive the same amount of resources.

3. Create a resource plan. This is a collection of directives that specify how Oracle should allocate resources to resource consumer groups.

4. Create a plan directive. This associates resource consumer groups with resource plans and allocates resources among resource consumer groups.

5. Validate the pending area. This process validates the resource consumer group, the resource plan, and the plan directive.

6. Submit the pending area. This creates the resource consumer group, the resource plan, and the plan directives, and makes them active.

Creating a Pending Area

Create a *pending area* to validate changes before their implementation. The pending area serves as a work area for your changes. Here's how you create the pending area:

```
SQL> EXECUTE dbms_resource_manager.create_pending_area;
PL/SQL procedure successfully completed.
```

Creating Resource Consumer Groups

Once the pending area is active, you can create the resource consumer groups to which you'll allocate your users. You can assign users initially to one group and switch them to another group later, if necessary

```
SQL> EXECUTE dbms_resource_manager.create_consumer_group -
> (consumer_group => 'local', comment => 'local councils');
PL/SQL procedure successfully completed.
SQL> EXECUTE dbms_resource_manager.create_consumer_group -
> (consumer_group => 'regional', comment => 'regional councils');
PL/SQL procedure successfully completed.

SQL> EXECUTE dbms_resource_manager.create_consumer_group -
> (consumer_group => 'national', comment => 'national office');
PL/SQL procedure successfully completed.
```

Creating Resource Plans

A resource plan contains directives that control the allocation of resources among various resource consumer groups. Resource plans enable you to set limits on resource use by specifying limits on four variables namely CPU_MTH: You use this resource allocation method to specify how you wish to allocate the CPU resource among the resource consumer groups. The default method is called EMPHASIS, and it uses percentages to allocate CPU among the various groups. The alternative method, RATIO, uses ratios instead.

• ACTIVE_SESS_POOL_MTH: This parameter determines the limit on the number of active session in a resource consumer group. The only method available is the ACTIVE_SESS_POOL_ABSOLUTE method, which is the default.

• PARALLEL_DEGREE_LIMIT_MTH: This is the parameter that determines the degree of parallelism used by a specific operation. The only option is PARALLEL_DEGREE_LIMIT_ABSOLUTE (which is the default).

• QUEUEING_MTH: This parameter determines the order in which queued sessions will execute. Only the default FIFO_TIMEOUT option is currently available.

Creating Plan Directives

You need to create a *resource plan directive* to assign resources to the various resource consumer groups in the database. You can allocate resources according to the following criteria:

• *CPU*: Using the CPU method, you can allocate resources among consumer groups or subplans. You can use multiple levels of CPU resource allocation to prioritize CPU usage. For example, you could specify that level 2 gets CPU resources only if any CPU resources are left after level 1 is taken care of.

• *Sessions*: You can control the maximum number of active sessions open at any time by using the ACTIVE_SESSION_POOL parameter. You can also allow for the termination of long-running SQL queries and user sessions.

• *Degree of parallelism*: You can set a limit on the degree of parallelism during any operation.

• *Automatic consumer group switching*: You can specify that, under some conditions, the database will automatically switch sessions to another consumer group.

• *Undo usage*: You can set limits on the number of undo operations a resource consumer group can generate. The database automatically terminates SQL statements that cause the undo generated by a consumer group to exceed its undo limit. This will prevent new members of the consumer group from issuing DML statements.

Validating the Pending Area

Here's how you do it:

```
SQL> EXEC DBMS_RESOURCE_MANAGER.VALIDATE_PENDING_AREA();
PL/SQL procedure successfully completed.
```

Submitting the Pending Area

By submitting the pending area, you actually create all the necessary entities, such as the resource consumer group, the resource plan, and the plan directives, and make them active. You submit the pending area as follows:

```
SQL> EXEC DBMS_RESOURCE_MANAGER.SUBMIT_PENDING_AREA();
PL/SQL procedure successfully completed.
```

Using the Database Resource Manager

To use the Database Resource Manager, the database administrator:

1. Creates resource plans using the PL/SQL package DBMS_RESOURCE_MANAGER.
2. Creates resource consumer groups using the PL/SQL package DBMS_RESOURCE_MANAGER.
3. Creates resource plan directives using the PL/SQL package DBMS_RESOURCE_MANAGER.
4. Assigns users to consumer groups using the PL/SQL package DBMS_RESOURCE_MANAGER_PRIVS.
5. Specifies the plan to be used by an instance. The initialization parameter RESOURCE_MANAGER_PLAN specifies which top plan to use for a given instance. The Database Resource Manager loads this top plan as well as all its descendants (subplans, directives, and consumer groups).

Controlling Access to Data

Understanding User Privileges and Roles

A user **privilege** is the right to run a particular type of SQL statement, or the right to access an object belonging to another user, run a PL/SQL package, and so on. The types of privileges are defined by Oracle Database.

Roles are created by users (usually administrators) to group together privileges or other roles. They are a means of facilitating the granting of multiple privileges or roles to users.

System Privileges

There are over 100 distinct system privileges. Each system privilege allows a user to perform a particular database operation or class of database operations

Here are some common system privileges in an Oracle database:

- ADVISOR
- ALTER DATABASE
- ALTER SYSTEM
- AUDIT SYSTEM
- CREATE DATABASE LINK
- CREATE TABLE
- CREATE ANY INDEX
- CREATE SESSION
- CREATE TABLESPACE
- CREATE USER
- DROP USER
- INSERT ANY TABLE

Granting System Privileges

You use the GRANT statement to grant system privileges to users. For example, to grant the CREATE SESSION system privilege to the sample user, hr, allowing hr to log on to an Oracle database, issue the following statement:

```
SQL> GRANT CREATE SESSION TO hr;
Grant succeeded.
```

The CREATE SESSION privilege enables a user to log on to an Oracle database.

You can grant a system privilege to a user, provided one of the following is true:

- You have been granted the system privilege with the ADMIN OPTION clause
- You have been granted the GRANT ANY PRIVILEGE system privilege.

Here's an example of the use of the ADMIN OPTION clause when granting a system privilege:

```
SQL> GRANT CREATE SESSION TO salapati WITH ADMIN OPTION;
Grant succeeded.
```

Revoking System Privileges

You use the REVOKE statement to revoke system privileges. The revoking of the privileges takes place immediately. Here's an example:

```
SQL> REVOKE DELETE ANY TABLE FROM pasowner;
Revoke succeeded.
```

The SYSDBA and SYSOPER System Privileges

The SYSDBA system privilege includes the RESTRICTED SESSION privilege and has all system privileges with ADMIN OPTION, including the SYSOPER system privilege. The SYSDBA privilege lets you do the following:

• Perform STARTUP and SHUTDOWN operations
• Use the ALTER DATABASE command to open, mount, back up, or change a character set
• Use the CREATE DATABASE command
• Perform ARCHIVELOG and RECOVERY operations
• Create an SPFILE

The SYSOPER privilege similarly includes the RESTRICTED SESSION privilege, and it lets you do the following:

• Perform STARTUP and SHUTDOWN operations
• Use the ALTER DATABASE command to open, mount, or back up
• Perform ARCHIVELOG and RECOVERY operations
• Create an SPFILE

Object Privileges

Object privileges are privileges on the various types of database objects. An object privilege allows a user to perform actions on a specific table, view, materialized view, sequence, procedure, function, or privileges can be assigned to the following types of database objects:

- Tables - select, insert, update, delete, alter, debug, flashback, on commit refresh, query rewrite, references, all
- Views- select, insert, update, delete, under, references, flashback, debug
- Sequence- alter, select
- Packeges, Procedures, Functions (Java classes, sources...) execute, debug
- Materialized Views- delete, flashback, insert, select, update
- Directories- read, write
- Libraries- execute
- User defined types-execute, debug, under
- Operators-execute

User Roles

A **role** groups several privileges and roles, so that they can be granted to and revoked from users simultaneously. A role must be enabled for a user before it can be used by the user.

Oracle Database provides some predefined roles to help in database administration

There are several predefined roles in an Oracle database, including the EXP_FULL_DATABASE, IMP_FULL_DATABASE, and RECOVERY_CATALOG_OWNER roles. In addition, every Oracle database contains the following three important roles, which have listed privileges:

• *The CONNECT role*: CREATE SESSION (prior to Oracle Database 10*g* Release 2, the CONNECT role had several other privileges, but now it has only the single CREATE SESSION privilege)

• *The RESOURCE role*: CREATE CLUSTER, CREATE INDEXTYPE, CREATE OPERATOR, CREATE PROCEDURE, CREATE SEQUENCE, CREATE TABLE, CREATE TRIGGER, CREATE TYPE
• *The DBA role*: All system privileges WITH ADMIN OPTION

You create an Oracle role with the create role command and then grant the role to the user with the grant command as seen in this command:

```
SQL> Create role select_data_role;
SQL> Grant select on emp, dept, bonus to select_data_role;
```

You can then grant that Oracle role to other users as in this case where we grant the select_data_role to the ROBERT user role. Once this is done, ROBERT will be able to query the EMP, DEPT and BONUS tables in the SCOTT schema:

```
SQL> GRANT select_data_role TO Robert;
```

Oracle roles have some limitations. In particular object privileges are granted through Oracle roles cannot be used when writing PL/SQL code. When writing PL/SQL code, you must have direct grants to the objects in the database that your code is accessing.

If you wish to revoke an Oracle role from a user, simply use the revoke command as demonstrated earlier in this chapter:

```
SQL> REVOKE select_data_role FROM Robert;
```

Auditing Database Usage

Auditing user activity can potentially lead to a large amount of data to keep track of, but fortunately, Oracle offers you a lot of control over what type of activities you want to audit. You can audit just at the session level or at the entire database level. Oracle makes a broad distinction between *standard auditing* and *fine-grained auditing*. Standard auditing is based on statement-, privilege-, and object-level auditing. Fine-grained auditing deals with data access at a granular level, with actions based on content, such as value > 100,000.

Standard Auditing

Oracle Database 10*g* lets you audit database use at three different levels: statement, privilege, and object.

A ***statement-level audit*** specifies the auditing of all actions on any type of object.

A ***privilege-level audit*** tracks actions that stem from system privileges

Finally, an ***object-level audit*** monitors actions such as UPDATE, DELETE, and INSERT statements on a specific table, so you could audit all deletes on the hr.employees table.

For each of the three levels of auditing, you can choose to audit either by *session* or by *access*

Enabling Auditing

In order for you to audit any user activity within the database, and even attempts to log into the database, you need to enable auditing by specifying the **AUDIT_TRAIL** parameter in your init.ora file.

The parameter can take the following values:

• **NONE**: Disables database auditing; NONE is the default value for this parameter
• **OS**: Specifies that Oracle will write the audit records to an operating system file (operating system audit trail)

• **DB**: Specifies that Oracle will write the audit records to the database audit trail, viewable asDBA_AUDIT_TRAIL (stored in the SYS.AUD$ table)
• **DB, EXTENDED**: Specifies that Oracle will send all audit records to the database audit trail (SYS.AUD$), and in addition, populates the SQLBIND and SQLTEXT CLOB columns
• **XML:** Specifies database auditing, with the XML-format audit records going to the OS files
• **XML, EXTENDED**: Same as the XML setting, but also records all audit-trail columns, including SQLTEXT and SQLBIND
You can use the DBA_AUDIT_TRAIL view to make use of the information in the database audit trail table (SYS.AUD$).

Oracle's Default Auditing
Even when you don't set up database auditing by specifying the AUDIT_TRAIL parameter at all, by default Oracle will log three types of database actions to the $ORACLE_HOME/rdbms/auditdirectory:
• Connections as SYSOPER or SYSDBA
• Database startup
• Database shutdown
You can audit all actions of the user SYS, including all users connecting with the SYSDBA or SYSOPER privileges, by setting the AUDIT_SYS_OPERATIONS init.ora parameter to true.
AUDIT_SYS_OPERATIONS=TRUE
Turning Auditing On
Here is a more powerful audit option that ensures the auditing of all privileges:
SQL> **AUDIT ALL PRIVILEGES**;
Audit succeeded.
Turning Auditing Off
To turn auditing off, you use a statement that is almost identical to the one you used to turn auditing on. The big difference, of course, is that you use the NOAUDIT keyword in place of AUDIT. Here are some examples:
SQL> **NOAUDIT SESSION**;
Noaudit succeeded.
SQL> NOAUDIT DELETE ANY TABLE BY salapati WHENEVER NOT SUCCESSFUL;
Noaudit succeeded.
SQL> NOAUDIT DELETE ANY TABLE BY salapati;
Noaudit succeeded.

Using System-Level Triggers for Auditing
The following are the main types of system-level triggers that Oracle Database 10*g* offers:
• ***Database startup triggers***: You can use these triggers mostly to execute code that you want to execute immediately after database startup.
• ***Logon triggers***: These triggers provide you with information regarding the logon times of a user and details about the user's session.
• ***Logoff triggers***: These triggers are similar to the logon triggers, but they execute right before the user's session logs off.

• ***DDL triggers*:** You can capture all database object changes with these triggers.
• ***Server error triggers***: These triggers capture all major PL/SQL code errors into a special table

Fine-Grained Auditing

Auditing all SELECT statements would lead to a colossal amount of audit data, but fortunately there's an easy out. Oracle lets you audit actions in the database on the basis of *content*. That is, you can specify that the audit records be written not for all SELECT, INSERT, UPDATE, and DELETE statements, but only for statements that meet certain criteria.

Enabling Fine-Grained Auditing

You don't need to turn on database-wide auditing to use FGA, and since the auditing is based on table access, it is virtually impossible to bypass FGA policies. FGA records are accessible through the **DBA_FGA_AUDIT_TRAIL** and **DBA_COMMON_AUDIT_TRAIL** views, with the latter view combining both standard and fine-grained audit log records.

You use the DBMS_FGA package's ADD_POLICY procedure to add a fine-grained audit policy.

Viewing the Audit Trail

The **DBA_FGA_AUDIT_TRAIL** view shows you the audit trail (stored in the sys.fga_aud$ table) when you use FGA in your database. It provides fine-grained audit information, such as the timestamp, database user ID, object name, and actual SQL text used in the statement flagged by your FGA policy. Here's an example:

```
SQL> SELECT timestamp,
db_user,
os_user,
object_schema,
object_name,
sql_text
FROM dba_fga_audit_trail;
```

Authenticating Users

Database authentication refers to the authentication of the user account and password directly by Oracle.

Database Authentication

Database authentication is the standard verification of a user's access privileges by using database passwords. Here's an example of database authentication:

```
SQL> CREATE USER scott IDENTIFIED BY tiger;
```

Locking Accounts

Any user account that is locked can be unlocked for free access with the following statement:

```
SQL> ALTER USER hr ACCOUNT UNLOCK;
User altered.
```

Here's an example of creating a profile with the time period for locking the account:

```
SQL> CREATE PROFILE test_profile
2 LIMIT FAILED_LOGIN_ATTEMPTS 5
3* PASSWORD_LOCK_TIME UNLIMITED
Profile created.
```

Password Expiration

Password aging policies, which ensure that users don't hang onto the same password for a long time, are a standard part of database security. Once a password expires, the user is forced to change it. You can make a password expire with the ALTER USER command, as shown here:

SQL> **ALTER USER hr IDENTIFIED BY hr PASSWORD EXPIRE**;
User altered.

The Password File

Oracle will let you choose how you want your privileged users to connect to the database. *Privileged users* are those users who can perform tasks such as starting up and shutting down the database. By default, only the SYS user has the SYSDBA and SYSOPER privileges, both of which are considered high-level privileges. The SYS user can grant these privileges to other users.

The **REMOTE_LOGIN_PASSWORDFILE** initialization parameter specifies whether Oracle checks for a password file.

The **REMOTE_LOGIN_PASSWORDFILE** parameter can take the following two values:

• **none**: No password file is used. This is the default, and it permits only operating system–authenticated users to perform privileged database administration tasks.

• **shared**: Creates a password file that can be used by multiple databases running on a single server. The password file includes both SYS and non-SYS users. Any user that is granted the SYSDBA or SYSOPER privilege is automatically added to the password file.

Encrypted Passwords

By default, Oracle user passwords aren't encrypted, and this leaves them vulnerable to unauthorized use. By setting the following environment variables, one on the client and the other on the server, you can ensure that Oracle will always encrypt a password when it's sending it across a network. Set this one on the client: **ora_encrypt_login=true**

And set this one on the server: **dblink_encrypt_login=true**

External Authentication

Another method of authenticating database users is the external authentication method, under which you match the user accounts at the operating system level with the usernames in the database. The advantage to this method is that you'll need only a single username for both the operating system and database connections. To enable operating system authentication, this is how you need to create your users:

SQL> CREATE USER salapati IDENTIFIED EXTERNALLY;
User created.

Proxy Authentication

You can use several middle-tier products to facilitate user interaction with the Oracle database. A web server is often used as the middle or application layer connecting the clients to the database. You can choose to have the middle tier authenticate your users, or you can have the

middle tier pass the username and password to the database for authentication. **Here is an example showing how to authorize a middle tier (appserv) to act as a proxy for a user, with authentication by a password**:

SQL> ALTER USER salapati
2 GRANT CONNECT THROUGH appserv
3* AUTHENTICATED USING PASSWORD;
User altered.

Enterprise User Security

Centralized directories are increasingly being seen as the best way to manage multiple systems within an organization. LDAP is a popular industry standard and Oracle has its own implementation of this standard. Information that has been managed in multiple systems and formats can be brought under one umbrella using a directory service like LDAP.

Shared Schemas

When users are registered and maintained in an LDAP repository, they are referred to as *shared schemas* or *schema-independent users*. When an LDAP-registered user connects to a specific database, the database will ask the LDAP server for confirmation of the user's identity and the roles that should be assigned to the user upon connection.

Data Encryption

Oracle supports encryption of network data through its Advanced Security option. For encryption of data, Oracle provides **two PL/SQL packages**, the older of which is the **DBMS_ OBFUSCATION_TOOLKIT** package. This package enables data encryption by using the **Data Encryption Standard** (DES) algorithm. The toolkit supports triple DES encryption for the highest level of security. It also supports the use of the MD5 secure cryptographic hash. In Oracle Database 10*g*, there is a new PL/SQL encryption package called **DBMS_CRYPTO**. Compared to DBMS_OBFUSCATION_TOOLKIT, DBMS_CRYPTO provides a wider range of advanced security encryption and cryptographic algorithms and is easier to use.

Transparent Data Encryption

Transparent data encryption means that the database will handle encryption and decryption automatically, without the user or the application having to manage the encryption key. This means the application no longer needs to handle the cumbersome process of managing the encryption key.

For example, when you create a table, you can simply specify the ENCRYPT keyword along with the column name, as shown in the following example. This statement creates a table that converts the ssn column values into an encrypted data format when they are stored on disk:

SQL> **CREATE TABLE employees (**
empno NUMBER(5) PRIMARY KEY
ename VARCHAR2(15) NOT NULL,
ssn NUMBER(9) ENCRYPT,

An Oracle wallet is used to store authentication and signing credentials, including private keys and certificates. Before you can start encrypting or decrypting a table, the encryption key is retrieved from the Oracle wallet and stored in the SGA.

Here are the steps you need to follow in order to use the transparent data encryption feature:

1. Create an Oracle wallet.

2. Open the Oracle wallet.

3. Generate the master encryption key that will be used to encrypt the column's encryption key.

Encrypting the Table Columns

Now that you've created the master encryption key, you can start encrypting your table data by using the ENCRYPT keyword after the name of the column you want to encrypt.

First, let's look at how to encrypt a column while creating the table. In the following example, the ssn column in the employees table is encrypted:

```
SQL> CREATE TABLE EMPLOYEES
first_name VARCHAR2(30),
last_name VARCHAR2(30),
emp_id NUMBER (9),
salary NUMBER(6),
ssn NUMBER(9) ENCRYPT;
Table created.
```

Table creation is not the only time you can encrypt a table's columns. You can also encrypt a column in an existing table by using the ALTER TABLE statement. Let's add a new column, ENCRYPT_ID, to the employees table:

```
SQL> ALTER TABLE EMPLOYEES ADD (ENCRYPT_ID NUMBER(9) ENCRYPT);
Table altered.
```

You can also encrypt an existing column in a table, as shown here:

```
SQL> ALTER TABLE EMPLOYEES MODIFY (EMP_ID ENCRYPT);
Table altered.
```

Database Security Dos and Don'ts

You can take several basic steps to enhance the security of your Oracle database. Let's review these security guidelines.

User Accounts

Oracle recommends that you **lock *and* expire all default user accounts except, of course, the SYS and SYSTEM accounts**, and other user accounts that you'll need, like DBSNMP, SYSMAN, and MGMT_VIEW.

Passwords

Change the passwords for all default user accounts immediately after creating the database. **You should set passwords for the SYS and SYSTEM** users while creating the database, although this isn't mandatory. You can also use Oracle's password-complexity verification routine to make sure your users' passwords meet standard password-complexity requirements.

Operating System Authentication

Two initialization parameters enable access to an Oracle database through authentication at the operating system level. One is the well-known **OS_AUTHENT_PREFIX** parameter, which many people use to create the OPS$ account for use in shell scripts and other places. The other initialization parameter affecting operating system authentication of users is the

REMOTE_OS_AUTHENT parameter, which enables users who authenticate themselves not on the server, but on a remote workstation, to gain access to your database.

Database Auditing

You should audit all unsuccessful attempts to log in to the database. In addition, you can audit all actions by any user connected as SYSDBA or SYSOPER. To enable auditing of all SYSDBA and SYSOPER operations, you need to set the following initialization parameter:

AUDIT_SYS_OPERATIONS=TRUE

Granting Privileges

Oracle recommends strongly that you avoid granting ANY privileges, as in delete ANY table, to reduce your vulnerability. You can avoid this problem generally by refraining from (carelessly) granting object privileges *directly* to users.

Protecting the Data Dictionary

Users that are granted the ANY system privilege can drop data dictionary tables. To protect your data dictionary, you must set the 07_DICTIONARY_ACCESSIBILITY configuration parameter to FALSE in your parameter file.

The Network and the Listener -Securing the Listener

You can also prevent a user from using the SET command to interfere with listener functions. To do this, you need to add the following line to your listener.ora configuration file: ADMIN_RESTRICTIONS=ON By default, this parameter is set to false.

Securing the Network

One of the basic security requirements for today's Internet-based database applications is that you must have a firewall protecting your system from the external world. In addition to having a normal firewall, you can use a feature of Oracle Net to add an additional

layer of protection called *server-side access controls.* Server-side access controls limit the capability of an address to connect to your database using the listener service.

Oracle's Advanced Security Option

Here are some of the additional security features available when you use Oracle's Advanced Security option:

• Encryption of network traffic among clients, application servers, and databases
• Sophisticated authentication methods for users
• Centralized user management
• Support for Public Key Infrastructure (PKI)

Application Security

There are some commonsense policies involving roles and SQL*Plus use that your organization must enforce to provide strong application security.

Granting Privileges Through Roles

You should minimize the number of direct object privileges by letting stored code such as procedures and packages be the means through which users can issue DML statements.

Disabling Roles

All application roles should use the SET ROLE statement to enable the roles granted to users. Application users should be granted roles only for specific purposes, and the roles should be revoked from them when they aren't needed any longer.

BACKING UP DATABASES

You can perform database backups in two different ways: use Oracle's Recovery Manager (RMAN) interface or use operating system utilities.

Backing Up Oracle Databases

Database backups are used to avoid the loss of data, so it's essential to have a backup system in place. Backups involve keeping copies of the key Oracle database files: data files, the control file, and the archived redo log files. Physical backups involve the copying of database files. You can perform physical backups in two main ways:

• Use operating system utilities like cp and dd to back up files to perform user-managed backups. You use a combination of operating system backup commands and SQL*Plus commands to back up the database files.

• Use the Oracle-provided utility, Recovery Manager (RMAN), to perform the backups. RMAN can be used in the command-line mode, as well as through the OEM Database Control interface.

Important Backup Terms

Archivelog and Noarchivelog Modes

Archivelog mode: In this mode, Oracle saves (archives) the filled redo logs. Thus, no matter how old the database backup is, if you are running in archivelog mode, you can recover the database to any point in time using the archived logs.

• *Noarchivelog mode*: In this mode, the filled redo logs are overwritten and not saved. The noarchivelog mode thus implies that you can restore only the backup, and you'll lose all the changes made to the database after the backup was performed. The noarchivelog mode of operation means that you can recover from a crash of only the database instance. If there is a media failure (for example, a loss of a disk), a database in noarchivelog mode may be restored from a backup, but it will lose all changes made to the database since the backup was made.

Whole and Partial Database Backups

You can back up either an entire database or part of it, such as a tablespace or a data file. Note that you can't back up a partial database if the database is running in noarchivelog mode, unless all the tablespaces and files in the partial backup are read-only. You can make a whole database backup in either archivelog or noarchivelog mode.

Consistent and Inconsistent Backups

The difference between consistent and inconsistent backups is simple. A *consistent backup* doesn't need to go through a recovery process. When a backup is used to recover a database or a part of a database (such as a tablespace or a data file), first you need to restore the backup, and then you recover the database. In the case of a consistent backup, you don't have to perform any recovery steps. An *inconsistent backup,* on the other hand, always needs to undergo a recovery.

Open and Closed Backups

Online or *open* (or *hot*/*warm*) backups are backups you make while the database is open and accessible to users. You can make an online backup of the entire database (or a tablespace or data file) as long as the database is being run in archivelog mode. You can't make an online backup if the database is running in noarchivelog mode.

A *closed* backup of a database, also called a *cold* backup, is made while the database is shut down. A closed backup is always consistent, as long as the database wasn't shut down with the SHUTDOWN ABORT command.

Physical and Logical Backups

Logical backups are backups made using the Data Pump Export utility, and they contain logical objects like tables and procedures. These backups are in proprietary binary form, and their data can be extracted only by using Oracle's own Data Pump Import utility.

Physical backups refer to the backing up of the key Oracle database files: data files, archived redo logs, and control files. Physical backups are made on disk or on tape drives.

Backup Levels

Following are the levels at which you can perform Oracle database backups:

• *Whole database*: You back up all files including the control file. This level is applicable to both archivelog and noarchivelog modes of operation.

• *Tablespace backups*: You back up all the data files belonging to a tablespace. Tablespace backups are applicable only in the archivelog mode.

• *Data file backups*: You back up a single data file. Data file backups are valid in the archivelog mode only.

Maintaining a Redundancy Set

Always keep a redundancy set online so you can recover faster. A *redundancy set* is defined as the following:

• Last backup of all data files
• Last backup of the control file
• Multiplexed copies of the current redo log files
• Copies of the current control file that's being used
• All the archived redo logs since the last backup

The Recovery Manager (RMAN)

What is RMAN?

- Recovery MANager is an Oracle Database client that performs backup and recovery tasks on your databases and automates administration of your backup strategies. It greatly simplifies backing up, restoring, and recovering database files.

RMAN Purposes:

- Data protection – physical backups of your database
- Data preservation – a copy for yearend accounting
- Data transfer – move a database from one platform to another

RMAN Benefits:

- Incremental backups
- Block validation
- Compression
- Encryption
- Duplication
- Cross-platform data conversion

RMAN components:

- Target database - the Oracle database which the backup or recovery operations are being performed.
- RMAN client – executable that interprets commands, directs server sessions to execute those commands, and records its activity in the target database control file (or catalog).
- Fast Recovery Area (Optional) – a disk location that you can use to store recovery-related files such as control file and online redo log copies, archived redo log files, flashback logs, and RMAN backups.
- Media manager (Optional) - an application required for RMAN to interface with sequential media devices such as tape libraries. A media manager controls these devices during backup and recovery, managing the loading, labeling, and unloading of media.
- Recovery catalog (Optional) - A separate database schema used to record RMAN activity against one or more target databases. A recovery catalog preserves RMAN repository metadata if the control file is lost, making it much easier to restore and recover following the loss of the control file.

Types of Database Connections

You can connect to the following types of databases. Target database RMAN connects you to the target database with the SYSDBA privilege. If you do not have this privilege, then the connection fails.

Recovery catalog database

This database is optional. You can also use RMAN with the default NOCATALOG option.

Auxiliary database

You can connect to a standby database, duplicate database, or auxiliary instance (standby instance or tablespace point-in-time recovery instance

Using Basic RMAN Commands

After you have learned how to connect to a target database, you can immediately begin performing backup and recovery operations. Use the examples in this section to go through a basic backup and restore scenario using a test database. These examples assume the following:
The test database is in ARCHIVELOG mode. You are running in the default NOCATALOG mode. The RMAN executable is running on the same host as the test database.

Connecting to the Target Database

rman TARGET /

If the database is already mounted or open, then RMAN displays output similar to the following:

Recovery Manager: Release 9.2.0.0.0
connected to target database: RMAN (DBID=1237603294)

Reporting the Current Schema of the Target Database

In this example, you generate a report describing the target datafiles. Run the report schema command as follows:

RMAN> REPORT SCHEMA; (RMAN displays the datafiles currently in the target database.)

Backing Up the Database

In this task, you back up the database to the default disk location. Because you do not specify the format parameter in this example, RMAN assigns the backup a unique filename.

Making a Full Backup

Run the backup command at the RMAN prompt as follows to make a full backup of the datafiles, control file, and current server parameter file (if the instance is started with a server parameter file) to the default device type:

RMAN> BACKUP DATABASE;

Making an Incremental Backup

Incremental backups are a convenient way to conserve storage space because they back up only database blocks that have changed. RMAN compares the current datafiles to a base backup, also called a level 0 backup, to determine which blocks to back up.

RMAN> BACKUP INCREMENTAL LEVEL 1 DATABASE;

Backing Up Archived Logs

Typically, database administrators back up archived logs on disk to a third-party storage medium such as tape. You can also back up archived logs to disk. In either case, you can delete the input logs automatically after the backup completes. To back up all archived logs and delete the input logs (from the primary archiving destination only), run the backup command at the RMAN prompt as follows:

RMAN> BACKUP ARCHIVELOG ALL DELETE INPUT;

Listing Backups and Copies

To list the backup sets and image copies that you have created, run the list command as follows:

RMAN> LIST BACKUP;

To list image copies, run the following command:

RMAN> LIST COPY;

RMAN Recovery: Basic Steps

If possible, make the recovery catalog available to perform the media recovery. If it is not available, then RMAN uses metadata from the target database control file. Assuming that you have backups of the datafiles and at least one autobackup of the control file.

The generic steps for media recovery using RMAN are as follows:

· Place the database in the appropriate state: mounted or open. For example, mount the database when performing whole database recovery, or open the database when performing online tablespace recovery.

· Restore the necessary files using the RESTORE command.

· Recover the datafiles using the RECOVER command.

· Place the database in its normal state.

Mechanism of Restore and Recovery operation:

The DBA runs the following commands:

RESTORE DATABASE;

RECOVER DATABASE;

The RMAN recovery catalog obtains its metadata from the target database control file.RMAN decides which backup sets to restore, and which incremental backups and archived logs to use for recovery. A server session on the target database instance performs the actual work of restore and recovery.

Mechanics of Recovery: Incremental Backups and Redo Logs

RMAN does not need to apply incremental backups to a restored level 0 incremental backup: it can also apply archived logs. RMAN simply restores the datafiles that it needs from available backups and copies, and then applies incremental backups to the datafiles if it can and if not applies logs.

About Block Media Recovery

You can also use the RMAN BLOCKRECOVER command to perform block media recovery. Block media recovery recovers an individual corrupt datablock or set of datablocks within a datafile. In cases when a small number of blocks require media recovery, you can selectively restore and recover damaged blocks rather than whole datafiles.

Benefits of Using the Recovery Catalog as the RMAN Repository

When you use a recovery catalog, RM AN can perform a wider variety of automated backup and recovery functions than when you use the control file in the target database as the sole repository of metadata. The following features are available only with a catalog:

- You can store metadata about multiple target databases in a single catalog.
- You can store metadata about multiple incarnations of a single target database in the catalog. Hence, you can restore backups from any incarnation.
- Resynchronizing the recovery catalog at intervals less than the CONTROL_FILE_RECORD_KEEP_TIME setting, you can keep historical metadata.
- You can report the target database schema at a noncurrent time.
- You can store RMAN scripts in the recovery catalog.

Types of Files That RMAN Can Back Up

The BACKUP command can back up the following types of files:

(i) Database, which includes all datafiles as well as the current control file and current server

(ii)parameter file: Tablespaces (except for locally-managed temporary tablespaces)

(iii)Current datafiles

(iv)Current control file

(v)Archived redo logs

(vi)Current server parameter file

Backup sets

RMAN does not back up the following:

(i) Online redo logs

(ii) Client-side initialization parameter files or noncurrent server parameter files

How to Configure RMAN

RMAN can invoked from the command line on the database host machine like so:

```
C:\>rman target sys/sys_password
Connected to target database: ORCL (DBID=1036216947)
RMAN> show all;
RMAN configuration parameters are:
CONFIGURE RETENTION POLICY TO REDUNDANCY 2;
CONFIGURE BACKUP OPTIMIZATION OFF; # default
CONFIGURE DEFAULT DEVICE TYPE TO DISK; # default
CONFIGURE CONTROLFILE AUTOBACKUP ON;
CONFIGURE CONTROLFILE AUTOBACKUP FORMAT FOR DEVICE TYPE DISK TO
'e:\backup\ctl_sp_bak_%F';
CONFIGURE DEVICE TYPE DISK PARALLELISM 2;
CONFIGURE DATAFILE BACKUP COPIES FOR DEVICE TYPE DISK TO 1; # default
CONFIGURE ARCHIVELOG BACKUP COPIES FOR DEVICE TYPE DISK TO 1; #
default
CONFIGURE CHANNEL 1 DEVICE TYPE DISK FORMAT 'e:\backup\%U.bak'
MAXPIECESIZE 4G;
CONFIGURE CHANNEL 2 DEVICE TYPE DISK FORMAT 'e:\backup\%U.bak'
MAXPIECESIZE 4G;
CONFIGURE MAXSETSIZE TO UNLIMITED; # default
CONFIGURE SNAPSHOT CONTROLFILE NAME TO
'C:\ORACLE\ORA92\DATABASE\SNCFORCL.ORA'; #
default
RMAN>
```

Backing Up the Control File

It's a good practice to back up your control file on a regular basis by using the BACKUP CONTROLFILE TO TRACE command, as shown here:

```
SQL> ALTER DATABASE BACKUP CONTROLFILE TO TRACE;
Database altered.
SQL>
```

You should immediately back up your control file after you perform any of the following operations:

- Create or drop a tablespace
- Add or rename a data file
- Add, rename, or drop an online redo log group or member

The Oracle Backup Tool

Oracle Backup is Oracle Corporation's own media manager for tape backups, which simplifies and automates backup and recovery operations.

You can use the following tools when working with Oracle Backup:

- A GUI tool called the Oracle Backup Web Interface, which allows you to configure administrative domains, manage operations, and backup and restore data.
- A command-line Oracle Backup interface, which lets you perform many of the same functions as the GUI tool
- OEM's interface to the Oracle Backup tool

Benefits of Oracle Backup

Oracle Backup provides the following benefits.

• Out-of-the-box integration with the RMAN tool

• Automated control of tape backups, automatic tape drive cleaning, and automatic tape expiration and recycling

• Ability to back up both the database and operating system files

• Easy configuration

• Ability to share tape libraries across platforms

• Flexible backup strategies, including full, incremental, and differential backups

• Secondary verification of backup data

Oracle Backup Administrative Domain

An administrative domain consists of three types of servers:

• *Administrative server*: This server maintains the Oracle Backup catalog files, which contain configuration and history information.

• *Media server*: This server has the secondary storage devices, such as tape drives and robotic tape libraries attached to it. A media server must have at least one tape drive attached to it. The media server transfers data to and from the attached media devices.

• *Client host server*: This server contains the Oracle databases that are backed up by Oracle Backup.

Installing Oracle Backup

1. Log in as root and create a working directory named backup.

$ mkdir -p /usr/local/oracle/backup

2. Move to the working directory and invoke the setup program.

$ cd /usr/local/oracle/backup

$ /mnt/cdrom/setup

3. The setup program's welcome page appears, with three choices regarding the operating system. Select option 2 for a Linux installation.

4. The setup process loads the Oracle Backup software onto the server and prompts you to choose yes to continue the installation.

6. In the next step, you're offered a choice between an interactive and batch mode of installation. Choose the interactive mode (option a).

7. You are now asked to select a host role,

8. The installation process will then ask you the following question:

Is localhost connected to any SCSI tape libraries that you'd like to use with ➥

Oracle Backup [no]?

You can answer yes to configure a tape library.

10. In the final step, the installer will ask you if you want to install Oracle Backup on another machine. Choose no.

Configuring Oracle Backup

When you install Oracle Backup, it creates default users, hosts, devices, classes, and the null media family. You can choose to use the defaults or configure your own entities

Users Oracle Backup uses the following classes:

• *admin*: For overall administration of a domain

• *operator*: For standard day-to-day operations

• *oracle*: For specific database privileges

• *reader*: For viewing index information

• *user*: For allowing specific users to interact in a limited way with their domains

Hosts

Hosts are the server machines that host the Oracle Backup tool. You can distinguish between two types of hosts, based on their access mode:

• *ob host*: These are servers on which Oracle Backup components run the background as daemons. These daemons participate in managing the backup and restore operations.

• *Network Data Management Protocol (NDMP) host*: This is a storage appliance from a third party vendor. An NDMP host implements the NDMP protocol and employs NDMP daemons instead of Oracle Backup daemons to back up and restore files.

Devices

Devices include both tape drives and tape libraries. A library is a medium changer that accepts commands to move media between storage locations and tape drives. Following are the basic components of libraries:

• *storage element (se)*: Contains a volume when it is not in use.

• *import-export element (iee)*: Used to move volumes into and out of the library without opening the door and is physically present only on certain libraries.

• *medium transport element (mte)*: Moves a volume from a storage element to a drive.

• *data transfer element (dte)*: A tape drive.

Media Families

Oracle Backup lets you classify your backup media using the following criteria:

• *Volume identification sequence*: Each tape volume has a unique identifier attached to it,when it's either written to the first time or overwritten from the beginning of the tape.

• *Write-allowed period*: Oracle Backup can write to a volume set until a predetermined write allowed period has expired, at which time it closes the volume to further updates.

• *Retention period*: Oracle Backup determines the expiration date and time for each volume set when you first create the set. You can't write to the set past the expiration date.

Performing Backups with Oracle Backup

You can back up data in two different ways:

• *On demand*: You can create immediate, one-time use backup jobs and send your requests to the scheduler when you're ready. Oracle Backup then turns it into a dataset job, making it eligible to run.

• *Scheduled jobs*: You can use the Oracle Backup scheduler to schedule jobs. You can specify backups in terms of day, days of the week, month, quarter, or year.

User-Managed Backups

RMAN is the Oracle-recommended method for backing up and recovering databases. RMAN is designed to take advantage of its knowledge of Oracle's block structures to provide excellent performance, including features like compression, resumable backups and recovery, block-change tracking, and integration with the MML. If you choose this approach, you must keep track of all the backups, check their validity, and also decide which of the backups you'll need during a recovery session. This is the reason Oracle calls this method *user-managed backups.*

Making Whole Database Backups

A whole database backup is a backup of every datafile in the database, plus the control file. Whole database backups are the most common type of backup. Whole database backups can be taken in either ARCHIVELOG or NOARCHIVELOG mode. Before performing whole database backups, however, be aware of the implications of backing up in ARCHIVELOG and NOARCHIVELOG modes.

Whole Closed Backup

To make a closed, or *cold*, backup, the database must have been shut down cleanly through a normal, immediate, or transactional shutdown. In the following sections, you'll learn how you to back up the three main types of files involved in a whole closed backup.

Backing Up the Data Files

You can get the list of all the data files in your database by using the following query:

```
SQL> SELECT file_name FROM dba_data_files;
```

Backing Up the Online Redo Log Files

You'll need to back up all the online redo log files when you perform a closed backup. You can get the list of online redo files by making the following query:

```
SQL> SELECT member FROM v$logfile;
MEMBER
-------------------------------------------------
C:\ORACLENT\ORADATA\HELPME\REDO03.LOG
C:\ORACLENT\ORADATA\HELPME\REDO02.LOG
C:\ORACLENT\ORADATA\HELPME\REDO01.LOG
```

Backing Up the Control Files

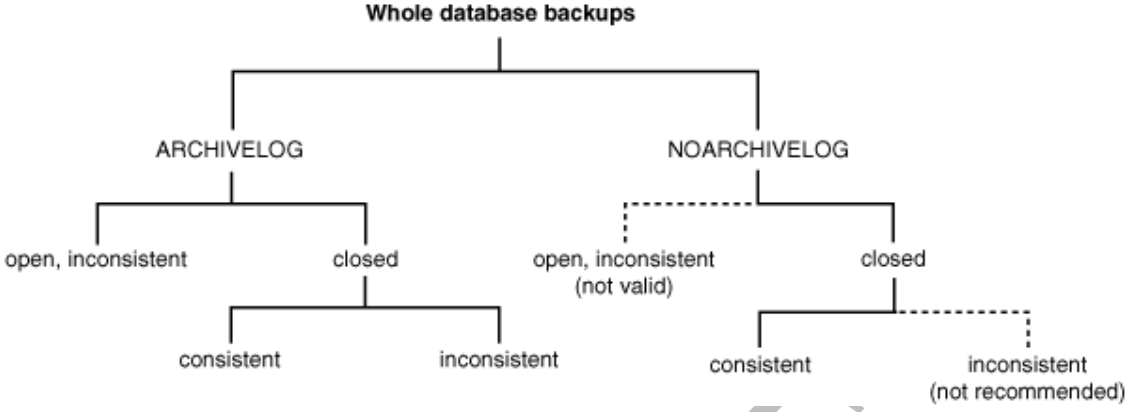
You can find the control file names and their location by querying the V$CONTROLFILE view:

```
SQL> SELECT name FROM v$controlfile;
NAME
-------------------------------------------------
C:\ORACLENT\ORADATA\HELPME\CONTROL01.CTL
C:\ORACLENT\ORADATA\HELPME\CONTROL02.CTL
C:\ORACLENT\ORADATA\HELPME\CONTROL03.CTL
SQL>
```

Whole Database Backups

Figure 5 illustrates the valid configuration options given the type of backup that is performed.

Figure 5 Whole Database Backup Options



Making Partial Database Backups

You don't need to back up the entire database at one time. You can back up a part of the database— for instance, a tablespace or just a single data file. You can make a tablespace backup with the tablespace either online or in an offline status, depending on your needs. First, let's look at an example of an offline backup of a tablespace. You first take the tablespace offline, and then you back up the files that compose the tablespace.

```
SQL> SELECT file_name FROM dba_data_files
WHERE tablespace_name = 'USERS';
/u05/oradata/nicko/users01.dbf
```

As you can see, only one data file belongs to the tablespace USERS. In order to back up the tablespace, you must back up this data file. But first, take the tablespace offline, in case users are accessing any of the data files in that tablespace.

```
SQL> ALTER TABLESPACE users OFFLINE;
```

Now, you can use an operating system utility like cp (or copy on a Windows system) to back up the data file belonging to the USERS tablespace.

Monitoring User-Managed Online Backups

Views for Monitoring Backups	**View Description**
V$BACKUP	This view is of great help in determining if any of the data files are still in backup mode.
V$DATAFILE	This view lists all the data files that belong to all the tablespaces that need to be backed up.
V$LOG	This view displays all the online redo logs for the database.
V$ARCHIVED_LOG	This view displays historical archived log information from the control file.
V$LOG_HISTORY	This view displays the redo logs that have been archived.

Database Corruption Detection

Backed-up database files may become useless during recovery for several reasons: corrupt data files and redo logs, accidentally overwritten files, defective tapes, or even nonexistent files. the term *corruption* to indicate the fact that the data is inconsistent with what it should be.

Detecting Media Corruption

Media corruption can be caused by myriad factors, ranging from user error to bugs in the operating system software, to bad disks, to a Logical Volume Manager (LVM) error, to faulty memory chips. Media defects could lead to corruption in the control files, redo logs, data dictionary, table data, and index data. Your detection of media corruption anywhere in the database involves using scripts to monitor your alert logs on a regular basis and using some Oracle features that enable early detection of problems.

Detecting Data Block Corruption

Block corruption is rare but it does happen. As databases get larger and larger – the probability of it happening at some point nears 100%. Block corruption is while the data is being written to the data blocks, if the write to the block fails abruptly, I mean that there is a partial write in the block, may be because of power disruption or I/O problem, leaving no time for header to be updated, or row data to be populated, oracle leaves the block corrupt. In case of block corruption you can normally use the database unless you try to read that particular block, against which it shoots up the block corruption error. Generally block corruption occurs if write fails on the block, when the transaction is being committed.

Using the ANALYZE Command

You can use the ANALYZE command to catch corrupted data blocks. The following command verifies each data block in the customer table, and if it finds any corrupted blocks, it adds the suspect rows to the invalid_rows table.

```
SQL> ANALYZE TABLE customer VALIDATE STRUCTURE;
```

Using the DBVERIFY Utility

When you suspect data block corruption, you can use the Oracle-provided DBVERIFY utility. The DBVERIFY tool is used from the operating system level. It checks the structural integrity of the database files for corruption.

Using the DBMS_REPAIR Package

dbms_repair is a utility that can detect and repair block corruption within Oracle. It is provided by Oracle as part of the standard database installation.

Enhanced Data Protection for Disaster Recovery

A disaster could easily put your organization data resources out of commission, causing severe service interruptions. For events like those, you need more than the ordinary backup systems in place—you need a *high-availability strategy* in place.

High-Availability Systems

The key to providing such high availability is to have *multiple* data systems

using various architectures. Oracle provides several alternatives, including the following:

(i) *Oracle Real Application Clusters (RAC)*

(ii)Oracle Streams:

(iii)Oracle Data Guard and standby databases

Oracle Data Guard and Standby Databases

Oracle Data Guard is the management and monitoring layer through which the standby databases are maintained. The standby databases are kept up-to-date by propagating changes from the primary server continuously. The main benefits of using the Oracle Data Guard standby database feature are

• High availability
• Protection against disasters
• Protection against physical data corruption
• Protection against user errors
• Failover and switchover capabilities, which can be used for both planned and unplanned switching of production and standby databases.
• Geographical separation of primary and secondary servers through Oracle Net.

Oracle provides the excellent Oracle Data Guard Broker to help create and manage the Oracle Data Guard configurations. The Oracle Data Guard Broker can support up to ten databases (one primary and nine standby) at a time.

Physical and Logical Standby Databases

Physical standby databases are updated by continuously applying the primary database's archived logs. Physical standby databases are identical to the production database. A physical standby database must undergo a constant recovery process for it to be in tune with the production database. Logical standby databases, on the other hand, use the same archived logs to derive transaction information, which is applied to the standby database using SQL statements.

Protection Modes

You can choose three data protection modes when you use the Oracle Data Guard feature to maintain standby databases. The protection modes are a reflection of the trade-off between availability and performance. The following modes are available:

- *Maximum protection mode*:
- *Maximum availability mode*:
- *Maximum performance mode*:

Database Recovery

Recovery is a process in which mistakes can be very expensive in terms of data loss.

Types of Database Failures

To understand Database reovery, a little understanding of different database failures are to be understood. Not all failures involve loss of database data. Based on the failures, the recovery operations carried out.

Statement	A single database operation fails, such as a DML (Data Manipulation

	Language) statement - INSERT, UPDATE, and so on.
User process	A single database connection fails.
Network	A network component between the client and the database server fails, and the session is disconnected from the database.
User error	An error message is not generated, but the operation's result, such as dropping a table, is not what the user intended.
Instance	The database instance fails unexpectedly.
Media	One or more of the database files is lost, deleted, or corrupted.

The Oracle Recovery Process

You can broadly divide Oracle database recoveries into crash and instance recoveries on one hand and media recoveries on the other. Oracle uses only the current data files and online redo log files to bring the database up to date. Crash and instance recovery involves the following two-step procedure:

1. *Roll-forward step*: During this step, formally called *cache recovery*, the database applies the committed and uncommitted data in the current online redo log files to the current online data files.

2. *Rollback step*: During this step, formally called *transaction recovery*, the database removes the uncommitted transactions applied in the previous step, using the undo data in the undo segments

Media Recovery

Oracle media recovery ensures the recovery of up-to-the-minute data, provided you have a copy of a recent backup and archived redo logs. The archived logs are transaction journals, and they contain the complete set of changes made to the database since the last backup.

Restoring vs. Recovering

Using backed-up copies of data files and control files to replace lost or damaged data files and control files is called *restoring*. Bringing the data files up to date using backed-up data files and archived redo log files is called *recovery*.

Open and Closed Media Recovery

Open recovery is recovery performed while the database is open to users. Only the affected data files or tablespaces are taken offline for recovery. A *closed recovery* is a recovery for which you need to shut down the database completely. You'll need to use closed recovery when your entire database needs to be recovered or when your system or rollback (undo) data files are damaged.

Complete and Incomplete Recovery

Complete recovery simply means a recovery with no loss of data. All the changes in the online and archived redo logs are applied to the most recent backup of the database.

Incomplete recovery implies data loss, because you restore only part of the data that existed when the database failure occurred.

Block Media Recovery

If only a few data blocks are corrupted, and the rest of the data file is good, you should consider performing a *block media recovery* instead of a data file recovery. You can perform block media recovery only through RMAN.

Media Recovery vs.Non-File-Based Recoveries

Oracle has developed several non-file-based recovery techniques. In these techniques, the emphasis isn't on restoring and recovering files, but on using either undo data, redo logs, or the new Flashback logs to restore lost objects. Here's a list of these non-file-based recovery techniques

- *Flashback*
- *LogMiner*
- *Data Pump*

Performing Recovery with RMAN

RMAN can help you perform all the user-managed types of recovery, and it provides several other benefits.

RMAN's Benefits for Recovery

- RMAN selects and applies the necessary data and log files during recovery.
- RMAN selects the most recent backup sets and image copies to recover with.
- RMAN can perform recovery at the data block level with the block media recovery feature (an option not otherwise available), which dramatically reduces recovery time.
- RMAN provides *restore optimization,* a great timesaving feature that enables you to bypass data files that are okay during the recovery process. RMAN can check the files that need to be restored and avoid recovering bad files.
- RMAN allows you to recover by applying incrementally updated backups, which drastically reduces recovery time.
- RMAN provides the DUPLICATE command, which lets you easily create clones of your production database.

Using the VALIDATE BACKUP Command to Validate RMAN Backups

You can use the LIST BACKUP command in RMAN to view information about backup sets, backup pieces, and proxy copies.

The VALIDATE BACKUPSET command checks the usability of RMAN backups. You can get the backup set information by first using the LIST BACKUP command. You can then use the VALIDATE BACKUPSET command to check a backup set's usability.

Using the RESTORE . . .VALIDATE Command

You can use the RESTORE . . . VALIDATE command to check whether a certain object of interest is among RMAN's backup sets. Here's an example:

```
RMAN> RESTORE TABLESPACE users VALIDATE;
Starting restore at 29-JUN-05
. . .
```

Finished restore at 29-JUN-05
RMAN>

Using the RESTORE . . . PREVIEW Command

RMAN provides a handy PREVIEW option you can use with the RESTORE command, which lets you identify all the backup files necessary for a specific restore operation.

RMAN Recovery Procedures

You use the following RMAN commands to recover the database (or a part of it):

• RESTORE
• RECOVER

Monitoring RMAN Jobs

You can monitor the status of both an RMAN backup as well as a recovery job by using the V$RMAN_STATUS view.

Typical Media Recovery Scenarios

The following sections take you through several common recovery scenarios using RMAN and user managed recoveries.

Complete Recovery of a Whole Database

Using RMAN for Whole Database Recovery- restore the database with the following command:

RMAN>recover database noredo;

If the online archived redo logs are available, then change the command to the following:

RMAN>restore database;

Once the database is restored, you can then recover the database. Simply use the RMAN recover database command as seen here:

RMAN>recover database;

Recovering a Tablespace

You need to perform a tablespace recovery when you lose one or more data files that belong to the tablespace and you don't have a mirrored copy of the files. The recovery may be open or closed, and it may be a full recovery or a point-in-time recovery. You can recover using either RMAN or user-managed techniques.

Using RMAN to Recover a Tablespace

Here are the recovery steps:

1. Take the tablespace you're going to recover offline. The rest of the database will be functioning normally after you do this:

RMAN> ALTER TABLESPACE sysaux OFFLINE;

2. Restore the tablespace using the RESTORE TABLESPACE command, as follows:

RMAN> RESTORE TABLESPACE sysaux;
Starting restore at 29-JUN-05
using channel ORA_DISK_1
. . .
channel ORA_DISK_1: restore complete
Finished restore at 29-JUN-05
RMAN>

3. Recover the tablespace, as follows:

```
RMAN> RECOVER TABLESPACE sysaux;
Starting recover at 29-JUN-05
using channel ORA_DISK_1
starting media recovery
archive log thread 1 sequence 12 is already on disk as file
. . .
media recovery complete
Finished recover at 29-JUN-05
RMAN>
```

4. Finally, bring the recovered tablespace online, as follows:

```
RMAN> ALTER TABLESPACE sysaux ONLINE;
```

User-Managed Recovery of a Tablespace

Here's a summary of the recovery process:

1. Take the affected tablespace offline:

```
SQL> ALTER TABLESPACE sales01 OFFLINE IMMEDIATE;
```

2. Restore the damaged files:

```
SQL> HOST cp /u01/app/oracle/backup/shan/sales_01.dbf
/u01/app/oracle/oradata/shan/sales_01.dbf
```

3. Recover the offline tablespace:

```
SQL> RECOVER TABLESPACE sales01;
```

4. Bring the tablespace you just recovered online:

```
SQL> ALTER TABLESPACE sales01 ONLINE;
```

POSSIBLE QUESTIONS:

PART B: 6 MARKS

1. Illustrate with an example creating and managing users in Oracle
2. State and explain the purpose database resource manager
3. How the access to the data is controlled? Explain with examples.
4. How auditing the database usage helps in security of the database?
5. Write in brief about the database security do's and don'ts.
6. Justify the need for backing up the oracle databases.
7. Discuss the role of recovery manager in backup
8. Write a note on database corruption detection
9. List out the various types database failures and explain the database recovery process.
10. How RMAN is used to perform recovery for various media recovery scenarios.

Karpagam Academy of Higher Education
Department of CS, CA & IT
Subject: Oracle 10g Administration (17CSP203)

Batch : 2017-2019 **Class: I M.Sc CS**

Objective Type Questions
UNIT IV

S.NO	QUESTIONS	OPTION 1	OPTION 2	OPTION 3	OPTION 4	KEY
1	The _____ clause gives the user a 500MB space allocation in the users tablespace	GRANT QUOTA	GRANT SPACE	GRANT USER	GRANT MEMORY	GRANT QUOTA
2	The _____ statement is used to alter a user in the database.	ALTER SYSTEM	ALTER SET	ALTER USER	ALTER USER SPACE	ALTER USER
3	To drop a user, use the _____ statement	DROP	DROP USER	DROP SYSTEM	DELETE USER	DROP USER
4	A _____ is a collection of resource usage and password related attributes that you can assign to a user	roles	profiles	user	groups	profiles
5	_____ parameters are purely concerned with limiting resource usage.	resource parameters	password parameters	system parameters	memory parameters	resource parameters
6	_____ specifies the total time a session may remain connected to the database	CPU_TIME	LOGICAL_TIME	CONNECT_TIME	WHOLE_TIME	CONNECT_TIME
7	A _____ is used to group together similar users based on their resource needs.	group user	resource plan	resource consumer group	resource cluster	resource consumer group
8	The _____ lays out how resource consumer groups are allocated resouces	resource group	resource parameters	resource plan	resource cluster	resource plan
9	The _____ privilege has to been granted to enable database resource manager	DBMS_RESOURCE	ADMINISTER_RESOURCE_MANAGER	ADMINISTER_MANAGER	DBMS_MANAGER	ADMINISTER_RESOURCE_MANAGER

10	_____ is used to validate the changes before their implementation	submitting area	validate area	pending area	query area	pending area
11	_____ is used to assign resources to the various resource consumer group	resource plan	resource group	resource plan directives	resource plan	resource plan directives
12	The _____ view shows all currently active resource plan.	V$SESSION	V$RSRC_PLAN	V$CURRENT	V$ACTIVE	V$RSRC_PLAN
13	A_____ is the right to execute a particular type of SQL statement or to access a database object owned by anotehr user.	roles	privilege	grant	accept	privilege
14	The SYSDBA system privilege includes the _____ privilege and has all system privileges with ADMIN OPTION	VALIDATED SESSION	ENQUIRY SESSION	QUERYREWRITE	RESTRICTED SESSION	RESTRICTED SESSION
15	_____ are privileges on the various types of database objects.	system privileges	object privileges	query privileges	sysdba privileges	object privileges
16	ALTER and SELECT comes under _____ privileges	view privileges	system privileges	object privileges	sequence privileges	sequence privileges
17	_____ is based on statement, privilege and object level auditing	fine grained auditing	standard auditing	user auditing	database auditing	standard auditing
18	A _____ specifies the auditing of all actions on any type of object	statement level audit	system level audit	object level audit	privilege level audit	statement level audit
19	_____ involves the copying of database files	physical backups	logical backups	server backups	client backups	physical backups
20	A _____ needs a recovery	physical backups	logical backups	inconsistent backups	consistent backups	inconsistent backups
21	An _____ backup is a backup in which the files contain data from different points in time	physical backups	logical backups	inconsistent backups	consistent backups	inconsistent backups
22	_____ are backups made using the datapump export utility	physical backups	logical backups	inconsistent backups	consistent backups	logical backups

23	_____ can create and manage backups on disk and on tape devices.	RMAN	ASH	AWR	ACC	RMAN
24	_____ script is kept in the RMAN recovery catalog	text	stored	valid	table	stored
25	_____ script is kept in regular text files	text	stored	valid	table	text
26	The _____option lets you to specify how many copies of the backups you want to retain.	COPY	REDUNDANCY	RETENTION WINDOW	BACKUP COPY	REDUNDANCY
27	The _____ option ensures that RMAN doesn't perform a file backup if it has already backed up identical versions of the file	BACKUP OPTIMIZATION	BACKUP RETENTION	BACKUP PARALLELISM	BACKUP COPY	BACKUP OPTIMIZATION
28	The _____ view displays all the online redo logs for the database	V$LOG	V$BACKUP	V$REDO	V$ACTIVE	V$LOG
29	_____ corruption occurs when you have inconsistent data in tables or indexes	repair	data block	block	files	data block
30	_____ tool if used from the operating system to check the structural integrity of the database files for corruption	DBVERIFY	DBATTACH	DBANALYSE	DBREPAIR	DBVERIFY
31	You can use the_____ command to validate backup sets before you use them from a recovery.	VALIDATE BACKUP	VALIDATE BACKUPSET	VALIDATE VOLUME	VALIDATE CHECKUP	VALIDATE BACKUPSET
32	_____are the means by which RMAN conducts its backup and recovery operations	Channels	volumes	triggers	functions	Channels
33	The default value of degree of parallelism is _____	2	4	5	1	1
34	The _____ command will remove the recovery catalog:	REMOVE CATALOG	DELETE CATALOG	DROP CATALOG	ESC CATALOG	DROP CATALOG
35	The _____ option tells the RMAN to finish the backup as fast as it can.	PARTIAL	MINIMIZE TIME	MINIMIZE LOAD	DURATION	MINIMIZE TIME

36	The _____file, is used to track the physical location of all database block changes.	modify tracking	track files	track database	change-tracking	change-tracking
37	_____contains the Oracle databases that are backed up by Oracle Backup.	Media server	Client host server	Administrative server	client server	Client host server
38	_____is the frequent writing of the dirty database buffers in the cache to disk by the database writer	Fast checking	Fast-Start Checkpointing	start checkpointing	fast start recovery	Fast-Start Checkpointing
39	Bringing the data files up to date using backed-up data files and archived redo log files is called _____	restoring	recovery.	backing	blocking	recovery.
40	The process of applying the contents of both the archived and redo log files to bring the data files up to date is called _____	transaction recovery	open recovery	cache recovery	closed recovery	cache recovery
41	A _____ is a recovery for which you need to shut down the database completely.	transaction recovery	open recovery	cache recovery	closed recovery	closed recovery
42	when you use the _____command, RMAN restores a data file from an image backup	RESTORE	RECOVER	BACKING	BLOCKING	RESTORE
43	If your redo logs are lost or damaged, you need to specify _____ in the CREATE CONTROLFILE statement.	RECOVERLOGS	BACKUPLOGS	RESETLOGS	RESTORELOGS	RESETLOGS
44	_____allows you to view old row data based on a point in time or an SCN	Flashback Versions Query	Flashback Query	Flashback Transaction Query	Flashback Drop	Flashback Query
45	A user can permanently remove the objects from the Recycle Bin using the _____ command	DROP	RESTORE	DELETE	PURGE	PURGE
46	You can use ____ package to manage alerts	dbms_alert	alert_manage	dbms_server	alert_server	dbms_server
47	_____ tablespaces are used to store objects for the duration of a users session only.	Temporary tablespaces	Undo tablespaces	Databases	Tablespaces	Temporary tablespaces

48	SQL Access Advisor to help determine _____	Cursor view	Physical View	Materialized view	Updatable view	Materialized view
49	_____limits the total CPU time used during a session.	SESSIONS_PER_USER	CPU_PER_CALL	CONNECT_TIME	CPU_PER_SESSION	CPU_PER_SESSION
50	By using _____ you can set a maximum allowable number of concurrent sessions within any consumer resource group.	Undo pool	active session pool	current pool	automatic pool	active session pool
51	There are____distinct system privileges	50	75	100	200	100
52	A_____ audit specifies the auditing of all actions on any type of object	privilege level	object level	function level	statement level	statement level
53	You can capture all database object changes with _____triggers	DDL triggers	startup triggers	Logon triggers	DML triggers	DDL triggers
54	When users are registered and maintained in an LDAP repository, they are referred to as _____	logical schemas	shared schemas	registered schemas	functional schemas	shared schemas
55	Expand RMAN	Recovery Manager	Regression Manager	Remedial Manager	Reordering Manager	Recovery Manager

UNIT-V

SYLLABUS

Improving database performance**:** SQL query optimization - approach to oracle performance tuning - optimizing oracle query processing - oracle optimization and oracle cost based optimizer - writing efficient SQL - DBA's role to improve SQL processing - SQL performance tuning tools - explain plan - SQL tuning advisor - simple approach to tuning SQL statement.Performance tuning: Tuning the instance - introduction to instance tuning - automatic performance tuning vs dynamic performance views - tuning oracle memory - evaluating system performance - measuring IO performance - measuring instance performance - simple approach to instance tuning.

Improving Database Performance: SQL Query Optimization

Performance tuning focuses primarily on writing efficient SQL, allocating appropriate computing resources, and analyzing wait events and contention in the system.

A Systematic Approach to Performance Tuning

It's important to follow a systematic approach to tuning database performance. Performance problems commonly come to the fore only after a large number of users start working on a new production database. Oracle suggests a specific design approach with the following steps:

1. Design the application correctly.
2. Tune the application SQL code.
3. Tune memory.
4. Tune I/O.
5. Tune contention and other issues.

Optimizing Oracle Query Processing

Query optimization is the process of choosing the most efficient execution plan. The goal is to achieve the result with the least cost in terms of resource usage. A user's SQL statement goes through the *parsing*, *optimizing*, and *execution* stages. In the next sections you'll examine what Oracle does during each of these steps.

Parsing

Parsing primarily consists of checking the syntax and semantics of the SQL statements. The end product of the parse stage of query compilation is the creation of the *parse tree*, which represents the query's structure. The SQL statement is decomposed into a relational algebra query that's analyzed to see whether it's syntactically correct.

Optimization

During the optimization phase, Oracle uses its optimizer—which is a cost-based optimizer (CBO)— to choose the best access method for retrieving data for the tables and indexes referred to in the query.

Query Rewrite Phase

In this phase, the parse tree is converted into an abstract logical query plan. This is an initial pass at an actual query plan, and it contains only a general algebraic reformulation of the initial query

Execution Plan Generation Phase

The physical query or execution plan takes into account the following factors:

- The various operations (for example, joins) to be performed during the query
- The order in which the operations are performed
- The algorithm to be used for performing each operation
- The best way to retrieve data from disk or memory
- The best way to pass data from one operation to another during the query

Query Execution

During the final stage of query processing, the optimized query (the physical query plan that has been selected) is executed. If it's a SELECT statement, the rows are returned to the user. If it's an INSERT, UPDATE, or DELETE statement, the rows are modified. The SQL execution engine takes the execution plan provided by the optimization phase and executes it.

Query Optimization and the Oracle Cost-Based Optimizer

The job of the Optimizer is to find the optimal or best plan to execute your DML statements such as SELECT, INSERT, UPDATE, and DELETE.

1. Choosing Your Optimization Mode
2. Providing Statistics to the Optimizer - The necessary statistics are as follows:
    - The number of rows in a table
    - The number of rows per database block
    - The average row length
    - The total number of database blocks in a table
    - The number of levels in each index
    - The number of leaf blocks in each index
3. Setting the Optimizer Mode
4. Setting the Optimizer Level

What Does the Optimizer Do?

1. SQL Transformation
2. Choosing the Access Path

(i) Full Table Scans
(ii) Table Access by ROWID
(iii) Index Scans
(iv) Choosing the Join Method
(v) Choosing the Join Order

Drawbacks of the CBO

However, the CBO isn't always perfect, and you need to watch out for the following:

- The CBO isn't fixed across Oracle versions. Execution plans can change over time as versions change. Later in this chapter, you'll see how to use stored outlines so the Optimizer always uses a known plan to maintain plan stability.
- Application developers may know more than the CBO when it comes to choosing the best access path. Application developers know the needs of the users, of which the CBO is

completely unaware. This could lead to a situation where the CBO may be optimizing throughput, when the users would rather have a quick set of results on their screen. By using hints such as FIRST_ROWS_*n*, you can overcome this drawback in the CBO.

• The CBO depends enormously on correct statistics gathering. If the statistics are absent or outdated, the Optimizer can make poor decisions.

Providing Statistics to the CBO

1. Using DBMS_STATS to Collect Statistics
2. Storing the Optimizer Statistics
3. Collecting the Statistics

The Cost Model of the Oracle Optimizer

The cost model of the Optimizer takes into account both I/O cost and CPU cost, both in units of time. The CBO evaluates alternate query costs by comparing the total time it takes to perform all the I/O operations, as well as the number of CPU cycles necessary for the query execution. The CBO takes the total number of I/Os and CPU cycles that will be necessary according to its estimates, and converts them into execution time. It then compares the execution time of the alternative executionpaths and chooses the best candidate for execution.

Collecting Statistics on Dictionary Objects

The two types of dictionary tables are *fixed* and *real*. You can't change or delete dynamic performance tables, which mean they are fixed. Real dictionary tables belong to schemas such as SYS and SYSTEM.

1. Collecting Statistics for Fixed Objects
2. Collecting Statistics for Real Dictionary Tables

Using the OEM to Collect Optimizer Statistics

Here are the steps to collect Optimizer statistics using the Database Control or Grid Control interfaces of the OEM:

1. From the Database Control home page, click the Administration tab.

2. In the Administration page, click the Manage Optimizer Statistics link under the Statistics Management group.

3. You're now in the Manage Optimizer Statistics page. Click the Gather Statistics link to start collecting statistics and follow the instructions for the five steps you must implement.

Writing Efficient SQL

Efficient code means fast performance, and an easy way to decrease the I/O your query requires is to try to lower the number of rows that the Optimizer has to examine. The Optimizer is supposed to find the optimal plan based on your query.

Efficient WHERE Clauses

Careful specification of WHERE conditions can have a significant bearing on whether the Optimizer will choose existing indexes. The principle of *selectivity*—the number of rows

returned by a query as a percentage of the total number of rows in a table—is the key idea here

```
SQL> SELECT * FROM national_employees
WHERE ss_no = 515086789
AND city='DALLAS';
```

(i) Using the Right Joins
(ii) Using the CASE Statement
(iii) Efficient Subquery Execution
(iv) Using WHERE Instead of HAVING
(v) Minimizing Table Lookups

Using Hints to Influence the Execution Plan

Hints can alter the join method, join order, or access path. You can also provide hints to parallelize the SQL statement operations. The following are some of the common hints that you can use in SQL statements:

• ALL_ROWS: The ALL_ROWS hint instructs Oracle to optimize throughput (that is, minimize total cost), not optimize the response time of the statement.

• FIRST_ROWS(*n*): The FIRST_ROWS(*n*) hint dictates that Oracle return the first *n* rows quickly. Low response time is the goal of this hint.

Selecting the Best Join Method

- Avoiding Cartesian Joins
- Nested Loops
- Hash Join
- Merge Join
- Using Bitmap Join Indexes
- Selecting the Best Join Order

Indexing Strategy- Using Appropriate Index Types

1. Bitmap Indexes

Bitmap indexes are ideal for column data that has a *low cardinality*, which means that the indexed column has few distinct values

2. Index-Organized Tables

Index-organized tables where data is stored in the order in which it is inserted. Indexes enable fast access to the rows.

3. Concatenated Indexes

4. Function-Based Indexes

A function-based index contains columns transformed either by an Oracle function or by an expression.

5. Reverse-Key Indexes

6. Partitioned Indexing Strategy

Here's a brief summary of important partitioned indexes:

• *Local partitioned indexes* correspond to the underlying partitions of the table. If you add a new partition to the table, you also add a new partition to the local partitioned index.

• *Global partitioned indexes* don't correspond to the partitions of the local table.

• *Prefixed indexes* are partitioned on a left prefix on the index columns.

• *Nonprefixed indexes* are indexes that aren't partitioned on the left prefix of the index columns.

Monitoring Index Usage- You can use the V$OBJECT_USAGE view to gather index usage information.

1. Removing Unnecessary Indexes
2. Using Similar SQL Statements
3. Avoiding Improper Use of Views
4. Avoiding Unnecessary Full Table Scans

How the DBA Can Help Improve SQL Processing

Performance tuning involves the optimization of SQL code and the calibration of the resources used by Oracle.

Using Partitioned Tables

Partitioned tables usually lead to tremendous improvements in performance, and they're easy to administer.

Using Compression Techniques

The Oracle database lets you use *table compression* to compress tables, table partitions, and materialized views. Table compression helps reduce space requirements for the tables and enhances query performance.

Using Materialized Views

Materialized views are objects with data in them—usually summary data from the underlying tables. Expensive joins can be done beforehand and saved in the materialized
view. When users query the underlying table, Oracle automatically rewrites the query to access the materialized view instead of the tables.

Using Stored Outlines to Stabilize the CBO

You can use Oracle's plan stability feature to ensure that the execution plan remains stable regardless of any changes in the database environment.

Using Parallel Execution

Parallel execution of statements can make SQL run more quickly, and it's especially suitable for large warehouse-type databases.

Other DBA Tasks

- Collecting System Statistics
- Refreshing Statistics Frequently
- Using Histograms

SQL Performance Tuning Tools Using the EXPLAIN PLAN

The EXPLAIN PLAN statement displays execution plans chosen by the Oracle optimizer for SELECT, UPDATE, INSERT, and DELETE statements. A statement's execution plan is the sequence of operations Oracle performs to run the statement.

The row source tree is the core of the execution plan. It shows the following information:

- An ordering of the tables referenced by the statement
- An access method for each table mentioned in the statement
- A join method for tables affected by join operations in the statement
- Data operations like filter, sort, or aggregation

The EXPLAIN PLAN results let you determine whether the optimizer selects a particular execution plan, such as, nested loops join. It also helps you to understand the optimizer decisions, such as why the optimizer chose a nested loops join instead of a hash join, and lets you understand the performance of a query.

How Execution Plans Can Change

With the cost-based optimizer, execution plans can and do change as the underlying costs change. EXPLAIN PLAN output shows how Oracle runs the SQL statement when the statement was explained. This can differ from the plan during actual execution for a SQL statement, because of differences in the execution environment and explain plan environment.

Execution plans can differ due to the following:

- Different Schemas
- Different Costs

Creating the PLAN_TABLE Output Table

Before issuing an EXPLAIN PLAN statement, you must have a table to hold its output. PLAN_TABLE is the default sample output table into which the EXPLAIN PLAN statement inserts rows describing execution plans. Use the SQL script UTLXPLAN.SQL to create the PLAN_TABLE in your schema. The exact name and location of this script depends on your operating system. On Unix, it is located in the $ORACLE_HOME/rdbms/admin directory

Example :Creating a PLAN_TABLE

CONNECT HR/*your_password*
@$ORACLE_HOME/RDBMS/ADMIN/UTLXPLAN.SQL

Table created.

Running EXPLAIN PLAN

To explain a SQL statement, use the following:

EXPLAIN PLAN FOR
SQL_Statement

For example:

EXPLAIN PLAN FOR
SELECT last_name FROM employees;

Using Autotrace

The user can now set the Autotrace feature on and view the EXPLAIN PLAN for any query that is used in the session. The Autotrace feature can be turned on with different options:

• SET AUTOTRACE ON EXPLAIN: This generates the execution plan only and doesn't execute the query itself.

• SET AUTOTRACE ON STATISTICS: This shows only the execution statistics for the SQL statement.

• SET AUTOTRACE ON: This shows both the execution plan and the SQL statement execution statistics.

The SQL Tuning Advisor

It has the following features:

• Advice on improving the execution plan

• Reasons for the SQL improvement recommendations

• Benefits you can expect by following the Advisor's advice

• Details of the commands to tune the misbehaving SQL statements

Using the SQL Tuning Advisor on SQL Statements

It is usual for the Advisor to take SQL statements from places such as these:

• New SQL statements. When working with a development database, this may be your best source of SQL statements.

• High-load SQL statements.

• SQL statements from the AWR.

• SQL statements from the database cursor cache.

How the SQL Tuning Advisor Works

The Oracle Optimizer running in tuning mode is called the Automatic Tuning Optimizer (ATO). The ATO does the following tasks:

• Statistics analysis

• SQL profiling *Dynamic data sampling*: *Partial execution*: *Past execution history statistics*:

• Access path analysis

• SQL structure analysis

Using the DBMS_SQLTUNE Package to Run the SQL Tuning Advisor

You can use the following views to manage your automatic SQL tuning efforts:

• DBA_ADVISOR_TASKS

• DBA_ADVISOR_FINDINGS

• DBA_ADVISOR_RECOMMENDATIONS

• DBA_ADVISOR_RATIONALE

• DBA_SQLTUNE_STATISTICS

• DBA_SQLTUNE_PLANS

A Simple Approach to Tuning SQL Statements

1. Identify Problem Statements
2. Locate the Source of the Inefficiency

3. Tune the Statement
4. Compare Performance

Performance Tuning: Tuning the Instance

An Introduction to Instance Tuning

Performance tuning an Oracle database instance involves tuning memory and I/O as well as operating system resources such as CPU, the operating system kernel, and the operating system memory allocation. There are two big advantages to being in a proactive mode of tuning. First, you have fewer sudden performance problems that force hurried reactions. Second, as your understanding of your system increases, so does your familiarity with the various indicators of poor performance and the likely causes for them, so you can resolve problems that do occur much more quickly

Automatic Performance Tuning vs. Dynamic Performance Views

Here's a brief summary of the automatic performance tuning features:

• The AWR collects all the performance data necessary for tuning as well as diagnosing instance problems.

• The ADDM automatically diagnoses database performance by analyzing the AWR data.

• The SQL Tuning Advisor provides SQL tuning recommendations.

• The database automatically runs the statistics collection job, thus keeping all statistics up to date.

• The Segment Advisor runs automatically during the maintenance interval and makes recommendations about which segments to shrink and which to reorganize (for example, due to excessive row chaining).

• The SQL Access Advisor provides recommendations about the ideal indexes and materialized views to create.

• The Memory Advisor, MTTR Advisor, and the Undo Advisor help you tune memory, redologs, and undo segments, respectively. However, the best way to diagnose and tune Oracle performance issues is through the OEM Database Control

Tuning Oracle Memory

You can easily adjust the memory allocation of Oracle, by simply changing a single initialization parameter—SGA_TARGET.

1. Tuning the Shared Pool

An improperly tuned shared pool leads to problems such as the following:

• Fragmentation of the pool

• Increased latch contention with the resulting demand for more CPU resources

• Greater I/O because executable forms of SQL aren't present in the shared pool

• Higher CPU usage because of unnecessary parsing of SQL code

The shared pool consists of two major areas: the library cache and the data dictionary cache.

2. Hard Parsing and Soft Parsing

During a hard parse, Oracle performs syntactic and semantic checking, checks the object and system privileges, builds the optimal execution plan, and finally loads it into the library cache. A hard parse involves a lot more CPU usage and is inefficient compared to a soft parse, which depends on reusing previously parsed statements.

3. Tuning the Buffer Cache

1. How to Size the Buffer Cache- *Physical reads*: *DB block gets*: *Consistent gets*: *Logical reads*: *Buffer gets*:

2. Using Multiple Pools for the Buffer Cache

4. Tuning the Large Pool, Streams Pool, and Java Pool

5. Tuning PGA Memory

Evaluating System Performance

System performance includes the CPU performance, memory usage, and disk I/O.

CPU Performance

You can use operating system utilities such as System Activity Reporter (sar) or vmstat to find out how the CPU is performing.

What Is the CPU Time Used For?

CPU time is generally understood as the processor time taken to perform various tasks, such as the following:

- Loading SQL statements into the library cache
- Searching the shared pool for parsed versions of SQL statements
- Parsing the SQL statements
- Querying the data dictionary
- Reading data from the buffer cache
- Traversing index trees to fetch index keys

Memory

Operating system physical memory holds all the data and programs by loading them from disk. System CPU executes programs only if they're loaded into the physical memory

The space for the virtual memory is called *swap space*. When the system needs room in the physical or main memory, it "swaps out" some programs to the swap area, thus freeing up additional physical memory for an executing program. The operating system swaps out data in units called *pages*, which are the smallest units of memory that can be used in transferring memory back and forth between physical memory and the swap area. One of the best ways to check operating system memory performance is by using the vmstat utility,

Disk I/O

Important factors that have a bearing on your I/O are as follows:

1. Choice of RAID configuration:
2. Raw devices or operating system file systems:

3. I/O size:
4. Logical volume stripe sizes:
5. Number of controllers and disks:
6. Distribution of I/O

Measuring I/O Performance

Several operating system utilities are easy to use and give you information about how busy your disks are.

Is the I/O Optimally Distributed?

If the numberof waits is higher than the number of CPUs, or if the service times are high (say, greater than 20 milliseconds), then your system is facing contention at the I/O level. One of the most useful pieces of information you can get is by using the sar –d command to find out if you're using any of your disks excessively compared to other disks in the system.

Reducing Disk Contention

If there's severe I/O contention in your system, you can undertake some of the following steps, depending on your present database configuration:

• Increase the number of disks in the storage system.
• Separate the database and the redo log files.
• For a large table, use partitions to reduce I/O.
• Stripe the data either manually or by using a RAID disk-striping system.
• Invest in cutting-edge technology, such as file caching, to avoid I/O bottlenecks.

Measuring Instance Performance

Oracle Database 10*g* uses the concept of DB time (discussed in detail in Chapter 17) to determine how well the instance is performing. You can look at some statistics to see how well the database is performing. These statistics fall into two groups: database hit ratios and database wait statistics.

Database Hit Ratios

Database hit ratios are the most commonly used measures of performance. These include the buffer cache hit ratio, the library cache and dictionary cache hit ratios, the latch hit ratio, and the disk sort ratios.

Database Wait Statistics

Four dynamic performance views contain wait information: V$SESSION, V$SYSTEM_EVENT, V$SESSION_EVENT, and V$SESSION_WAIT. These four views list just about all the events the instance was waiting for and the duration of these waits. Understanding these wait events is essential for resolving performance issues.

Wait Events and Wait Classes

Any time a server process waits for an event to complete, it's classified as a *wait event*.

A wait class is a grouping of related wait events, and every wait event belongs to a wait class. Important wait classes include Administrative, Application, Concurrency, Configuration, Idle, Network, System I/O, and User I/O.

Analyzing Instance Performance

1. The following columns from the V$SESSION_WAIT view are important for troubleshooting performance issues:

• EVENT: These are the different wait events described in the next section (for example, latch free and buffer busy waits).

• P1, P2, P3: These are the additional parameters that represent different items, depending on the particular wait event. For example, if the wait event is db file sequential read, P1 stands for the file number, P2 stands for the block number, and P3 stands for the number of blocks. If the wait is due to a latch free event, P1 stands for the latch address, P2 stands for the latch number, and P3 stands for the number of attempts for the event.

• WAIT_CLASS_ID: Identifies the wait class.

• WAIT_CLASS#: Number of the wait class.

• WAIT_CLASS: Name of the wait class.

Using V$ Tables for Wait Information

2. Obtaining Wait Information

3. The V$SESSION_WAIT_HISTORY View

4. Using the V$ACTIVE_SESSION_HISTORY View

Examining System Performance

Here are some of the key system usage statistics:

• NUM_CPUS: Number of processors.

• IDLE_TICKS: Number of hundredths of a second that all processors have been idle.

• BUSY_TICKS: Number of hundredths of a second that all processors have been busy executing code.

• USER_TICKS: Number of hundredths of a second that all processors have been busy executing user code.

• SYS_TICKS: Number of hundredths of a second that all processors have been busy executing kernel code.

• IOWAIT_TICKS: Number of hundredths of a second that all processors have been waiting for I/O to complete.

System Usage Problems

Here are some of the important things you need to examine:

• Make sure your system isn't suffering from a severe paging and swapping problem, which could result in a slower-performing database.

• Use top, sar, vmstat, or similar operating-system–level tools to check resource usage. Large queries, sorting, and space management operations could all lead to an increase in CPU usage.

• Runaway processes and excessive snapshot (SNP) processes could gobble excessive CPU resources.

A Simple Approach to Instance Tuning

The following sections present a brief summary of how you can start analyzing the instance to find out where the problem lies. First, examine all the major resources such as the memory, CPUs, and storage subsystem to make sure your database isn't being slowed down by bottlenecks in these critical areas.

1. What's Happening in the Database?
2. Using the OEM Database Control to Examine
3. Database Performance
4. Are There Any Long-Running Transactions?
5. Is Oracle the Problem?
6. Is the Network Okay?
7. Is the System CPU Bound?
8. Is the System I/O Bound?
9. Is the Database Load Too High?
10. Checking Memory-Related Issues
11. Are the Redo Logs Sized Correctly?
12. Is the System Wait Bound?
13. The Compare Periods Report
14. Eliminating the Contention

POSSIBLE QUESTIONS:

PART B: 6 MARKS

1. State DBA's role to improve SQL processing
2. Write a brief note on oracle optimization and oracle cost based optimizer
3. State at least six ways for writing efficient SQL queries.
4. Discuss in brief about SQL performance tuning tool EXPLAIN PLAN.
5. Describe the process of using SQL tuning advisor
6. Explain the process of tuning oracle memory
7. Discuss in detail about measuring IO performance.
8. Hoe to evaluate system performance in oracle? Explain.
9. Describe the simple approach for instance tuning.
10. Describe the simple approach to tuning SQL statement.

Karpagam Academy of Higher Education
Department of CS, CA & IT
Subject: Oracle 10g Administration (17CSP203)

Batch : 2017-2019 **Class: I M.Sc CS**

Objective Type Questions
UNIT V

S.NO	QUESTIONS	OPTION 1	OPTION 2	OPTION 3	OPTION 4	KEY
1	Performance tuning is _____ process	sequential	iterative	linear	ordering	iterative
2	_____is the process of choosing the most efficient execution plan	Query Processing	Query Transaction	Query execution	query loading	Query optimization
3	_____consists of checking the syntax and semantics of the SQL statements.	Parsing	Executing	Ordering	Loading	Parsing
4	The_____ procedure collects Optimizer statistics for the entire database.	GATHER_STATISTICS	GATHER_FULLDATABASE_STATISTICS	GATHER_EXPORTDATABASE_STATISTICS	GATHER_DATABASE_STATISTICS	GATHER_DATABASE_STATISTICS
5	You can provide_____ to the Optimizer to override the CBO's execution plans	hints	clues	queries	models	hints
6	_____ can alter the join method, join order, or access path.	hints	clues	queries	models	hints
7	_____hint forces the join order for the tables in the query.	INDEX	ORDERED	FULL	CLUES	ORDERED
8	_____ prestore the results of a join between two tables in an index	Best join order	best index order	Bitmap join indexes	bitmap index order	Bitmap join indexes
9	A _____contains columns transformed either by an Oracle function or by an expression.	Best join order	best index order	Bitmap join indexes	function-based index	function-based index
10	_____don't correspond to the partitions of the local table.	Prefixed indexes	Global partitioned indexes	Local partitioned indexes	Nonprefixed indexes	Global partitioned indexes

11	_____ is used to compress the primary key columns of IOTs.	index key compression	bitmap key compression	global key compression	local key compression	index key compression
12	The output of the EXPLAIN PLAN tool goes into _____ table	work_table	plan_table	future_table	workplan_table	plan_table
13	Any SQL statements can override theinstance-or session level settings with the use of _____	optimizer mode	optimizer plan	optimizer hints	optimizer tools	optimizer hints
14	The _____ attribute refers to the percentage of rows that should be used to esitmate the statistics.	ESTIMATE_STAT	ESTIMATE_PERCENT	ESTIMATE_ROWS	ESTIMATE_VALUES	ESTIMATE_PERCENT
15	The _____ determines the way the query optimizer performs optimization throughout the database	OPTIMIZER_MODE	OPTIMIZER_PLAN	OPTIMIZER_HINTS	OPTIMIZER_TOOLS	OPTIMIZER_MODE
16	The _____ feature lets oracle use a composite index even when the leading column isn't used in the query	index query	index skip scan	index increment scan	index full	index skip scan
17	A _____ index contains columns transformed either by an oracle function or by an expression	index based	key based	function based	hash based	function based
18	You can use _____ compression to compress the primary key columns of IOTs.	function key	index key	hash key	table key	index key
19	_____ histograms determines the number of buckets based on the distinct values in the column.	height based	frequeny based	fucntion based	hash based	frequeny based
20	You can use _____ package to display the output of an EXPLAIN PLAN statement in a readable format	DBMS_XPLAN	DBMS_RESULTS	DBMS_OUTPUT	DBMS_PLAN_OUTPUT	DBMS_XPLAN
21	_____ is the directory on your server where your SQL trace files will be sent.	USER_DUMP_DEST	USER_TRACE_FILE	USER_DBMS_FILES	USER_SQL_FILES	USER_DUMP_DEST

22	The _____ holds the parsed and executable versions of SQL and PL/SQL code	library cache	system cache	fetching	reordering	library cache
23	_____ includes syntatic and semantic verfication of SQL statements and checking of object privileges to perform the actions	Parsing	optimization	execution	fetching	Parsing
24	In _____, the SQL statements has to be reloaded into the shared pool and parsed completely	soft parse	hard parse	light parse	medium parse	hard parse
25	The _____ view provides the information about the parse time savings you can expect for various sizes of the shared pool	V$_SHARE	V$SHARED_POOL_ADVICE	V$LIBRARY_CACHE_MEMORY	V$LIBRARY_MEMORY	V$SHARED_POOL_ADVICE
26	Use the _____ initalization parameter to reuse the open cursors in a session	SESSION_CURSORS	SESSION_CACHED_CURSORS	SESSION_SPACE_CURSOR	SESSION_SPACE	SESSION_CACHED_CURSORS
27	_____ indicates the percentage of time the system was waiting for I/O	%idle	%usr	%wio	%sys	%wio
28	_____ shows the proportion of time the sytem itself was using the CPU	%idle	%usr	%wio	%sys	%sys
29	_____ is the proportion of time the CPU was idle	%idle	%usr	%wio	%sys	%idle
30	_____shows the proportion of total CPU time taken up by the various users of the sytem	%idle	%usr	%wio	%sys	%usr
31	The space for the virtual memory is called _____	swap space	idle space	virtual space	free space	swap space
32	The operating system swaps out data in units called _____	hints	pages	freespace	index	pages
33	_____ is a function of the stripe depth and the number of drives in the striped sets.	stripe size	stripe height	stripe volume	stripe length	stripe size

34	The _____ facility helps to tune SQL by letting to see the execution plan selected by the oracle optimizer	EXPLAIN PLAN	ASH	AWR	TRACE	EXPLAIN PLAN
35	The _____ tool indicates clearly whether the optimizer is using an index	EXPLAIN PLAN	ASH	AWR	TRACE	EXPLAIN PLAN
36	The _____ script is an alternative to using DBMS_XPLAN package directly	utl.sql	catproc.sql	utlxpls.sql	plan.sql	utlxpls.sql
37	The oracle 10g has _____ tracing with which you can uniquely identify and track the same client through multiple sessions.	user to user	end to end	session to session	table to table	end to end
38	_____ is the total parse and execution time	rows_process	buffer_gets	cpu_time	parse_call	cpu_time
39	The _____ view helps to find out which query have high logical I/Oand high physical I/O	V$SQL	V$QUERY	V$HIGH	V$I/O	V$SQL
40	The _____ automatically diagnoses database performance by analyzing the AWR data	ASH	ADDM	AWT	MTTR	ADDM
41	____ variables allow binding of application data to the SQL statement	hints	sql	bind	order	bind
42	_____ are the data blocks that oracle reads from disk	DB blocks	physical reads	consistent gets	logical reads	physical reads
43	_____ term refers to the number of database cache buffers retrieved.	physical reads	logical reads	buffer gets	DB block	buffer gets
44	Oracle 8.1 version has used a concept called _____ to measure how many times an object is accessed in the buffer cache	buffer count	memory count	touch count	query count	touch count
45	The _____ parameter in the init.ora file sets the maximum limit on the total memory allocated to the PGA	PGA_MEMORY	PGA_AGGREGATE_TARGET	PGA_TARGET_DATABASE	PGA_SPACE_MEMORY	PGA_AGGREGATE_TARGET

46	The goal of the _____ disk storage strategy is to eliminate I/O hot spots and maximize I/O bandwidth	SWAP	SAME	STRIPE	SMALL	SAME
47	The _____ view holds information about the last ten wait events for each active session.	V$_SESSION_WAIT	V$SESSION_WAIT_HISTORY	V$SESSION_HISTORY	V$SESSION_ACTIVE_HISTORY	V$SESSION_WAIT_HISTORY
48	The _____ wait event indicates that full table scans are occuring in the database.	db file scan	db file scan read	db file scattered read	db file full scan	db file scattered read
49	The _____ events are waits that occur while performing a direct read into the PGA, bypassing the SGA buffer cache.	direct path read	read	direct read	indirect path read	direct path read
50	_____ are similar to locks in that they are internal mechanisms that control access to resources	latches	enqueues	streaming	processing	enqueues
51	_____ are internal serialization mechanisms used to protect shared data structures in oracle's SGA	latches	enqueues	streaming	processing	latches
52	Expand CBO	Cost based optimizer	Commerce based optimizer	Calculate based optimizer	Customer Based optimizer	Cost based optimizer
53	_____ indexes are ideal for column data that has a low cardinality	Concatenated	Reverse-Key	Bitmap	Function-Based	Bitmap

Register Number_____

[17CSP203]

KARPAGAM ACADEMY OF HIGHER EDUCATION

(Deemed to be University Established under Section 3 of UGC Act 1956)

M.Sc Computer Science

FIRST INTERNAL TEST- JANUARY 2018

ORACLE 10G ADMINISTRATION

CLASS: I MSc (CS) **Time:2 Hours**

DATE & SESSION: 31.01.2018, FN **Maximum: 50 marks**

PART – A (20 x 1 = 20 Marks)

Answer ALL the Questions

1. Decision support system is also known as _____.
   a. OLTP b. OLAP c. OLDP d. OLTM
2. A/An _____ consists of files, both data files and Oracle system files.
   a. production database c. Oracle database
   b. test databases d. execution database
3. A _____consists of a number of bytes of disk space in the operating system's storage system.
   a. extends b. data block c. segments d. tablespaces
4. An/A _____ is two or more contiguous Oracle data blocks.
   a. extends b. data block c. segments d. tablespaces
5. A ___ is a set of extents that you allocate to a logical structure like a table or an index.
   a. extends b. data block c. segments d. tablespaces
6. The __ file is a file that the Oracle maintains to manage the state of the database.
   a. Control b. data c. redo d. log
7. The _____ is a purely logical construct and is the primary logical storage structure of an Oracle database.
   a. extends b. data block c. segments d. tablespaces
8. You can create a new init.ora file from the SPFILE in the default location by using the following command:
   a. create init.ora from spfile c. create init from spfile
   b. create pfile from spfile; d. create pfile from spfile;
9. You can use the _____ parameter to embed another initialization file in it.
   a. IFILE c. DB_FILES
   b. CONTROL_FILES d. USER_FILE
10. _____ contain data that was read from disk and then modified, but hasn't yet been written to the data files on disk.
    a. shared buffer b. Pinned buffers c. dirty buffers d. free buffers
11. _____tablespaces are used to store objects for the duration of a user's session only
    a. undo b. temporary c. permanent d. locally

12. _____ Tablespaces are a type of a tablespace that are used to store undo data.
    a. undo b. temporary c. permanent d. locally
13. _____ indexes are the unique indexes in a table that must always posses a value
    a. secondary b. non unique c. primary d. composite
14. _____ are static objects that derive their data from the underlying base tables.
    a. view c. index
    b. materialized views d. functions
15. If a transaction encounters a _____ statement, all the changes to that point are made permanent in the database.
    a. ROLLBACK b. COMMIT c. RECOVERY d. BACKUP
16. You can partially rollback the effects of a transaction by using _____
    a. SAVEPOINT c. WAIT
    b. BACK d. NOWAIT
17. To remove the tablespaces from the database use the command_____
    a.REMOVE b.DROP c.DELETE d.ALTER
18. You can use the new _____ data dictionary view to manage the temporary tablespace groups in your database
    a. DBA_TABLESPACE_GROUPS c. DBA_TABLES
    b. DBA_TABLESPACE_VIEW d. DBA_GROUPS
19. Oracle keeps track of how much free space is in its data blocks by maintaining __.
    a. buffers b. freelists c. freespace d. freeorder
20. _____ Indexes are structured in the form of an inverse tree.
    a. primary c. B-tree
    b. composite d. h-tree

PART – B (3 x 2 = 6 Marks)

Answer ALL the Questions

21. What are the three major tasks performed by a DBA?
22. What is an SPFILE?
23. Write a statement to create a permanent tablespace with default options.

PART – C (3 x 8 = 24 Marks)

Answer ALL the Questions

24. (a) Write about the role of DBA in various circumstances **(OR)**
    (b) List out the Oracle Background processes and explain their usage.
25. (a) Describe the Oracle Memory Structures in detail. **(OR)**
    (b) Write in detail about creating and altering tablespaces with example.
26. (a) Explain in detail about Indexes and Materialized Views in Oracle **(OR)**
    (b) Outline the transaction management concepts in Oracle.

Register Number_____

[17CSP203]

KARPAGAM ACADEMY OF HIGHER EDUCATION

(Deemed to be University Established under Section 3 of UGC Act 1956)

M.Sc Computer Science

FIRST INTERNAL TEST- JANUARY 2018

ORACLE 10G ADMINISTRATION

CLASS: I MSc (CS) **Time:2 Hours**

DATE & SESSION: 31.01.2018, FN **Maximum: 50 marks**

ANSWER KEY

PART – A (20 x 1 = 20 Marks)

Answer ALL the Questions

1. Decision support system is also known as _____.

   a. OLTP **b. OLAP** c. OLDP d. OLTM

2. A/An _____ consists of files, both data files and Oracle system files.

   a. production database **c. Oracle database**

   b. test databases d. execution database

3. A _____consists of a number of bytes of disk space in the operating system's storage system.

   a. extends **b. data block** c. segments d. tablespaces

4. An/A _____ is two or more contiguous Oracle data blocks.

   a. extends b. data block c. segments d. tablespaces

5. A ___ is a set of extents that you allocate to a logical structure like a table oran index.

   a. extends b. data block **c. segments** d. tablespaces

6. The __ file is a file that the Oracle maintains to manage the state of the database.

   a. Control b. data c. redo d. log

7. The _____ is a purely logical construct and is the primary logical storage structure of an Oracle database.

   a. extends b. data block c. segments **d. tablespaces**

8. You can create a new init.ora file from the SPFILE in the default location by using the following command:

   a. create init.ora from spfile c. create init from spfile

   b. create pfile from spfile; **d. create pfile from spfile;**

9. You can use the _____ parameter to embed another initialization file in it.

a. IFILE c. DB_FILES

b. CONTROL_FILES d. USER_FILE

10. _____ contain data that was read from disk and then modified, but hasn't yet been written to the data files on disk.

a. shared buffer b. Pinned buffers **c. dirty buffers** d. free buffers

11. _____tablespaces are used to store objects for the duration of a user's session only

a. undo **b. temporary** c. permanent d. locally

12. _____ Tablespaces are a type of a tablespace that are used to store undo data.

a. undo b. temporary c. permanent d. locally

13. _____ indexes are the unique indexes in a table that must always posses a value

a. secondary b. non unique **c. primary** d. composite

14. _____ are static objects that derive their data from the underlying base tables.

a. view c. index

b. materialized views d. functions

15. If a transaction encounters a _____ statement, all the changes to that point are made permanent in the database.

a. ROLLBACK **b. COMMIT** c. RECOVERY d. BACKUP

16. You can partially rollback the effects of a transaction by using _____

a. SAVEPOINT c. WAIT

b. BACK d. NOWAIT

17. To remove the tablespaces from the database use the command_____

a.REMOVE **b. DROP** c.DELETE d.ALTER

18. You can use the new _____ data dictionary view to manage the temporary tablespace groups in your database

a. DBA_TABLESPACE_GROUPS c. DBA_TABLES

b. DBA_TABLESPACE_VIEW d. DBA_GROUPS

19. Oracle keeps track of how much free space is in its data blocks by maintaining __.

a. buffers **b. freelists** c. freespace d. freeorder

20. _____ Indexes are structured in the form of an inverse tree.

a. primary b. composite **c. B-tree** d. h-tree

PART – B (3 x 2 = 6 Marks)

Answer ALL the Questions

21. What are the three major tasks performed by a DBA?

The main responsibility of a DBA is to make corporate data available to the end users and the decision makers of an organization.

• *Security:* Ensuring that the data and access to the data are secure

• *Backup:* Ensuring that the database can be restored in the event of either human or systems failure

• *Performance:* Ensuring that the database and its subsystems are optimized for performance

• *Design:* Ensuring that the design of the database meets the needs of the organization

• *Implementation:* Ensuring proper implementation of new database systems and applications.

In a small organization a DBA could be managing the entire information technology (IT) infrastructure, including the databases, whereas in a large organization there could be a number of DBAs, each charged with managing a particular area of the system.

22. What is an SPFILE?

A server parameter file is basically a repository for initialization parameters. Initialization parameters stored in an SPFILE are persistent, meaning any parameter changes made while an instance is running can persist across instance shutdown and startup. A server parameter file is initially built from the traditional text initialization parameter file, using the create SPFILE statement. It is a binary file that cannot be browsed or edited with a text editor.

23. Write a statement to create a permanent tablespace with default options.

```
CREATE TABLESPACE lmtbsb
DATAFILE '/u02/oracle/data/lmtbsb01.dbf' SIZE 50M
EXTENT MANAGEMENT LOCAL
SEGMENT SPACE MANAGEMENT AUTO;
```

PART – C (3 x 8 = 24 Marks)

Answer ALL the Questions

24. (a) Write about the role of DBA in various circumstances

The Oracle DBA's Role

The main responsibility of a DBA is to make corporate data available to the end users and the decision makers of an organization.

• *Security:* Ensuring that the data and access to the data are secure

• *Backup:* Ensuring that the database can be restored in the event of either human or systems failure

• *Performance:* Ensuring that the database and its subsystems are optimized for performance

• *Design:* Ensuring that the design of the database meets the needs of the organization

• *Implementation:* Ensuring proper implementation of new database systems and applications.

In a small organization a DBA could be managing the entire information technology (IT) infrastructure, including the databases, whereas in a large organization there could be a number of DBAs, each charged with managing a particular area of the system.

The DBA's Security Role

- Protecting the Database – The Oracle DBA is the person the information departments entrust with safeguarding the organization's data, and this involves preventing unauthorized use of and access to the database.
- Monitoring the System- The tasks involved in monitoring the system include the following:
    - Monitoring space in the database to ensure it is sufficient for the system
    - Checking to ensure that batch jobs are finishing as expected
    - Monitoring log files on a daily basis for evidence of unauthorized attempts to log in (something DBAs want to keep close tabs on)
- Creating and Managing Users - Every database has users, and it's the DBA's job to create them based on requests from the appropriate people.

The DBA's System Management Role

The following sections describe the various facets of the system management part of the Oracle DBA's job.

1. Troubleshooting
2. Ensuring Performance Tuning - all database tuning efforts can be grouped into two classes—proactive and reactive tuning.
3. Minimizing Downtime
4. Estimating Requirements – estimation of physical requirements, planning for future growth.
5. Developing Backup and Recovery Strategies
6. Loading Data
7. Overseeing Change Management

The DBA's Database Design Role

1. Designing the Database
2. Installing and Upgrading Software
3. Creating Databases
4. Creating Database Objects

DBA Job Classifications

A DBA's job description is not exactly the same in all organizations. There are several variations in the job's classification and duties across organizations.

***Production* DBA** refers to database administrators in charge of production databases. DBAs who are involved in the preproduction design and development of databases are usually called ***development* or *logical* DBAs.**

(OR)

(b) List out the Oracle Background processes and explain their usage.

The Background Processes

The *background processes* are the real workhorses of the Oracle instance—they enable large numbers of users to concurrently and efficiently use information stored in database files. Each of the Oracle background processes is in charge of a separate task, thus increasing the efficiency of the database instance. These processes are automatically created by Oracle when you start the database instance, and they terminate when the database is shut down.

Background Process	Function
Database writer	Writes modified data from the buffer cache to disk (data files)
Log writer	Writes redo log buffer contents to the online redo log files
Checkpoint	Updates the headers of all data files to record the checkpoint details
Process monitor	Cleans up after finished and failed processes
System monitor	Performs crash recovery and coalesces extents
Archiver	Archives filled online redo log files
Manageability Monitor	Performs database-manageability-related tasks
Manageability Monitor Light	Performs tasks like capturing session history and metrics
Memory manager	Coordinates the sizing of the SGA components
Job queue coordination process	Coordinates job queues to expedite job processes
Database writer	Writes modified data from the buffer cache to disk (data files)
Log writer	Writes redo log buffer contents to the online redo log files

The Database Writer

Oracle doesn't modify data directly on the disks—all modifications of data take place in Oracle memory. The database writer (DBWn) process is then responsible for writing the "dirty" (modified) data from the memory areas known as database buffers to the actual data files on disk. It is the database writer process's job to monitor the use of the database buffer cache, and if the free space in the database buffers is getting low, the database writer process makes room available by writing some of the data in the buffers to the disk files. The database writer process uses the least recently used (LRU) algorithm (or a modified version of it), which retains data in the memory buffers based on how long it has been since someone asked for that data. If a piece of data has been requested very recently, it's more likely to be retained in the memory buffers. The database writer process writes dirty buffers to disk under the following conditions:

1. When the database issues a checkpoint
2. When a server process can't find a clean reusable buffer after checking a threshold number of buffers
3. Every 3 seconds

The Log Writer

The job of the log writer (LGWR) process is to transfer the contents of the redo log buffer to disk. Whenever you make a change to a database table (whether an insertion, update, or deletion), Oracle writes the committed and uncommitted changes to a redo log buffer (memory buffer). The log writer process then transfers these changes from the redo log buffer to the redo log files on disk. The log writer writes a commit record to the redo log buffer and writes it to the redo log on disk immediately, whenever a user commits a transaction. The log writer writes all redo log buffer entries to the redo logs under the following circumstances:

• Every 3 seconds.
• When the redo log buffer is one-third full.
• When the database writer signals that redo records need to be written to disk. Under Oracle's write-ahead protocol, all redo records associated with changes in the block buffers must be written to disk (that is, to the redo log files on disk) before the data files on disk can be modified. While writing dirty buffers from the buffer cache to the storage disks, if the database writer discovers that certain redo information has not been written to the redo log files, it signals the log writer to first write that information, so it can write its own data to disk. The redo log files, as you learned earlier, are vital during the recovery of an Oracle database from a lost or damaged disk.

The Checkpoint

The checkpoint (CKPT) process is charged with telling the database writer process when to write the dirty data in the memory buffers to disk. After telling the database writer process to write the changed data, the checkpoint process updates the data file headers and the control file to indicate when the checkpoint was performed. The purpose of the checkpoint process is to synchronize the buffer cache information with the information on the database disks. Each checkpoint record consists of a list of all active transactions and the address of the most recent log record for those transactions. A checkpointing process involves the following steps:

1. Flushing the contents of the redo log buffers to the redo log files
2. Writing a checkpoint record to the redo log file
3. Flushing the contents of the database buffer cache to disk
4. Updating the data file headers and the control files after the checkpoint completes

There is a close connection between how often Oracle checkpoints and the recovery time after a database crash. Because database writer processes write all modified blocks to disk at checkpoints, the more frequent the checkpoints, the less data will need to be recovered when the instance crashes. However, checkpointing involves an overhead cost. Oracle lets you configure the database for automatic checkpoint tuning, whereby the database server tries to write out the dirty buffers in the most efficient way possible, with the least amount of adverse impact on throughput and performance. If you use automatic checkpoint tuning, you don't have to set any checkpoint-related parameters.

The Process Monitor

When user processes fail, the process monitor (PMON) process cleans up after them, ensuring that the database frees up the resources that the dead processes were using. For example, when a user process dies while holding certain table locks, the PMON process releases those locks so other users can use the tables without any interference from the dead process. In addition, the PMON process restarts failed server processes and dispatcher processes. The PMON process sleeps most of the time, waking up at regular intervals to see if it is needed. Other processes will also wake up the PMON process if necessary.

The PMON process automatically performs dynamic service registration. When you create a new database instance, the PMON process registers the instance information with the listener, which is the entity that manages requests for database connections. This dynamic service registration eliminates the need to register the new service information in the listener.ora file, which is the configuration file for the listener. The System Monitor

The system monitor (SMON) process, as its name indicates, performs system-monitoring tasks for the Oracle instance, such as these:

• Upon restarting an instance that crashed, SMON determines whether the database is consistent.
• SMON coalesces free extents if you use dictionary-managed tablespaces, which enables you to assign larger contiguous free areas on disk to your database objects.
• SMON cleans up unnecessary temporary segments.
Like the PMON process, the SMON process sleeps most of the time, waking up to see if it is needed. Other processes will also wake up the SMON process if they detect a need for it.

The File Mapping Monitor

File systems are increasingly complex, and to help you in monitoring I/O, Oracle provides the file mapping monitor (FMON) process to map files to immediate storage layers and physical devices. This will help you understand exactly how your data files are stored in a disk system managed by a Logical Volume Manager (LVM).
The FMON process interacts with mapping libraries provided by the operating system to perform the file mapping. The results are in the DBMS_STORAGE_MAP view.

The Archiver

The archiver (ARCn) process is used when the system is being operated in an archivelog mode— that is, the changes logged to the redo log files are being saved and not being overwritten by new changes. If you run your database in the no archivelog mode, Oracle will overwrite the redo log files with new redo log records. When you choose to run the instance in an archivelog mode, no such overwriting can take place—each filled log will be saved or archived in a special location. The archiver process will archive the redo log files to the location you specify. You usually copy these archived logs to tape and send them to an offsite storage location to ensure you have a complete set of backups and archived redo logs so that you can perform a database recovery if the need arises

25. (a) Describe the Oracle Memory Structures in detail.

Oracle Memory Structures

Oracle uses two kinds of memory structures, one shared and the other process-specific. The *system global area* (SGA) is the part of total memory that all server processes (including background processes) share. The process-specific part of the memory is known as the *program global area*(PGA), or *process-private memory*..

What is the SGA

The system global area (SGA) is just shared memory structures that are created at instance startup, hold information about the instance and control its behavior. The following table gives a brief synopsis of the particular components of the SGA, the variables that control the size of memory allocated, some of the areas of the Oracle server the particular component has an influence on, and then a very brief description.

Components of the SGA

SGA COMPONENT	SIZE CONTROLED BY	AREAS OF INFLUENCE	SIMPLE DESCRIPTONS
Shared Pool	SHARED_POOL_SIZE	Library Cache • Shared SQL areas • Private SQL areas • PL/SQL procedures and packages • Various control structures	Oracle needs to allocate & deallocate memory as SQL or procedural code is executed based on the individual needs of users' sessions and in accordance to the LRU algorithm.
		Dictionary Cache • Row cache • Library cache	Highly accessed memory structures that provide information on object structures to SQL statements being parsed.
Java Pool	JAVA_POOL_SIZE	• Run state • Methods • Classes • Session code • Data in JVM	Memory available for the Java memory manager to use for all things Java.
Streams Pool	STREAMS_POOL_SIZE	• Stream activity	New to Oracle 10g, memory available for stream processing.
Redo Log Buffer	LOG_BUFFER	• Redo entries	Holds changes made to data and allows for the reconstruction of data in the case of failure.
Database Buffer Cache	DB_2K_CACHE_SIZE DB_4K_CACHE_SIZE DB_8K_CACHE_SIZE DB_16K_CACHE_SIZE DB_32K_CACHE_SIZE DB_KEEP_CACHE_SIZE DB_RECYCLE_CACHE_SIZE	• Write list • LRU list	Holds copies of data requested by SQL and reduces requests to disk by having data in memory. You may have many different buffer caches that help segregate on usage patterns.
Large Pool	LARGE_POOL_SIZE	• Shared server • Oracle XA • I/O server processes • Backup & restore	For large memory allocations.

The Program Global Area (PGA)

A **program global area (PGA)** is a memory region that contains data and control information for a server process. It is a nonshared memory created by Oracle when a server process is started.

You can classify the PGA memory into the following types:

• Private SQL area: This area of memory holds SQL variable bind information and runtime memory structures. Each session that executes a SQL statement will have its own private SQL area.

• Runtime area: The runtime area is created for a user session when the session issues a

SELECT, INSERT, UPDATE, or DELETE statement. After an INSERT, DELETE, or UPDATE statement is run, or after the output of a SELECT statement is fetched, the runtime area is freed by Oracle.

The size and content of the PGA depends on the Oracle-server options installed. This area consists of the following components:

- stack-space: the memory that holds the session's variables, arrays, and so on.
- session-information: unless using the multithreaded server, the instance stores its session-information in the PGA. (In a multithreaded server, the session-information goes in the SGA.)
- private SQL-area: an area in the PGA which holds information such as bind-variables and runtime-buffers.
- sorting area: an area in the PGA which holds information on sorts, hash-joins, etc.

The sum of all the PGA memory used by all sessions makes up the PGA used by the instance. Oracle recommends that you use *automatic PGA management*, which automates the allocation of PGA memory.

(OR)

(b) Write in detail about creating and altering tablespaces with example.

Creating and Managing Tablespaces

Tablespaces are logical entities—each of an application's tables and indexes are stored as a segment, and the segments are stored in the data files that are parts of tablespaces. A tablespace is thus a logical allocation of space for Oracle schema objects. The following are the important types of Oracle tablespaces:

• Temporary tablespaces are used to store objects for the duration of a user's session only. You use temp files to create a temporary tablespace, instead of data files.

• Undo tablespaces are a type of permanent tablespace that are used to store undo data, which is used to undo changes to data.

You can create two basic types of tablespaces in an Oracle database, which differ by how they manage the database extents: *locally managed* and *dictionary-managed* tablespaces.

Locally and Dictionary-Managed Tablespaces

The basic unit of space allocation in Oracle databases, and dictionary managed tablespaces store extent information in the data dictionary. Locally managed tablespaces, on the other hand, manage extents by referring to the bitmaps kept in

each physical data file header for all the blocks within that data. Locally managed tablespaces have several advantages over the traditional dictionary-managed tablespaces. Dictionary-managed tablespaces have to constantly check the data dictionary during the course of extent management—whenever an extent is allocated to an object or reclaimed from an object, Oracle will update the relevant tables in the data dictionary.

Allocating the Extent Size: Autoallocate vs.Uniform

You create a locally managed tablespace by specifying LOCAL in the EXTENT MANAGEMENT clause of the CREATE TABLESPACE statement. This is the default for new permanent tablespaces, but you must specify if it you want to specify the management of the locally managed tablespace. You can have the database manage extents for you automatically with the AUTOALLOCATE clause (the default), or you can specify that the tablespace is managed with uniform extents of a specific size (UNIFORM). The following statement creates a locally managed tablespace named lmtbsb and specifies AUTOALLOCATE:

CREATE TABLESPACE lmtbsb DATAFILE '/u02/oracle/data/lmtbsb01.dbf' SIZE 50M
EXTENT MANAGEMENT LOCAL AUTOALLOCATE;

AUTOALLOCATE causes the tablespace to be system managed with a minimum extent size of 64K. In contrast, dictionary-managed tablespaces have a minimum extent size of two database blocks. Therefore, in systems with block size smaller than 32K, autoallocated locally managed tablespace will be larger initially.

The alternative to AUTOALLOCATE is UNIFORM which specifies that the tablespace is managed with extents of uniform size. You can specify that size in the SIZE clause of UNIFORM. If you omit SIZE, then the default size is 1M.

The following example creates a tablespace with uniform 128K extents. (In a database with 2K blocks, each extent would be equivalent to 64 database blocks). Each 128K extent is represented by a bit in the extent bitmap for this file.

CREATE TABLESPACE lmtbsb DATAFILE '/u02/oracle/data/lmtbsb01.dbf' SIZE 50M
EXTENT MANAGEMENT LOCAL UNIFORM SIZE 128K;

You cannot specify the DEFAULT storage clause, MINIMUM EXTENT, or TEMPORARY when you explicitly specify EXTENT MANAGEMENT LOCAL. If you want to create a temporary locally managed tablespace, use the CREATE TEMPORARY TABLESPACE statement.

Automatic vs. Manual Segment Space Management

When you create a locally managed tablespace using the CREATE TABLESPACE statement, the SEGMENT SPACE MANAGEMENT clause lets you specify how free and used space within a segment is to be managed. You can choose either manual or automatic segment-space management.

- MANUAL: Manual segment-space management uses free lists to manage free space within segments. Free lists are lists of data blocks that have space available for

inserting rows. With this form of segment-space management, you must specify and tune the `PCTUSED`, `FREELISTS`, and `FREELIST GROUPS` storage parameters for schema objects created in the tablespace. `MANUAL` is the default.

- `AUTO`: Automatic segment-space management uses bitmaps to manage the free space within segments. The bitmap describes the status of each data block within a segment with respect to the amount of space in the block available for inserting rows. As more or less space becomes available in a data block, its new state is reflected in the bitmap. These bitmaps allow the database to manage free space automatically.

Automatic segment-space management delivers better space utilization than manual segment-space management. The following statement creates tablespace `lmtbsb` with automatic segment-space management:

```
CREATE TABLESPACE lmtbsb
DATAFILE '/u02/oracle/data/lmtbsb01.dbf' SIZE 50M
EXTENT MANAGEMENT LOCAL
SEGMENT SPACE MANAGEMENT AUTO;
```

Creating Tablespaces

Oracle strongly recommends the use of locally managed tablespaces and will eventually stop supporting dictionary-managed tablespaces. In Oracle Database 10*g*, locally managed tablespaces are the default for new permanent tablespaces.

Data Files and Tablespaces

A tablespace can have one or more data files, and a data file can belong to only one tablespace. Oracle creates a data file for a tablespace by specifying the keyword DATAFILE during tablespace creation. As a segment grows in size, Oracle allocates extents to it from the free space in its data files. When the tablespace starts to fill up, you can either add new data files to it or extend the size of the existing data files by using the RESIZE commandFor a temporary tablespace, you must use the clause TEMPFILE instead.

```
SQL> CREATE TABLESPACE test01
2 DATAFILE '/pasx02/oradata/pasx/test01.dbf'
3* SIZE 500M;
Tablespace created.
```

Now, let's execute the following query to determine the Oracle Database 10*g* Release 2 defaults for extent management, extent allocation type, and segment space management:

```
SQL> SELECT extent_management,
2 allocation_type,
3 segment_space_management
4 FROM dba_tablespaces
5* WHERE tablespace_name='TEST01';
EXTENT_MAN              ALLOCATIO SEGMEN
---------- ---------    -------
```

LOCAL SYSTEM AUTO

Note the defaults in Oracle Database 10*g* Release 2 carefully:

• Extent management: LOCAL

• Allocation of extent sizes: AUTOALLOCATE (shows up as SYSTEM in the preceding output)

• Segment space management: AUTO

Extent Allocation and Deallocation

Each Oracle data block corresponds to a specific number of bytes of disk space. Each of your database tables and indexes is called a *segment*, which is a set of extents allocated for a specific data structure Once Oracle allocates space to a segment by allocating a certain number of extents to it, that space will remain with the extent unless you make an effort to deallocate it. If you truncate a table with the DROP STORAGE option (TRUNCATE TABLE *table_name* DROP STORAGE), for example, Oracle deallocates the allocated extents. You can also manually deallocate unused extents using the following command:

SQL> ALTER TABLE *table_name* DEALLOCATE UNUSED;

When Oracle frees extents, it automatically modifies the bitmap in the data file where the extents are located, to indicate that they are free and available again.

26.(a) Explain in detail about Indexes and Materialized Views in Oracle

Oracle Indexes

Oracle indexes provide speedy access to table rows by storing sorted values of specified columns. Indexes enable you to find a row with a certain column value without your having to look at more than a small fraction of the total rows in the table. An index is a tree structure that allows direct access to a row in a table. Indexes can be classified based on their logical design or physical implementation. For example, some classifications are:

- *Single column and composite indexes*. Composite index is built on multiple columns in a table whereas single column index has only one column . The maximum number of columns in a composite key is 32.
- *Unique and non-unique key*. A unique index guarantees no two rows of a table have duplicate values.
- *Function-based indexes*. Function-based indexes can be created as either a B-tree or a bitmap index. B-tree is the default index created by Oracle and bitmap indexes can be used when a table has millions of rows and there are few distinct values for the column to be indexed. For example, bitmap indexes may be more preferable for the gender and martial status column of a passport table.

Oracle Index Schemes

Oracle Database provides several indexing schemes that provide complementary performance functionality. These are:

- B-tree indexes: the default and the most common
- B-tree cluster indexes: defined specifically for cluster
- Hash cluster indexes: defined specifically for a hash cluster
- Global and local indexes: relate to partitioned tables and indexes
- Reverse key indexes: most useful for Oracle Real Application Clusters applications
- Bitmap indexes: compact; work best for columns with a small set of values
- Function-based indexes: contain the precomputed value of a function/expression
- Domain indexes: specific to an application or cartridge.

Creating Index

The syntax of creating a B-tree index is

CREATE [UNIQUE] INDEX <index name> ON <table name> (<column name>, [column name]...) ...

The syntax of creating a bitmap index is

CREATE BITMAP INDEX <index name> ON <table name> (<column name>, [column name]...) ...

There are other optional clauses after the ON clause which specify the detailed storage characteristics of an index. They are omitted here for simplicity.

Examples:

SQL> CREATE INDEX info ON employee (empno);

Bitmap Indexes

To create a bitmap index, you use the CREATE INDEX statement with the BITMAP keyword added to it:

SQL> CREATE BITMAP INDEX gender_idx ON employee(gender)
TABLESPACE emp_index_05;

Reverse-Key Indexes

The biggest advantage to using reverse-key indexes is that they tend to avoid hot spots when you do sequential insertion of values into the index. Here's how to create one:

SQL> CREATE INDEX reverse_idx ON employee(emp_id) REVERSE;

Function-Based Indexes

Function-based indexes precompute functions on a given column and store the results in an index. When WHERE clauses include functions, function-based indexes are an ideal way to index the column. Here's how to create a function-based index, using the LOWER function:

SQL> CREATE INDEX lastname_idx ON employee(LOWER(l_name));

Partitioned Indexes

Partitioned indexes are used to index partitioned tables. Oracle provides two types of indexes for partitioned tables: *local* and *global*

Global Indexes

Global indexes on a partitioned table can be either partitioned or nonpartitioned. The globally nonpartitioned indexes are similar to the regular Oracle indexes for nonpartitioned tables. You just use the regular CREATE INDEX syntax to create these globally nonpartitioned indexes. Here's an example of a global index on the ticket_sales table:

```
SQL> CREATE INDEX ticketsales_idx ON ticket_sales(month)
GLOBAL PARTITION BY range(month)
(PARTITION ticketsales1_idx VALUES LESS THAN (3)
PARTITION ticketsales1_idx VALUES LESS THAN (6)
PARTITION ticketsales2_idx VALUES LESS THAN (9)
PARTITION ticketsales3_idx VALUES LESS THAN (MAXVALUE);
```

Local Indexes

Locally partitioned indexes, unlike globally partitioned indexes, have a one-to-one correspondence with the table partitions. You can create locally partitioned indexes to match partitions or even subpartitions. Here is a simple example of creating a locally partitioned index on a partitioned table:

```
SQL> CREATE INDEX ticket_no_idx ON
ticket_sales(ticket__no) LOCAL
TABLESPACE localidx_01;
```

Monitoring Index Usage

Oracle Database provides a means of monitoring indexes to determine whether they are being used. If an index is not being used, then it can be dropped, eliminating unnecessary statement overhead. To start monitoring the usage of an index, issue this statement:

ALTER INDEX *index* MONITORING USAGE; Later, issue the following statement to stop the monitoring:

```
ALTER INDEX index NOMONITORING USAGE;
```

Rebuilding an Existing Index

You can use the REBUILD command on a periodic basis to reorganize indexes to make them more compact and thus more efficient. You can also use the REBUILD command to alter the storage parameters you set during the initial creation of the index. Here's an example:

```
SQL> ALTER INDEX sales_idx REBUILD;
Index altered
Sql>
```

Using Materialized Views

A materialized view is a database object that contains the results of a query. The `FROM` clause of the query can name tables, views, and other materialized views. Collectively these objects are called **master tables** (a replication term) or **detail tables** (a data warehousing term). This reference uses "master tables" for consistency. The databases containing the master tables are called the **master databases**. You can do the following with a materialized view:

• Create indexes on a materialized view
• Create a materialized view on partitioned tables
• Partition a materialized view

You can use various types of aggregations like SUM, COUNT(*), AVG, MIN, and MAX in a materialized view. You can also use multiple table joins in the materialized view definition.

Query Rewriting

The materialized views are completely transparent to users. If users write qu ries using the underlying table, Oracle will automatically rewrite those queries to use the materialized views—this query-optimization technique is known as *query rewrite.*

The automatic query rewrite optimization technique is at the heart of materialized view usage. The QUERY_REWRITE_ENABLED initialization parameter determines whether Oracle will rewrite a query or not. When you enable query rewriting by setting QUERY_REWRITE_ENABLED = TRUE in your initialization parameter file, query rewriting is enabled system-wide, for the entire database.

The REWRITE_OR_ERROR Hint

If the queries take too long to complete without the materialized view, you can force Oracle to stop executing the query without the materialized view. You can use a hint to tell Oracle to issue an error instead of executing the unrewritten query. The hint is called the REWRITE_OR_ERROR hint, and here's how you use it:

```
SQL> SELECT /*+ REWRITE_OR_ERROR */
prod_id SUM(quantity_sold) AS sum_sales_qty FROM sales_data
GROUP BY prod_id
```

Refreshing Materialized View Data

The following sections present the materialized view refresh options.

Refresh Mode You can choose between the ON COMMIT and ON DEMAND modes of data refresh.

• ON COMMIT: In this mode, whenever a data change in one of the master tables is committed, the materialized view is refreshed automatically to reflect the change.

• ON DEMAND: In this mode, you must execute a procedure like DBMS_MVIEW.REFRESH to update the materialized view.

Refresh Type

You can choose from the following four refresh types:

• COMPLETE: This refresh option will completely recalculate the query underlying the materialized view.
FAST: Under the fast refresh mechanism, Oracle will use a *materialized view log* to log all changes to the master tables.
FORCE: If you choose this option, Oracle will try to use the FAST refresh mechanism.
NEVER: This refresh option never refreshes a materialized view.

Creating Materialized Views
There are three steps required to get the materialized views going, although the creation itself is simple:
1. Grant the necessary privileges.
2. Create the materialized view log (assuming you're using the FAST refresh option).
3. Create the materialized view.
Granting the Necessary Privileges
You must first grant the necessary privileges to the user who is creating the materialized views. The main privileges are those that enable the user to create a materialized view. In addition, you must grant the QUERY REWRITE privilege to the user, either by using the GLOBAL QUERY REWRITE privilege or specific QUERY REWRITE privileges on each object that is not part of the user's schema.

Here are the GRANT statements that enable a user to create a materialized view in the user's schema:
SQL> GRANT CREATE MATERIALIZED VIEW TO salapati;
SQL> GRANT QUERY REWRITE TO salapati;

Creating the Materialized View Log
Here's how you create the materialized view log:
SQL> CREATE MATERIALIZED VIEW LOG ON products;
Materialized view log created.

SQL> CREATE MATERIALIZED VIEW LOG ON sales;
Materialized view log created.

Creating the Materialized View
Now you are ready to create your materialized view. The example, shown in Listing 4 uses the REFRESH COMPLETE clause, to specify the COMPLETE refresh option.

Listing 4. *Creating a Materialized View*
SQL> CREATE MATERIALIZED VIEW test_mv
2 BUILD IMMEDIATE
3 REFRESH FAST ON COMMIT
4 ENABLE QUERY REWRITE
5 AS

```
6 SELECT sh.products.prod_category,
7 SUM(sh.sales.quantity_sold),
8 COUNT(sh.sales.quantity_sold), count(*)
10 FROM sh.sales, sh.products
11 WHERE sh.products.prod_id = sh.sales.prod_id
12 AND sh.products.prod_category <= 'Women'
13 AND sh.products.prod_category >= 'Boys'
14 GROUP BY sh.products.prod_category;
Materialized view created.
```

(OR)

(b) Outline the transaction management concepts in Oracle.

Oracle Transaction Management

A *transaction* is a logical unit of work consisting of one or more SQL statements. A transaction may perform one operation or an entire series of operations on the database objects, either interactively or as part of a program.

Oracle Transactions

The effects of all the SQL statements in a transaction can be either all **committed** (applied to the database) or all **rolled back** (undone from the database).A transaction begins with the first executable SQL statement. A transaction ends when it is committed or rolled back, either explicitly with a `COMMIT` or `ROLLBACK` statement or implicitly when a DDL statement is issued. A transaction starts implicitly when the first executable SQL statement begins, and it continues as the following SQL statements are processed until one of the following events occurs:

• COMMIT: If a transaction encounters a COMMIT statement, all the changes to that point are made permanent in the database.

• ROLLBACK: If a transaction encounters a ROLLBACK statement, all changes made up to that point are cancelled.

• *DDL statement*: If a user issues a DDL statement, such as CREATE, DROP, RENAME, or ALTER, Oracle first commits any current DML statements that are part of the transaction, before executing and committing the results of the DDL statement. This is called an *implicit commit*, since the committing of the DML statements immediately preceding the DDL statements isn't explicitly done by the user.

• *Normal program conclusion*: If a program ends without errors, all changes are implicitly committed by the database.

• *Abnormal program failure*: If the program crashes or is terminated, all changes made by it are implicitly rolled back by the database.

COMMIT Statement

The COMMIT statement ends a transaction successfully. All changes made by all SQL statements since the transaction began are recorded permanently in the database.

You can commit a transaction by using either of the following statements, which make the changes permanent:

SQL> COMMIT;
SQL> COMMIT WORK;

When an Oracle transaction is committed, the following three things happen:
1. The transaction tables in the redo records are tagged with the unique system change number (SCN) of the committed transaction.
2. The log writer writes the redo log information for the transaction from the redo log buffer to the redo log files on disk, along with the transaction's SCN. This is the point at which a commit is considered complete in Oracle.
3. Any locks that Oracle holds are released, and Oracle marks the transaction as complete. The default behavior for the COMMIT statement, which is generally the only type you'll encounter, is to use the IMMEDIATE and WAIT options:
• IMMEDIATE vs. BATCH: With the IMMEDIATE option, the log writer writes the redo log records for the committing transaction immediately to disk. If you'd rather the log writer write the redo records by buffering them in memory until it's convenient to write them, you can use the alternative BATCH option.
• WAIT vs. NOWAIT: With the WAIT option, the COMMIT statement doesn't return as successful until the redo records are successfully written to the redo logs. If you'd rather have the COMMIT statement return without waiting for the writing of the redo records, you can use the NOWAIT option. You can modify this default behavior by using the COMMIT_WRITE initialization parameter at either the system or the session level. To specify the BATCH and NOWAIT options by default, you can use the COMMIT_WRITE initialization parameter in the following way:

COMMIT_WRITE = BATCH, NOWAIT

ROLLBACK Statement
The ROLLBACK statement undoes, or rolls back, the changes made by SQL statements within a transaction, so long as you didn't already commit the transaction. Once you issue the ROLLBACK statement, none of the changes made to the tables by SQL statements since the transaction began are recorded to the database permanently.

SQL> ROLLBACK;
You can also partially roll back the effects of a transaction by using *save points* in the transaction. Using a save point, you can roll back to the last SAVEPOINT command in the transaction, as follows:
SQL> ROLLBACK TO SAVEPOINT POINT A;

Transaction Properties
Transactions in RDBMSs must possess four important properties, symbolized by the ACID acronym, which stands for *atomicity*, *consistency*, *isolation*, and *durability* of transactions. Let's look at the transaction properties

- *Atomicity*: Either a transaction should be performed entirely or none of it should be performed.

- *Consistency*: The database is supposed to ensure that it's always in a consistent state.
- *Isolation*: Isolation means that although there's concurrent access to the database by multiple transactions, each transaction must appear to be executing in isolation. The isolation property of transactions ensures that a transaction is kept from viewing changes made by another transaction before the first transaction commits
- *Durability*: The last ACID property, durability, ensures that the database saves commit transactions permanently.

Transaction Concurrency Control

Transaction concurrency is achieved by managing various users' simultaneous transactions without permitting any interference among them

One solution to concurrency control is to lock the entire table for the duration of each operation, so one user's transactions do not impact another's. Oracle *does use* locking mechanisms to keep the data consistent, but the locking is done in the least restrictive fashion, with the goal of maintaining the maximum amount of concurrency.

Concurrency Problems

Some of the most important problems potentially encountered in concurrent transaction processing are dirty reads, phantom reads, lost updates, and nonrepeatable reads.

The Dirty-Read Problem

A *dirty read* occurs when a transaction reads data that has been updated by an ongoing transaction but has not been committed permanently to the database.

The Phantom-Read Problem

Phantom-read problems are caused by the appearance of new data in between two database operations in a transaction.

The Lost-Update Problem

The *lost-update* problem is caused by transactions trying to read data while it is being updated by other transactions.

The Nonrepeatable-Read (Fuzzy-Read) Problem

When a transaction finds that data it has read previously has been modified by some other transaction, you have a *nonrepeatable-read* (or fuzzy-read) problem.

Schedules and Serializability

If the database permits concurrent access, then you need to consider the cumulative effect of all the transactions on database consistency. To do this, the database uses a *schedule*, which is a sequence of operations from one or more transactions. If all the transactions executed serially, one after another, the schedule would also be *serial*. If the database can produce a schedule that is equivalent in its effect to a serial schedule, even though it may be derived from a set of concurrent transactions, it is called a *serializable schedule.*

Isolation Levels

Isolation of transactions keeps concurrently executing database transactions from viewing incomplete results of other transactions. The main isolation levels are the serializable, repeatable read, read-uncommitted, and read-committed isolation levels. Here's what the different levels of transaction isolation levels mean:

Serializable: Under the serializable level of isolation, the transaction will lock all the tables it is accessing, thereby preventing other transactions from updating any of the tables underneath it until it has completed its transaction by using a COMMIT or ROLLBACK command.

• *Repeatable read*: The repeatable-read isolation level guarantees read consistency—a transaction that reads the data twice from a table at two different points in time will find the same values each time. You avoid both the dirty-read problem and the nonrepeatable-read probe-

• *Read uncommitted*: The read-uncommitted level, which allows a transaction to read another transaction's intermediate values before it commits, will result in the occurrence of all the problems of concurrent usage.

• *Read committed*: Oracle's default isolation level is the read-committed level of isolation at the statement level. Oracle queries see only the data that was committed at the beginning of the query. Because the isolation level is at the statement level, each statement is allowed to see only the data that was committed before the commencement of that statement. The read-committed level of isolation guarantees that the row data won't change while you're accessing a particular row in an Oracle table.

Transaction- and Statement-Level Consistency

Oracle automatically provides *statement-level* read consistency by default. That is, all data that a query sees comes from a single point in time. This means that a query will see consistent data when it begins. The query sees only data committed before it starts, and no data committed during the course of the query is visible to it.

Changing the Default Isolation Level

You can change the isolation level from the default level of read-committed to a serializable isolation level using the following statement:

SQL> ALTER SESSION SET ISOLATION LEVEL SERIALIZABLE;

A serializable level of isolation is suited for databases where multiple consistent queries need to be issued during an update transaction. However, serialization isn't a simple choice, because it seriously reduces your concurrency.

Reg. No.....

[16CSP203]

KARPAGAM UNIVERSITY
Karpagam Academy of Higher Education
(Established Under Section 3 of UGC Act 1956)
COIMBATORE – 641 021
(For the candidates admitted from 2016 onwards)

M.Sc., DEGREE EXAMINATION, APRIL 2017
Second Semester

COMPUTER SCIENCE

ORACLE 10G ADMINISTRATION

Time: 3 hours Maximum : 60 marks

PART – A (20 x 1 = 20 Marks) (30 Minutes)
(Question Nos. 1 to 20 Online Examinations)

(Part - B & C 2 ½ Hours)

PART B (5 x 6 = 30 Marks)
Answer ALL the Questions

21. a. Briefly explain the role of DBA and their different job classifications.
Or
b. Discuss the task of server parameter file.

22. a. Illustrate with examples on how to create and manage table spaces.
Or
b. Explain in detail about oracle locking methods with examples.

23. a. Enlighten the process of using external tables to load data.
Or
b. Discuss about data pump types, modes and its parameters.

24. a. Show with examples, how to manage users in an oracle database.
Or
b. Elucidate the process of backing up oracle databases.

25. a. Explain the process of optimizing oracle query processing.
Or
b. How does an automatic performance tuning Vs dynamic performance views.

PART C (1 x 10 = 10 Marks)
CASE STUDY (Compulsory)

26. Write the coding for creating different users for a student table with the field as roll number, name, date of birth, course, department, semester, percentage and performing granting and revoking the privileges for the user.
