DATABASE ADMINISTRATION

A COMPLETE HANDOUT

(In accordance to the DBA Syllabus provided by Department of Information Technology, Tribhuvan University)

Prepared by:

Sushant Thapa

Himalaya College of Engineering

2070 Batch

Tribhuvan University
Institute of Science and Technology
B.Sc. CSIT Seventh Semester Detailed-syllabus

Course Title: **Database Administration**
Course no: CSC-406 Full Marks: 60+20+20
Credit hours: 3 Pass Marks: 24+8+8

Nature of course: Theory (3 Hrs.) + Lab (3 Hrs.)

Course Synopsis: DBA Roles, DB backup, restoration and recovery, Tuning of database

Goal: The course covers about: principles of DBA Roles, DB backup, restoration and recovery, Tuning of database and overall DB administration which could be useful for administrator in the future.

Course contents:

Unit	Course Content-breakdown	Lecture Hours
1	**Introduction:** . DBMS architecture and data independence . DBA roles and responsibilities . SQL *PLUS Overview . SQL *Plus Fundamentals . Producing more readable outputs . Accepting values at runtime . Using iSQL *Plus.	5 hrs.
2	**Log File Management:** . Introduction to Control and Redo Log Files . Managing the control files . Maintaining and monitoring redo log files . Multiplexing redo log files . Archiving log files	5 Hrs
3	**Managing users and security:** . Profiling and Managing users . managing user privileges and roles . managing and querying role information . Database Security and Auditing . Creating and managing DB objects . Tables, indexes, triggers, views, stored procedures, etc. . Transaction concurrency management	10 Hrs
4	**DB Backup and Recovery:**	10 Hrs

	. Backup and Recovery Overview . Database backup, restoration and recovery . defining of backup and recovery procedure . Testing the backup and recovery plan . parallel instance recovery . recovering from non-critical loses	
5	**Oracle Recovery Manager (RMAN):** . Database corruption . automatic storage management . RMAN configuration . Database Archival	5 Hrs
6	**Unit 6: DB Performance Tuning:** . Introduction . Tuning methodology . Tuning concepts . AADM (Automatic Database Diagnostic Monitor) . SQL Tuning Advisor . AWR Report . Virtual Private Database . Policy types, selective columns, column masking	10 Hrs

Laboratory works:

1. Installation of Oracle Database
2. Database Creation
3. User Creation
4. Role, Privileges and group management
5. Database object creation
   - a. Tables
   - b. Indexes
   - c. Views
   - d. Triggers
   - e. Stored Procedures
   - f. Function
   - g. Package, etc.
6. Database Backup
   - a. Online backup
   - b. Offline backup
7. DB Recovery technique
   - a. Export and Import utility
   - b. Data Pump
   - c. Data guard
8. RMAN

9. Database Archiving
10. Performance Tuning
    a. ADDM Report
    b. AWR Report
    c. Spot Light
    d. OEM

Text Books:

1. Introduction to Database Administration, by O'reilly
2. ORACLE DBA handbooks
3. C.J. Date, Database Systems, Addison Wesley, 2000

Homework Assignment:

Assignment should be given throughout the semester and must be done individually.

Computer Usage: No Specific

Prerequisite: Database Management System

Category Content: Science Aspect: 40%
Design Aspect: 60%

Committee:

Prof. Dr. Subarna Shakya (Expert) -Tribhuvan University
Bishnu Nath Gautam -New Summit College
Ram Krishna Dahal -Kathford College
Laxman Adhikari -Amrit Science Campus

May 29-30-2012 (Jestha 16th and 17th 2069)
Central Department of Computer Science and Information Technology, TU

Unit 1:

Topics:

DBMS architecture and data independence

DBA roles and responsibilities

SQL *PLUS Overview

SQL *Plus Fundamentals

- Producing more readable outputs
- Accepting values at runtime

Using iSQL *Plus.

Overview:

Data:

Data are raw facts achieved through some observation or experiments and doesn't provide information until we process it.

Information:

Information are the outcomes of processed data that provides some knowledge about something of real time that it relates to.

Database:

It is a collection of data related to particular subjects or purpose. A database organizes data in easily accessible manner. A user can retrieve information from database in effective and efficient manner.

Database Management System (DBMS):

DBMS is a collection of interrelated data and set of programs to access those data. Its primary goal is to provide a way to store and retrieve database information in both efficient and convenient way.

Data Administrator and Database Administrator:

Data Administrator	Database Administrator
DAs are concerned with the data needs and data flows throughout the entire organization.	DBAs are concerned with storage and management of the information in the database.
The DA is involved more in the requirements gathering, analysis, and design phases	DBA in the design, development, testing, and operational phases.
DA determines long-term goals and enforces standards, policies and procedures.	DBA executes plans to achieve goals. He enforces standards, policies and procedures developed by DA.
DA also determines data requirements and develops conceptual and logical design.	DBA develops logical and physical design.
Work of DA is DBMS independent.	Work of DBA is DBMS-dependent.

DBMS Architecture

Three Level Database Architecture:

In database, data are stored as bits, or numbers and strings. But it is difficult to work with data at this level. So it is necessary to view data at different levels of abstraction.

DBMS architecture can be classified as three level schema architecture as:

- Internal/Physical level

- Conceptual level
- External level

Physical level

This level is the closest to the physical storage and is concerned with the way data is stored inside the system. It also provides a low level description of the physical database. This level defines the record types and methods of storage as well as how stored files are represented, what physical sequence the stored records are in, and what other physical structure exists.

Conceptual Level:

Conceptual level describes the structure of the whole database for a group of users. It is also called as the data model. Conceptual schema is a representation of the entire content of the database. These schema contains all the information to build relevant external records. It hides the internal details of physical storage.

External Level:

This is the highest level which is closest to the user. This level includes a number of user views or external schemas. External level is related to the data which is viewed by individual end users. External view describes the segment of the database that is required for a particular user group and hides the rest of the database from that user group.

DATA INDEPENDENCE

The ability to modify schema definition in one level without affecting schema definition in next higher level is called **data independence.**

There are two types of data independence:

- Logical Data Independence
- Physical Data Independence

Logical Data Independence:

It refers to the immunity of external models to the changes in the logical model. Changes to conceptual schema such as addition or removal of new entities, attributes or relationships should be possible without any changes to existing external schema.

It occurs at user interface level.

Physical Data Independence:

It refers to the immunity of conceptual schema to the changes in physical/internal schema. Changes to internal schema such as using different file organization or storage structure, using different storage devices, modifying indexes should be possible without having changes to conceptual or external schema.

It occurs at logical interface level.

Modifications at physical level are occasionally necessary to improve performance and so we can change physical level without affecting conceptual or external view of data.

Modifications at logical level are necessary whenever logical structure of database is altered. Example of Logical Data Independence: if we add/remove some new columns from able then user view and program should not change.

Logical independence is more difficult to achieve than physical independence, since application program are highly dependent on the logical structure of data that they access.

DBA ROLES AND RESPONSIBILITIES

Database Administrator is an Information Technology professional who creates actual data and put in place technical controls needed to enforce the various policies, decisions made by the data administrator.

Some of the roles and responsibilities of DBA are:

- Installing and upgrading the Oracle server and application tools
- Allocating system storage and planning future storage requirements for the database system
- Creating primary database storage structures (tablespaces) after application developers have designed an application
- Creating primary objects (tables, views, indexes) once application developers have designed an application
- Modifying the database structure, as necessary, from information given by application developers
- Enrolling users and maintaining system security
- Ensuring compliance with your Oracle license agreement
- Controlling and monitoring user access to the database
- Monitoring and optimizing the performance of the database
- Planning for backup and recovery of database information
- Maintaining archived data on tape
- Backing up and restoring the database
- Contacting Oracle Corporation for technical support

Tasks of a Database Administrator

The following tasks present a prioritized approach for designing, implementing, and maintaining an Oracle Database:

Task 1: Evaluate the Database Server Hardware

Task 2: Install the Oracle Software

Task 3: Plan the Database

Task 4: Create and Open the Database

Task 5: Back Up the Database

Task 6: Enroll System Users

Task 7: Implement the Database Design

Task 8: Back Up the Fully Functional Database

Task 9: Tune Database Performance

SQL *PLUS Overview

SQL*Plus is an interactive and batch query tool that is installed with every Oracle Database Server or Client installation. It has a command-line user interface, a Windows Graphical User Interface (GUI) and the *i*SQL*Plus web-based user interface.

SQL*Plus has its own commands and environment, and it provides access to the Oracle Database. It enables you to enter and execute SQL, PL/SQL, SQL*Plus and operating system commands to perform the following:

- Format, perform calculations on, store, and print from query results
- Examine table and object definitions
- Develop and run batch scripts
- Perform database administration

You can use SQL*Plus to generate reports interactively, to generate reports as batch processes, and to output the results to text file, to screen, or to HTML file for browsing on the Internet. You can generate reports dynamically using the HTML output facility of SQL*Plus, or using the dynamic reporting capability of *i*SQL*Plus to run a script from a web page.

SQL*Plus Command-line and Windows GUI Architecture

SQL*Plus command-line and the Windows GUI use a two-tier model comprising:

- Client (command-line user interface).
- Database (Oracle Database).

The two tiers may or may not be on the same machine.

SQL*Plus Client

The command-line user interface is the character based terminal implementation. The Windows GUI is an alternate user interface available in Windows installations.

Oracle Database

Oracle Database Net components provide communication between the SQL*Plus Client and Oracle Database.

PRODUCING MORE READABLE OUTPUT

Often, results returned from SQL *Plus wrap to the next line or do not have the proper formatting. You can use simple SQL *Plus formatting commands to produce more readable output and better looking reports.

Setting Page and Line sizes:

Use SHOW command to find the values of the PAGESIZE and LINESIZE environment variables.

SQL> SHOW PAGESIZE LINESIZE

Now, adjust these setting as follows:

SQL> SET PAGESIZE 55 LINESIZE 54

Formatting Columns:

Use COLUMN command to format the column headings and display column data. To display a different heading for the EMP_NAME column, use syntax:

SQL>COLUMN oldname HEADING "newname"

Use FORMAT to change the column display width.

SQL> COLUMN emp_name HEADING "Employee Name" FORMAT A20

To display salary in money format:

SQL> COLUMN sal FORMAT "$9,999.99"

To wrap text : SQL> COLUMN comments HEADING "Comments" WORD_WRAPPED

For justification: SQL> JUSTIFY RIGHT FORMAT A30

Suppressing Duplicate Values:

You can suppress display of duplicate column values using BREAK ON column_name command. The BREAK command has options to skip lines, pages and so on along with NODUPLICATE option.

Eg: if you want to suppress the display of duplicate JOB_ID values use:

SQL> BREAK ON job_id SKIP 2 NODUPLICATES

Single-Ampersand Substitution Variable

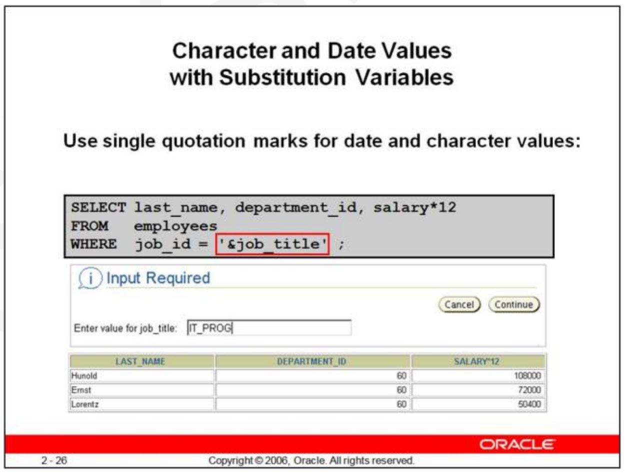
When running a report, users often want to restrict the data that is returned dynamically. iSQL*Plus provides this flexibility with user variables. Use an ampersand (&) to identify each variable in your SQL statement. You do not need to define the value of each variable.
The example in the slide creates an iSQL*Plus substitution variable for an employee number. When the statement is executed, iSQL*Plus prompts the user for an employee number and then displays the employee number, last name, salary, and department number for that employee.

With the single ampersand, the user is prompted every time the command is executed, if the variable does not exist.

Double-Ampersand Substitution Variable

You can use the double-ampersand (&&) substitution variable if you want to reuse the variable value without prompting the user each time. The user sees the prompt for the value only once. In the example in the slide, the user is asked to give the value for variable column_name only once. The value that is supplied by the user (department_id) is used for both display and ordering of data.

iSQL*Plus stores the value that is supplied by using the DEFINE command; it uses it again whenever you reference the variable name



Character and Date Values with Substitution Variables

Use single quotation marks for date and character values:

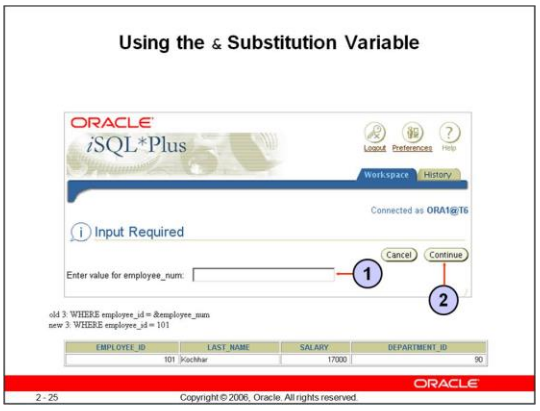
```
SELECT last_name, department_id, salary*12
FROM   employees
WHERE  job_id = '&job_title' ;
```

Input Required

Enter value for job_title: IT_PROG

LAST_NAME	DEPARTMENT_ID	SALARY*12
Hunold	60	108000
Ernst	60	72000
Lorentz	60	50400

2 - 26 Copyright © 2006, Oracle. All rights reserved.



Using the & Substitution Variable

Connected as ORA1@T6

Input Required

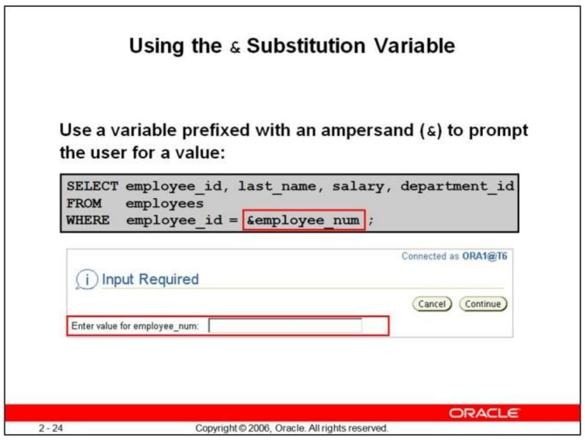
Enter value for employee_num:

```
old 3: WHERE employee_id = &employee_num
new 3: WHERE employee_id = 101
```

EMPLOYEE_ID	LAST_NAME	SALARY	DEPARTMENT_ID
101	Kochhar	17000	90

2 - 25 Copyright © 2006, Oracle. All rights reserved.

Sushant Thapa
2070 Batch
+977-98442-06228
Himalaya College of Engineering



Using the & Substitution Variable

Use a variable prefixed with an ampersand (&) to prompt the user for a value:

```
SELECT employee_id, last_name, salary, department_id
FROM   employees
WHERE  employee_id = &employee_num ;
```

Connected as ORA1@T6
Input Required
Cancel
Continue
Enter value for employee_num:
ORACLE
2 - 24
Copyright © 2006, Oracle. All rights reserved.

USING iSQL *PLUS

*i*SQL*Plus

*i*SQL*Plus is an environment in which you can do the following:Execute SQL statements to retrieve, modify, add, and remove data from the database

- Format, perform calculations on, store, and print query results in the form of reports
- Create script files to store SQL statements for repeated use in the future

To log in from a browser environment:

- Start the browser.
- Enter the URL address of the iSQL*Plus environment.
- On the Login page, enter appropriate values in the Username and Password

iSQL*Plus Environment

In the browser, the iSQL*Plus Workspace page has several key areas:

- **Text box**: Area where you type the SQL statements and iSQL*Plus commands
- **Execute button**: Click to execute the statements and commands in the text box
- **Load Script button**: Brings up a form where you can identify a path and file name or a URL that contains SQL, PL/SQL, or SQL*Plus commands and load them into the text box
- **Save Script button:** Saves the contents of the text box to a file
- **Cancel button**: Stops the execution of the command in the text box
- **Clear Screen button**: Click to clear text from the text box

- **Logout icon**: Click to end the iSQL*Plus session and return to the iSQL*Plus Login page
- **Preferences icon**: Click to change your interface configuration, system configuration, or password
- **Help icon**: Provides access to iSQL*Plus help documentation

Unit 2

Log File Management:

- Introduction to Control and Redo Log Files
- Managing the control files
- Maintaining and monitoring redo log files
- Multiplexing redo log files
- Archiving log files

Control Files

It is a small binary file that records the physical structure of the database. It includes *(can be written for application of Control files)*:

- Database Name
- Names and locations of associated datafiles and redo log files
- Timestamp of database creation
- Current log sequence number
- Checkpoint information

Control file must be available for writing by the Oracle Database server whenever the database is open. Without control file, the database cannot be mounted and recovery is difficult.

It is created at the same time as database. By default, at least one copy of control file is created but we should create two or more copies of control file during database creation and also if we lose a control file or want to change particular settings in the control files.

Managing Control Files:

Managing Size of Control files:

The main determinants of the size of a control file are the values set for the

- MAXDATAFILES,
- MAXLOGFILES,
- MAXLOGMEMBERS,
- MAXLOGHISTORY, and
- MAXINSTANCES

parameters in the CREATE DATABASE statement that created the associated database. Increasing the values of these parameters increases the size of a control file of the associated database.

Creating Control Files:

- **Creating initial control files**
- **Creating additional copies, renaming and relocating control files**
- **creating new control files**

Creating Initial control file

Initial control file is created when you issue the CREATE DATABASE statement. The names of control files are specified by CONTROL_FILES parameter in the initialization parameter file during database creation.

Example:

```
CONTROL_FILES = (/u01/oracle/prod/control01.ctl,
                                /u02/oracle/prod/control02.ctl,
/u03/oracle/prod/control03.ctl)
```

If file with same name already exist, you must specity CONTROL FILE REUSE clause in the CREATE DATABASE statement or else error will occur. If size of old control file differ from size parameter of new one, you cannot use REUSE clause.

Creating Additional Copies (Multiplexing), Renaming and Relocating control files:

Steps for multiplexing or renaming a control file:

1. Shut down the database
2. Copy existing control file from old location to new location using operating system commands
   eg: $ cp /u01/oracle/ica/control.ora   /u02/oracle/ica/control.ora
3. Edit the CONTROL_FILES parameter in database initialization parameter file
   - to add the new control file name, or renaming control filename, or specifying new location for multiplexing
   eg. for multiplexing: CONTROL_FILES= /u01/oracle/ica/control.ora, /u02/oracle/ica/control.ora
4. Restart the database

Creating new control file

You can create a new control file for database using CREAE CONTROLFILE

Sushant Thapa
2070 Batch
+977-98442-06228
Himalaya College of Engineering

```
CREATE CONTROLFILE
   SET DATABASE prod
   LOGFILE GROUP 1 ('/u01/oracle/prod/redo01_01.log',
                    '/u01/oracle/prod/redo01_02.log'),
           GROUP 2 ('/u01/oracle/prod/redo02_01.log',
                    '/u01/oracle/prod/redo02_02.log'),
           GROUP 3 ('/u01/oracle/prod/redo03_01.log',
                    '/u01/oracle/prod/redo03_02.log')
   RESETLOGS
   DATAFILE '/u01/oracle/prod/system01.dbf' SIZE 3M,
            '/u01/oracle/prod/rbs01.dbs' SIZE 5M,
            '/u01/oracle/prod/users01.dbs' SIZE 5M,
            '/u01/oracle/prod/temp01.dbs' SIZE 5M
   MAXLOGFILES 50
   MAXLOGMEMBERS 3
   MAXLOGHISTORY 400
   MAXDATAFILES 200
   MAXINSTANCES 6
   ARCHIVELOG;
```

Steps for creating new control files

1. Make a list of all datafiles and redo log files of the database.
   - SELECT MEMBER FROM V$LOGFILE;
   - SELECT NAME FROM V$DATAFILE;
   - SELECT VALUE FROM V$PARAMETER WHERE NAME = 'control_files';

2. Shut down the database. If the database is open, shut down the database normally if possible. Use the IMMEDIATE or ABORT clauses only as a last resort.

3. Back up all datafiles and redo log files of the database.

4. Start up a new instance, but do not mount or open the database:

   STARTUP NOMOUNT

5. Create a new control file for the database using the CREATE CONTROLFILE statement.
6. Store a backup of the new control file on an offline storage device.

Sushant Thapa
2070 Batch
+977-98442-06228
Himalaya College of Engineering

7. Edit the CONTROL_FILES initialization parameter for the database to indicate all of the control files now part of your database as
8. Recover the database if necessary. If you are not recovering the database, skip to
9. Open the database using one of the following methods:
   - ALTER DATABASE OPEN;
   -

Backing up Control Files:

Backing up control files is needed every time you change the physical structure of your database. Such structural changes include:

■ Adding, dropping, or renaming datafiles.

■ Adding or dropping a tablespace, or altering the read/write state of the tablespace.

■ Adding or dropping redo log files or groups.

Use the ALTER DATABASE BACKUP CONTROLFILE statement to back up your control files. You have two options:

■ Back up the control file to a binary file (duplicate of existing control file) using the following statement:

- ALTER DATABASE BACKUP CONTROLFILE TO '/oracle/backup/control.bkp';

■ Produce SQL statements that can later be used to re-create your control file:

- ALTER DATABASE BACKUP CONTROLFILE TO TRACE;

Recovering a control file using a current copy

- Recovering from Control File Corruption Using a Control File Copy.

- Recovering from Permanent Media Failure Using a Control File Copy.

Recovering from Control File Corruption Using a Control File copy

This procedure assumes that one of the control files specified in the CONTROL_FILES parameter is corrupted, the control file directory is still accessible, and that you have a multiplexed copy of the control file.

1. With the instance shut down, use an operating system command to overwrite the bad control file with a good copy:
   - % cp /u03/oracle/prod/control03.ctl /u02/oracle/prod/control02.ctl

2. Start SQL*Plus and open the database:
   - SQL> STARTUP

Recovering from Permanent Media Failure Using a Control File Copy.

This procedure assumes that one of the control files specified in the CONTROL_FILES parameter is inaccessible due to a permanent media failure and that you have a multiplexed copy of the control file.

1. With the instance shut down, use an operating system command to copy the current copy of the control file to a new, accessible location:

```
%cp /u01/oracle/prod/control01.ctl
/u04/oracle/prod/control03.ctl
```

2. Edit the CONTROL_FILES parameter in the initialization parameter file to replace the bad location with the new location:

- CONTROL_FILES = (/u01/oracle/prod/control01.ctl, /u02/oracle/prod/control02.ctl, /u04/oracle/prod/control03.ctl)

3. Start SQL*Plus and open the database:

```
SQL> STARTUP
```

Dropping Control Files

When we want to drop control files from the database, for example, if the location of a control file is no longer appropriate Remember that the database should have at least two control files at all times.

1. Shut down the database.
2. Edit the CONTROL_FILES parameter in the database initialization parameter file to delete the old control file name.
3. Restart the database.

Note: This operation does not physically delete the unwanted control file from the disk. Use operating system commands to delete the unnecessary file after you have dropped the control file from the database.

REDO LOG FILES

Redo log files are those files that logs history of all changes that are made to database. Each redo log file consists of redo records that holds a group of change vectors, each of which describes or represents a change made to single block in the database. Redo records are buffered in a circular fashion in the redo log buffer of the SGA and are written to one of the redo log files by the Log Writer (LGWR) database background process.

Whenever something changes in a data file, oracle records the changes in redo file. If database crashes, the RDBMS can redo all changes on data file which will take back the database to the state it was when the last redo record was written.

There are **two types** of redo log files:

1. Online Redo Logs
2. Archived Redo logs

Oracle Database uses only one redo log files at a time to store redo records written from the redo log buffer. The redo log file that LGWR is actively writing to is called the **current redo log file.**

Redo log files that are required for instance recovery are called **active redo log files.**

Redo log files that are no longer required for instance recovery are called **inactive redo** log files.

If you have enabled archiving (the database is in ARCHIVELOG mode), then the database cannot reuse or overwrite an active online log file until one of the archiver background processes (ARC*n) has archived its contents. If archiving is disabled (the* database is in NOARCHIVELOG mode), then when the last redo log file is full, LGWR continues by overwriting the first available active file.

LOG SWITCHES AND LOG SEQUENCE NUMBER

A log switch is the point at which the database stops writing to one redo log file and begins writing to another.

Normally, a log switch occurs when the current redo log file is completely filled and writing must continue to the next redo log file. However, you can configure log switches to occur at regular intervals, regardless of whether the current redo log file is completely filled. You can also force log switches manually.

Oracle Database assigns each redo log file a new **log sequence number** every time a log switch occurs and LGWR begins writing to it. When the database archives redo log files, the archived log retains its log sequence number. A redo log file that is cycled back for use is given the next available log sequence number.

Each online or archived redo log file is uniquely identified by its log sequence number. During crash, instance, or media recovery, the database properly applies redo log files in ascending order by using the log sequence number of the necessary archived and redo log files.

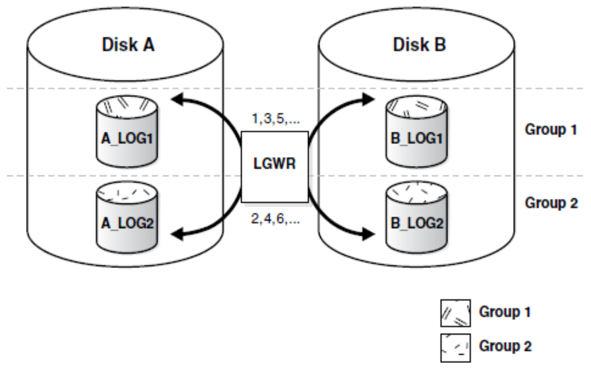
MAINTAINING AND MONITORING REDO LOG FILES

- Multiplexing Redo Log Files.
- Placing Redo Log Members on Different Disks.
- Setting the Size of Redo Log Members.
- Choosing the Number of Redo Log Files.
- Controlling Archive Lag.

Multiplexing Redo Log Files

Redo log files must be multiplexed in separate location (better in separate disks) in order to protect it against the failure involving the redo log itself. The redundancy of redo log file can help protect against I/O errors, file corruption and so on.
Multiplexing is implemented by creating group that consists of redo log file and its multiplexed copies where each group is defined by a number, such as group 1, group 2, etc.

Figure 10–2 Multiplexed Redo Log Files



Placing Redo log Members on Different Disks

When setting up a multiplexed redo log, place members of a group on different physical disks. If a single disk fails, then only one member of a

Sushant Thapa
2070 Batch
+977-98442-06228
Himalaya College of Engineering

group becomes unavailable to LGWR and other members remain accessible to LGWR, so the instance can continue to function.

Setting size of Redo log members

Size of Redo log file should be set in such a way that a filled group can be archived to a single unit of storage media with least amount of space on the medium left unused. All members of the same multiplexed redo log group must be of same size. The minimum size permitted for a redo log file is 4 MB.

Choosing number of Redo Log files

The MAXLOGFILES parameter used in CREATE DATABASE statement determines the maximum number of groups of redo log files for each database. The group value can range from 1 to MAXLOGFILES. If MAXLOGFILES is not specified for the CREATE DATABASE statement, then the database uses an operating system specific default value.

Controlling Archive Lag

In primary/standby database configuration, changes are made available to standby database by archiving redo logs at primary site and then shipping them to standby database. The changes that are being applied to standby database can lag the changes that are occurring on primary database as the standby database must wait for the changes in primary database redo log to be archived and then shipped to it.

To limit this lag, you can set ARCHIVE_LAG_TARGET initialization parameter and specify in seconds how long that lag can be.

eg: ARCHIVE_LAG_TARGET=1800

Creating redo log group

To create a new group of redo log files, use the SQL statement ALTER DATABASE with the ADD LOGFILE clause.

ALTER DATABASE ADD LOGFILE ('/oracle/dbs/log1c.rdo', '/oracle/dbs/log2c.rdo') SIZE 4M;

You can also specify the number that identifies the group using the GROUP clause:

ALTER DATABASE ADD LOGFILE GROUP 10 ('/oracle/dbs/log1c.rdo', '/oracle/dbs/log2c.rdo') SIZE 4M;

Creating Redo log members:

In some cases, it might not be necessary to create a complete group of redo log files. A group could already exist, but not be complete because one or more members of the group were dropped (for example, because of a disk failure). In this case, you can add new members to an existing group.

To create new redo log members for an existing group

ALTER DATABASE ADD LOGFILE MEMBER '/oracle/dbs/log2b.rdo' **TO GROUP 2;**

~~~~~~**REL OCATING AND RENAMING REDO LOG MEMBERS**

You can use operating system commands to relocate redo logs, then use the ALTER DATABASE statement to make their new names (locations) known to the database.
~~~~~~

To rename redo log members, you must have the ALTER DATABASE system privilege.

Before relocating your redo logs, or making any other structural changes to the database, completely back up the database in case you experience problems while performing the operation. As a precaution, after renaming or relocating a set of redo log files, immediately back up the database control file.

Steps:

1. Shut down the database.

```
SQL>SHUTDOWN
```

2. Copy the redo log files to the new location.

3. Startup the database, mount, but do not open it.

```
SQL> CONNECT / as SYSDBA

SQL>STARTUP MOUNT
```

4. Rename the redo log members.

Use the ALTER DATABASE statement with the RENAME FILE clause to rename the database redo log files.

```
ALTER DATABASE

RENAME FILE '/diska/logs/log1a.rdo', '/diska/logs/log2a.rdo'

TO '/diskc/logs/log1c.rdo', '/diskc/logs/log2c.rdo';
```

5. Open the database for normal operation. The redo log alterations take effect when the database is opened.

```
ALTER DATABASE OPEN;
```

DROPPING REDO LOG GROUP

To drop a redo log group, you must have the ALTER DATABASE system privilege.

Before dropping a redo log group, consider the following restrictions and precautions:

- An instance requires at least two groups of redo log files, regardless of the number of members in the groups. (A group comprises one or more members.)
- You can drop a redo log group only if it is inactive. If you need to drop the current group, first force a log switch to occur.
- Make sure a redo log group is archived (if archiving is enabled) before dropping it.

To see whether this has happened, use the V$LOG view.

Example:

- SELECT GROUP#, ARCHIVED, STATUS FROM V$LOG;

```
GROUP#   ARC  STATUS

--------- --- ----------------

1               YES   ACTIVE

2                NO   CURRENT

3               YES   INACTIVE

4                YES  INACTIVE
```

Drop a redo log group with the SQL statement ALTER DATABASE with the DROP LOGFILE clause.

e.g. ALTER DATABASE DROP LOGFILE GROUP 3;

DROPPING REDO LOG MEMBERS

To drop a redo log member, you must have the ALTER DATABASE system privilege.

You can drop a redo log member only if it is *not part of an active or current group.* If you want to drop a member of an active group, first force a log switch to occur.

To drop specific inactive redo log members, use the ALTER DATABASE statement with the DROP LOGFILE MEMBER clause.

The following statement drops the redo log /oracle/dbs/log3c.rdo:

```
ALTER DATABASE DROP LOGFILE MEMBER   '/oracle/dbs/log3c.rdo';
```

FORCING LOG SWITCHES

A log switch occurs when LGWR stops writing to one redo log group and starts writing to another. By default, a log switch occurs automatically when the current redo log file group fills.

You can force a log switch to make the currently active group inactive and available for redo log maintenance operations.

For example, you want to drop the currently active group, but are not able to do so until the group is inactive.

To force a log switch, you must have the ALTER SYSTEM privilege. Use the ALTER SYSTEM statement with the SWITCH LOGFILE clause.

The following statement forces a log switch:

```
ALTER SYSTEM SWITCH LOGFILE
```

MANAGING ARCHIVED REDO LOG FILES

Oracle Database lets you save filled groups of redo log files to one or more offline destinations, known collectively as the **archived redo log.**

The process of turning redo log files into archived redo log files is called **archiving. This process is only possible if** the database is running in **ARCHIVELOG mode.**

You can choose automatic or manual archiving.

Use of Archived Redo Log Files

■ Recover a database.

■ Update a standby database.

■ Get information about the history of a database using the LogMiner utility.

Running Database in NONARCHIVELOG mode

Running database in NOARCHIVELOG mode, disables archiving of the redo log. When a filled group becomes inactive after a log switch, the group is available for reuse by LGWR.

NOARCHIVELOG mode protects a database from instance failure but not from media failure.

In NOARCHIVELOG mode you cannot perform online tablespace backups, nor can you use online tablespace backups taken earlier while the database was in ARCHIVELOG mode.

To restore a database operating in NOARCHIVELOG mode, you can use only whole database backups taken while the database is closed. Therefore, if you decide to operate a database in NOARCHIVELOG mode, take whole database backups at regular, frequent intervals.

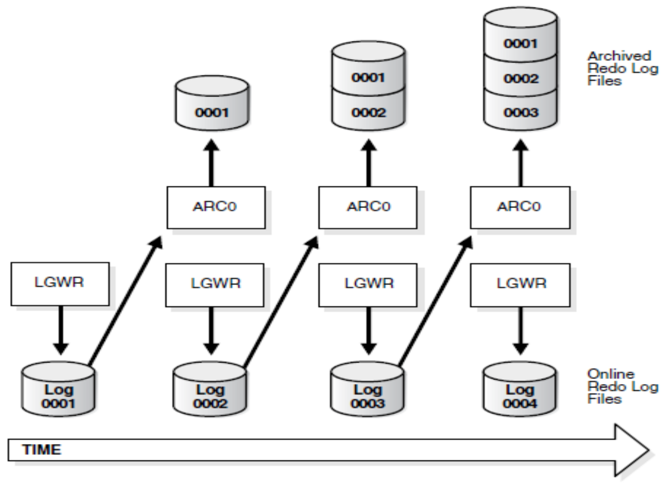
Running Database in ARCHIVELOG mode

Running database in ARCHIVELOG mode, enables the archiving of the redo log. A filled group becomes available for archiving immediately after a redo log switch occurs and can be reused by LGWR once they are finished archiving.

Advantages

- A database backup, together with online and archived redo log files, guarantees that you can recover all committed transactions in the event of an operating system or disk failure.
- If you keep an archived log, you can use a backup taken while the database is open and in normal system use.
- You can keep a standby database current with its original database by continuously applying the original archived redo logs to the standby.
- You can configure an instance to archive filled redo log files automatically, or you can archive manually. For convenience and efficiency, automatic archiving is usually best.

Figure 11–1 Redo Log File Use in ARCHIVELOG Mode



Steps for switching from NONARCHIVELOG mode to ARCHIVELOG mode

1. Shut down the database instance.

   SHUTDOWN

2. Back up the database:

   Before making any major change to a database, always back up the database to protect against any problems. This will be your final backup of the database in NOARCHIVELOG mode and can be used if something goes wrong during the change to ARCHIVELOG mode.

3. Edit the initialization parameter file to include the initialization parameters that specify the destinations for the archived redo log files.

4. Start a new instance and mount, but do not open, the database.

   – STARTUP MOUNT
   – To enable or disable archiving, the database must be mounted but not open.

Sushant Thapa
2070 Batch
+977-98442-06228
Himalaya College of Engineering

5. Change the database archiving mode. Then open the database for normal operations.

   – ALTER DATABASE ARCHIVELOG;

   – ALTER DATABASE OPEN;

6. Shut down the database.

   – SHUTDOWN IMMEDIATE

7. Back up the database:

Changing the database archiving mode updates the control file. After changing the database archiving mode, you must back up all of your database files and control file. Any previous backup is no longer usable because it was taken in NOARCHIVELOG mode.

Unit 3

Managing users and security:

- Profiling and Managing users
- managing user privileges and roles
- managing and querying role information
- Database Security and Auditing
- Creating and managing DB objects: Tables, indexes, triggers, views, stored procedures, etc

---

Managing Oracle Users:

Access to Oracle database is provided using database accounts known as usernames (users).

A user account is identified by a user name and defines the user's attributes, including the following:

- Password for database authentication

- Profile
- Default tablespace for database objects
- Default temporary tablespace for query processing work space

When you create a user, you are also implicitly creating a schema for that user. A **schema** is a logical container for the database objects (such as tables, views, triggers, and so on) that the user creates.

When you drop (delete) a user, you must either first drop all the user's schema objects, or use the cascade feature of the drop operation, which simultaneously drops a user and all of his schema objects.

Viewing Users

You can view database user by using a SQL command:

select * from all_users;

After viewing a list of users, you can then select an individual users to alter or drop (delete).

Creating Users

Before creating a user, determine the following:

- Whether or not you want to permit the user to create database objects in his own schema.

  If so, grant the RESOURCE role or grant individual create object system privileges

- Whether or not you want to grant the user DBA privileges.

  If so, grant the DBA role. Because DBA privileges include the ability to create database objects in any schema, if you grant the DBA role, you do not need to grant the RESOURCE role or individual create object system privileges.

Caution: Granting the DBA role to a user has security implications, because the user can modify objects in other users' schemas.

- Whether or not to create the user with an expired password.

  When you do this, the password that you assign the user is used only for the user's first login. Upon first login, the user is prompted to select a new password.

Example:

```
SQL> CREATE USER RAMAN
  IDENTIFIED BY RAMAN123
  PROFILE CSIT
  DEFAULT TABLESPACE USERS
  PASSWORD EXPIRE
  ACCOUNT UNLOCK;


User created.
```

Altering User:

Altering a user means changing some of his user attributes. You can change all user attributes except the user name, default tablespace, and temporary tablespace. If you want to change the user name, you must drop the user and re-create him with a different name

One of the attributes that you can alter is the user password. To do this you can either assign new password to the user, or request the new password from the user and then enter it. An easier and more secure way to cause a password change is to expire the password.

When you **expire** a password, the user is prompted to change his password the next time that he logs in.

Example:

```
SQL> ALTER USER RAMAN
 IDENTIFIED BY RAAMAAN;
User altered.
```

Locking and Unlocking User

To temporarily deny access to the database for a particular user, you can lock the user account. If the user then attempts to connect, the database displays an error message and disallows the connection. You can unlock the user account when you want to allow database access again for that user.

Note:

Many internal user accounts are locked (or both expired and locked). You should not attempt to log in with these locked user accounts.

The HR user account, which contains a sample schema, is initially expired and locked. You must log in as SYSTEM, unlock the account, and assign a password before you can log in as HR.

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

Dropping User:

Dropping a user removes the user from the database. Before you can drop a user, you must first drop all the user's schema objects. Or, you can use the cascade feature of the drop operation, which simultaneously drops a user and all his schema objects.

The following are two alternatives to dropping a user and losing all the user's schema objects:

1. To temporarily deny access to the database for a particular user while preserving the user's schema objects, you can lock the user account.
2. To drop a user but retain the data from the user's tables, export the tables first.

Caution
Under no circumstances should you attempt to drop the SYS or SYSTEM users, or any other internal user account. Doing so could cause Oracle Database to malfunction.

Profiles:

A profile is a collection of parameters that sets limits on database resources. When you change the profile to a user, you assign a profile but you apply also a set of parameters.

If you assign the profile to a user, then that user cannot exceed these limits parameters.

You can use profiles to configure database settings such as:

- ✓ sessions per user,
- ✓ logging and tracing features,
- ✓ controlling user passwords, and so on.

Creating a profile:

You can create many profiles in the database that specify both resource management parameters and password management parameters.

However, you can assign a user only one profile at any given time. To create a profile, use CREATE PROFILE command by assigning a name of profile and then specifying the parameter names and their values separated by space(s).

Profiles only take effect when resource limits are "turned on" for the database as a whole.

To see status of resource limit:

```
SQL> show parameter resource_limit
NAME                                 TYPE
VALUE
------------------------------------ ----------- ---
------
resource_limit                       boolean
FALSE
```

To enable resource limit, use the ALTER SYSTEM statement and set RESOURCE_LIMIT=TRUE.

```
SQL> ALTER SYSTEM SET RESOURCE_LIMIT = TRUE;
System altered.
SQL> show parameter resource_limit
NAME                                 TYPE
VALUE
------------------------------------ -----------
---------
resource_limit                       boolean
TRUE
```

Example:

```
SQL> CREATE PROFILE csit LIMIT
  SESSIONS_PER_USER 2
  IDLE_TIME 5
  CONNECT_TIME 10;
Profile created.
```

Assigning Profiles:

Profile can be assign in two ways either during USER creation or by using ALTER statement.

```
SQL> ALTER USER shraddha
 PROFILE csit;
User altered.
```

Altering Profiles:

To alter profile use ALTER PROFILE command. A DBA must have the ALTER PROFILE system privilege to use this command. You can change any parameter in profile by using this command. The changes takes effect next time the user connects to the database.

```
SQL> ALTER PROFILE csit LIMIT
  COMPOSITE_LIMIT 1500
  PASSWORD_LOCK_TIME 5;
Profile altered.
```

Dropping Profile:

Profiles that are no longer needed can be dropped using DROP PROFILE command. If the profile you want to delete is assigned to a user, trying to delete that profile will return error. For such profile use CASCADE command which will delete the profile and assigns default profile to the respective user.

```
SQL> DROP PROFILE CSIT;
```

```
*ERROR at line 1:ORA-02382: profile CSIT has users assigned, cannot drop without CASCADE
```

```
SQL> DROP PROFILE csit CASCADE;
```

```
Profile dropped.
```

Managing User Privileges and Roles:

In Oracle database, privileges control access to the data and restricts the action user can perform. With proper privileges, user can create, drop, or modify objects in their own schema or in another user's schema. Privileges also determines the data to which a user should have access.

There are two main types of user privileges:

- ✓ System privileges—A system privilege is the right to perform a particular action, or to perform an action on any schema objects of a particular type. For example, the privileges to create tables and to delete the rows of any table in a database are system privileges.
- ✓ Object privileges—An object privilege is a right to perform a particular action on a specific schema object. Different object privileges are available for different types of schema objects. The privilege to delete rows from the DEPARTMENTS table is an example of an object privilege.

You can grant privileges to a user by two methods:

1. assign directly to user
2. assign privilege to role and then assign role to the user.

Object Privilege:

Object privileges are granted on a specific object. The owner of the object has all privileges on the object. The owner can grant privileges on that object to any other users of the database or can also authorize another user in the database to grant privileges on the object to other users.

eg:

SQL> GRANT SELECT, UPDATE ON CUSTOMER TO SUSHANT;

Here SUSHANT can only query and update rows in the CUSTOMER table but cannot insert or delete, or grant privilege to another user in the database.

SQL> GRANT SELECT, UPDATE ON CUSTOMER

TO SUSHANT WITH GRANT OPTION;

Now, SUSHANT can grant the privilege SELECT and UPDATE on CUSTOMER to other users.

Some of the object privileges that can be granted to users are: SELECT, UPDATE, DELETE, INSERT, EXECUTE, ALTER, etc.

System Privilege:

System privileges are the privileges that enable the user to perform an action; that are not specified on any particular object. Like object privilege, system privilege can also be granted to a user, a role, or PUBLIC. When a privilege is specified with ANY, the user is authorized to perform the actions on any schema in the database.

The use of WITH ADMIN OPTION clause gives privilege to a user to the grant privilege to another user.

eg: GRANT SELECT ANY TABLE TO SUSHANT WITH ADMIN OPTION;

Some of the system privileges that can be granted to users are: SELECT ANY TABLE, ALTER DATABASE, CREATE SESSION, BACKUP ANY TABLE, etc.

Revoking Privileges:

User's object privileges and system privileges can be revoked by using REVOKE statement.

eg:

To revoke UPDATE privilege granted to SUSHANT on CUSTOMER:

SQL> REVOKE UPDATE ON CUSTOMER FROM SUSHANT;

To revoke multiple privileges, use comma to separate multiple privileges:

SQL> REVOKE CREATE SESSION, SELECT ANY TABLE FROM SUSHANT;

To revoke REFERENCES privilege, specify CASCADE CONSTRAINTS clause, which will drop the referential integrity constraints created using the privileges. You must use this clause if any constraints exists.

SQL> REVOKE REFERENCES ON CUSTOMER FROM SUSHANT CASCADE CONSTRAINTS;

To revoke all privileges:

SQL> REVOKE ALL ON CUSTOMER FROM SUSHANT;

Roles:

Role is a named set of related privileges that are granted to users or to other roles, which eases the management of privileges and hence improve security.

Role Characteristics

- Privileges are granted to and revoked from roles as if the role were a user.
- Roles can be granted to and revoked from users or other roles as if they were system privileges.
- A role can consist of both system and object privileges.
- A role can be enabled or disabled for each user who is granted the role.
- A role can require a password to be enabled.
- Roles are not owned by anyone; and they are not in any schema.

Benefits of Roles

- Easier privilege management.
- Dynamic privilege management.
- Selective availability of privileges.
- Can be granted through the operating system.

Easier Privilege Management

Use roles to simplify privilege management. Rather than granting the same set of privileges to several users, you can grant the privileges to a role, and then grant that role to each user.

Dynamic Privilege Management

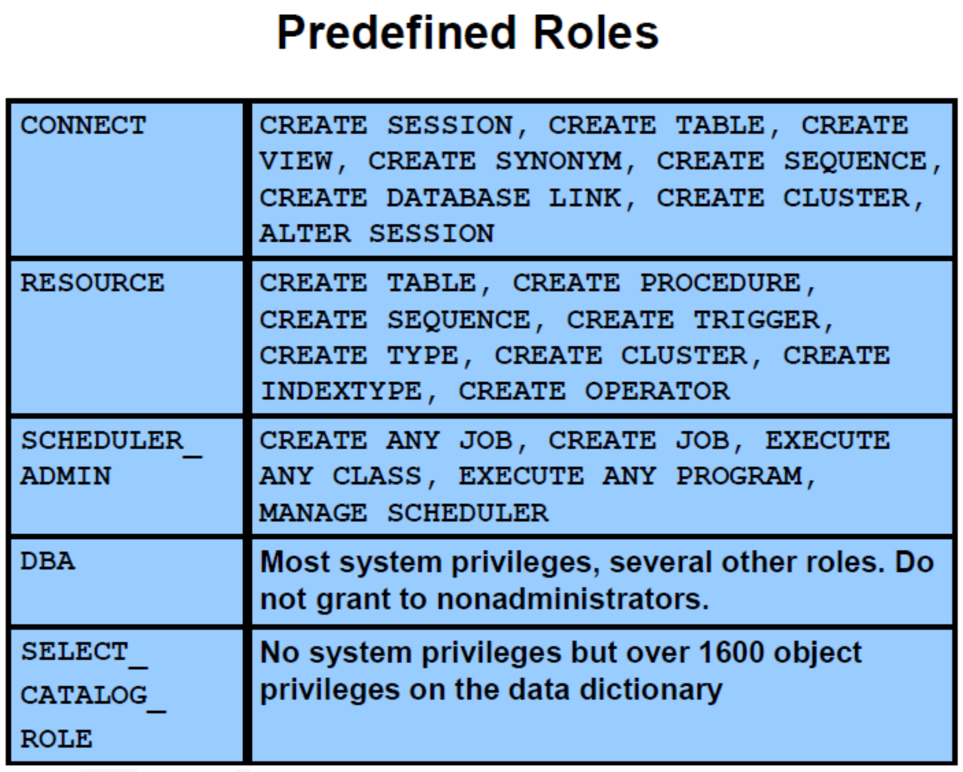
If the privileges associated with a role are modified, all the users who are granted the role acquire the modified privileges automatically and immediately.

Selective Availability of Privileges

Roles can be enabled and disabled to turn privileges on and off temporarily. Enabling a role can also be used to verify that a user has been granted that role.

Granting Through the Operating System

Operating system commands or utilities can be used to assign roles to users in the database.



Predefined Roles

CONNECT	CREATE SESSION, CREATE TABLE, CREATE VIEW, CREATE SYNONYM, CREATE SEQUENCE, CREATE DATABASE LINK, CREATE CLUSTER, ALTER SESSION
RESOURCE	CREATE TABLE, CREATE PROCEDURE, CREATE SEQUENCE, CREATE TRIGGER, CREATE TYPE, CREATE CLUSTER, CREATE INDEXTYPE, CREATE OPERATOR
SCHEDULER_ ADMIN	CREATE ANY JOB, CREATE JOB, EXECUTE ANY CLASS, EXECUTE ANY PROGRAM, MANAGE SCHEDULER
DBA	Most system privileges, several other roles. Do not grant to nonadministrators.
SELECT_ CATALOG_ ROLE	No system privileges but over 1600 object privileges on the data dictionary

Creating Role:

Using CREATE ROLE command creates the role. No user owns the role; it is owned by the database. When roles are created, no privileges are associated with it. Privileges will be granted to roles later.

eg: SQL> CREATE ROLE MANAGER;

SQL> GRANT SELECT ON CUSTOMER TO MANAGER;

SQL> GRANT INSERT, UPDATE ON CUSTOMER TO MANAGER;

Similar to user, role can also be authenticated. The default is NOT IDENTIFIED which means no authorization is required.

Two authorization methods are available:

1. Database:
   Database authorizes role by using a password associated with the role. Whenever such roles are enabled, user is prompted for a password. Eg.
   SQL> CREATE ROLE MANAGER IDENTIFIED BY MNGR123;

2. Operating System:
   Role is authorized by Operating System when the OS can associate its privileges with the application privileges, and information about each user is configured in operating system files. To enable OS role authorization, set the parameter OS_ROLES=TRUE;
   Eg. SQL> CREATE ROLE APPLICATION_USER IDENTIFIED EXTERNALLY;

Removing Roles:

Use DROP ROLES statement to remove role from database. Dropping a role will cause all privileges, that users had through the role, to get lost.

eg: SQL> DROP ROLE admin;

Quering Role Information:

- **The data dictionary view DBA_ROLES lists the roles defined in the database. The column PASSWORD specifies the authorization method.**
  SQL> SELECT * FROM DBA_ROLES;
  ```
  ROLE                                  PASSWORD
  ```

```
-----------------------------          --------
CONNECT                                NO
RESOURCE                               NO
DBA                                    NO
LOGSTDBY_ADMINISTRATOR                 NO
```

- **The view SESSION_ROLES lists the roles that are enabled in the current session**

  ```
  SQL> SELECT * FROM SESSION_ROLES;
  ```

- **The view DBA_ROLE_PRIVS (or USER_ROLE_PRIVS) lists all the roles granted to users and roles.**

  ```
  SQL> SELECT * FROM DBA_ROLE_PRIVS;
  ```

- **The view ROLE_ROLE_PRIVS lists the roles granted to the roles, ROLE_SYS_PRIVS lists the system privileges granted to role and ROLE_TAB_PRIVS shows information on the object privileges granted to role.**

  ```
  SQL> SELECT * FROM ROLE_ROLE_PRIVS WHERE ROLE='DBA';
  ROLE                      GRANTED_ROLE                   ADM
  ------------------------- ------------------------------ ---
  DBA                       SCHEDULER_ADMIN                YES
  DBA                       DELETE_CATALOG_ROLE            YES
  DBA                       EXP_FULL_DATABASE              NO

  SQL> SELECT * FROM ROLE_SYS_PRIVS WHERE ROLE='CONNECT';
  ROLE                      PRIVILEGE                                ADM
  ------------------------- ---------------------------------------- ---
  CONNECT                   CREATE SESSION                           NO

  SQL> SELECT * FROM ROLE_TAB_PRIVS WHERE
  TABLE_NAME='EMPLOYEE';
  ```

Database Security and Auditing

Database Security:

Database administrator should follow best practices and continuously monitor database activity for a secure system.

A secure system ensures the confidentiality of the data it contains. There are several aspects of security:

- Restricting access to data and services.
- Authenticating users.
- Monitoring for suspicious activity.

Restricting Access to Data and Services:

All users should not have access to all data. Depending on what is stored in your database, restricted access can be mandated by business requirements, customer expectations, and increasingly by legal restrictions. Credit card information, health care data, identity information, and more must be protected from unauthorized access. Oracle provides extremely fined-grained authorization controls to limit database access. Restricting access should include applying the principal of least privilege.

Authenticating User:

The most basic form of user authentication is by challenging the user to provide something they know such as a password. Ensuring that passwords by following simple rules can greatly increase the security of your system. Stronger authentication methods include requiring the user to provide something they have, such as a token or Public Key Infrastructure (PKI) certificate. An even stronger form of authentication is to identify the user through a unique biometric characteristic such as a fingerprint, iris scan, bone structure patterns, and so on. Oracle supports advanced authentication techniques such as token-, biometric-, and certificate-based identification through the Advanced Security Option. User accounts that are not in use should be locked to prevent attempts to compromise authentication.

Monitoring for Suspicious Activity:

Even authorized, authenticated users can sometimes compromise your system. Identifying unusual database activity such as an employee who suddenly begins querying large amounts of credit card information, research results, or other sensitive information, can be the first step to detecting information theft. Oracle provides a rich set of auditing tools to track user activity and identify suspicious trends.

Monitoring or auditing should be an integral part of your security procedures. Oracle's built-in audit tools include:

- Standard Database auditing
- Value-based auditing
- Fine-grained auditing (FGA)

Standard database auditing captures several pieces of information about an audited event including that the event occurred, when it occurred, the user who caused the audited event, and which client machine the user was on when the event happened.

Use value-based auditing to audit changes to data (inserts, updates, deletes). Value-based auditing extends standard database auditing, capturing not only that the audited event occurred, but the actual values that were inserted, updated, or deleted. Value-based auditing is implemented through database triggers.

Use fine-grained auditing (FGA) to audit SQL statements. FGA extends standard database auditing, capturing the actual SQL statement that was issued rather than only that the action occurred.

Auditing Database:

Auditing is storing information about database activity. You can use auditing to monitor suspicious database activity and to collect statistics on database usage. On creation of database, Oracle creates a SYS.AUD$ table, known as the audit trail, which stores the audited records.
To enable auditing, set the initialization parameter AUDIT_TRAIL to TRUE or DB. When this parameter is set to OS, Oracle writes the audited records to an OS file instead of inserting them to SYS.AUD$ table.
Use the AUDIT command to specify the audit actions.

Oracle has three types of auditing capabilities:

- **Statement auditing:** Audits SQL statements. (AUDIT SELECT BY SUSHANT audits all SELECT statements performed by SUSHANT)
- **Privilege auditing:** Audits privileges (AUDIT CREATE TRIGGER audits all user who exercises their CREATE TRIGGER privileges)
- **Object auditing:** Audits the use of specific object. (AUDIT SELECT ON HR.EMPLOYEES monitors the SELECT statement performed on the EMPLOYEES table)

BY clause: restricts the auditing scope by specifying the user list in it.

WHENEVER SUCCESSFUL clause: to specify that only successful statements are to be audited.

WHENEVER NOT SUCCESSFUL clause: to limit auditing to failed statements.

BY SESSION clause: specifies that one audit record is inserted for one session, regardless of number of times statement is executed.

BY ACCESS clause: specifies that one audit record is inserted each time the statement is executed.

Example:

To audit connections and disconnections to the database: AUDIT SESSION;

To audit only successful logins: AUDIT SESSION WHENEVER SUCCESSFUL;

To audit only failed logins: AUDIT SESSION WHENEVER NOT SUCCESSFUL;

To audit successful logins for specific users: AUDIT SESSION BY SUSHANT, RAM WHENEVER SUCCESSFUL;

To audit successful updates and deletes on the EMPLOYEES table: AUDIT UPDATE, DELETE ON HR.EMPLOYEES BY ACCESS WHENEVER SUCCESSFUL;

To **turn off** all auditing, use NOAUDIT command. You can use all commands available in AUDIT statement except BY SESSION and BY ACCESS.

NO AUDIT UPDATE, DELETE ON HR.EMPLOYEES;

Creating and managing DB objects (Tables, indexes, triggers, views, stored procedures, etc.**)**

Creating Tables:

Use CREATE TABLE command to create table. You can create a table under the username used to connect to the database or with proper privileges; you can create a table under another username.

```
CREATE TABLE STUDENT(

SSN NUMBER,

NAME VARCHAR2(20),

GRADE VARCHAR(10),

AGE NUMBER);
```

Some of the data types used for creating table are: CHAR(<size>[BYTE | CHAR]), VARCHAR(<size>[BYTE | CHAR]), NUMBER(<precision>,<scale>), DATE, etc.

Altering Tables:

Use ALTER TABLE command to change the table's storage settings, add or drop columns, or modify the columns characteristics such as default value, datatypes, and length.

Dropping table:

If a table is no longer used, you can drop it to free up space. Once dropped, it cannot be undone. The syntax is :

DROP TABLE [schema.] table_name [CASCADE CONSTRAINTS]

Truncating a table:

It is similar to DROP, but it doesn't remove the structure of table, so none of the indexes, constraints, triggers, and privileges on the table are dropped. You cannot roll back a truncate operation. The syntax is:

TRUNCATE {TABLE | CLUSTER} [<schema>.] table_name [{DROP | REUSE} STORAGE].

You cannot truncate the parent table of an enabled referential integrity constraint. You must first disable constraint and then truncate the table even if the child table has no rows.

Querying table information:

DBA_TABLES/USER_TABLES/ALL_TABLES: use this view to query information about tables (TABS is a synonym for USER_TABLES)

DBA_TAB_COLUMNS: use this view to display information about the columns in a table.

Indexes:

Indexes are used to access data more quickly than reading the whole table, and they reduce disk I/O considerably when the queries use the available indexes. You can create any number of indexes on a table. A column can be a part of many indexes and you can specify as many as 30 columns in an index. When you specify more than one column, the index is known as a composite index.

Following types of indexes can be created:

Bitmap: A bitmap index doesn't repeatedly store the index column values. Each value is treated as a key, and a bit is set for the corresponding ROWIDs. Bitmap indexes are suitable for columns with low cardinality, such as the SEX column in an EMPLOYEE table, in which possible values are M and F.

B-tree: This type of index is the default. You can create the index by using the b-tree algorithm. The b-tree includes nodes with the index column values and the ROWID of the row. The ROWIDs identify the rows in the table.

Types of b-tree indexes:

- **Non-unique**
- **Unique**
- **Reverse Key**

- **Function-based**

Creating Index:

The CREATE INDEX statement creates a non-unique b-tree index on the column specified. You must specify a name for the index and the table name on which the index should be built.

Eg: **to create index on ORDER_DATE column of the ORDERS table:**

CREATE INDEX IND1_ORDERS ON ORDERS (ORDER_DATE);

To create a unique index: CREATE UNIQUE INDEX IND2_ORDERS ON ORDERS (ORDER_DATE);

To create a bitmap index: CREATE BITMAP INDEX IND3_ORDERS ON ORDERS (STATUS);

To create a reverse key index on ORDER_NUM and ORDER_DATE columns:

CREATE UNIQUE INDEX IND4_ORDERS ON ORDERS (ORDER_DATE, ORDER_NUM)

TABLESPACE USER_INDEX REVERSE;

To create a function-based index on SUBSTR(PRODUCT_ID,1,2):

CREATE INDEX IND5_ORDERS ON ORDERS (SUBSTR(PRODUCT_ID,1,2))

TABLESPACE USER_INDEX;

Altering Indexes:

Using the ALTER INDEX command, you can make following changes on an Index:

- Change it's STORAGE clause, except for the parameter INITIAL and MINEXTENTS
- Deallocate unused blocks
- Rebuild the index
- Coalesce leaf nodes
- Manually allocate extents
- Change the PARALLEL/NOPARALLEL, LOGGING/NOLOGGING clauses
- Modify partition storage parameters, rename partitions, drop partitions and so on.
- Rename the index
- Specify the ENABLE/DISABLE clause to enable or disable function-based indexes

Eg:

To allocate an extent:

ALTER INDEX STUDENT.IND1_ORDERS ALLOCATE EXTENT SIZE 200K;

Dropping Indexes:

Use DROP INDEX command to drop index. Oracle frees up all the space used by the index when it is dropped.

DROP INDEX STUDENT.IND4_ORDERS;

Managing Constraints:

Constraints are created in the database to enforce a business rule and to specify relationships between various tables. You can also enforce business rules by using database triggers and application code.

There are five types of integrity constraints that prevents bad data from entering database:

NOT NULL: Prevents NULL values from being entered into the column.

CHECK: Checks whether the condition specified in the constraints is satisfied.

UNIQUE: Ensures that there are no duplicate values for the column(s) specified. Every value or set of values is unique within the table.

PRIMARY KEY: Uniquely identifies each row of the table. Prevents NULL values. A table can have only one PRIMARY KEY constraint.

FOREIGN KEY: Establishes a parent-child relationship between tables by using common columns.

Creating Constraints:

Use CREATE TABLE or ALTER TABLE statements to create constraints. You can specify the constraint definition at the column level if the constraint is defined on a single column. You define multiple column constraints at the table level; specifying the columns in parenthesis separated by comma. To provide name for constraint use keyword CONSTRAINT followed by constraint_name.

NOT NULL:

Characteristics:

- Is defined at column level
- Use CREATE TABLE to define constraint when creating a table

```
CREATE TABLE STAFF(
SSN NUMBER NOT NULL,
NAME VARCHAR(19),
JOIN_DATE DATE
CONSTRAINT SS_JOIN_DATE NOT NULL
);
```

- Use ALTER TABLE MODIFY to add or remove a NOT NULL constraint on the column of existing table.

```
ALTER TABLE STAFF MODIFY JOIN_DATE NULL;
```

CHECK:

Characteristics:

- Can be defined at column level or table level.
- The condition specified in the CHECK clause should evaluate to a Boolean result and can refer to values in other columns of the same row; the condition cannot use queries.
- One column can have more than one CHECK constraint defined. The column can have a NULL value.
- CREATE TABLE BONUS(
  EMP_ID VARCHAR2(20) NOT NULL,
  SALARY NUMBER(9,2),
  BONUS NUMBER(9,2),
  CONSTRAINT CK_BONUS CHECK(BONUS > 0));

Unique:

Characteristics:

- Can be defined at column level for single-column unique keys. For multiple-column unique key(maximum number of columns can be 32), the constraint should be defined at table level.
- Unique constraints allow NULL values in the constraint columns.
- ALTER TABLE BONUS
  ADD CONSTRAINT UQ_EMP_ID UNIQUE (DEPT, EMP_ID);

Primary key:

Characteristics:

- All characteristics of UNIQUE key are applicable except that NULL values are not allowed in the primary key columns.
- A table can have only one PRIMARY KEY.

- Oracle creates a unique index and NOT NULL constraints for each column in the key.
- CREATE TABLE EMPLOYEE(
  DEPT_NO VARCHAR2(2),
  EMP_ID NUMBER(4),
  NAME VARCHAR2(20),
  CONSTRAINT PK_EMPLOYEE PRIMARY KEY (DEPT_NO, EMP_ID));

Foreign Key:

The foreign key is the column or columns in the table (child table) in which the constraint is created; the referenced key is the primary key column(s) in the table (parent table) that is referenced by the constraint.

Characteristics:

- Can be defined at the column level or table level. Define multiple-column foreign keys at the table level.
- The foreign key column(s) and referenced key column(s) can be in the same table
- NULL values are allowed in the foreign key columns.
- The ON DELETE clause specifies the action to be taken when a row in the parent table is deleted and child rows exist with the deleted parent primary key. You can delete the child rows (CASCADE) or set the foreign key column values to NULL.
  ALTER TABLE CITY ADD CONSTRAINT FK_STATE
  FOREIGN KEY (COUNTRY_CODE, STATE_CODE)
  REFERENCES STATE (COUNTRY_CODE, STATE_CODE)

ON DELETE CASCADE;

Creating Disabled Constraints:

Newly created constraint is automatically enabled. To create disabled constraint use DISABLE keyword after constraint definition.

Dropping Constraints:

To drop constraint, use ALTER TABLE and specify the constraint name.

ALTER TABLE BONUS DROP CONSTRAINT CK_BONUS2;

To drop unique key constraints with referenced foreign keys, specify the CASCADE clause to drop the foreign key constraints and the unique constraint. Specify the unique key columns.

ALTER TABLE EMPLOYEE DROP UNIQUE (EMP_ID) CASCADE;

Views:

A view is a SQL statement that is stored in the database with an associated name. A view is actually a composition of a table in the form of a predefined SQL query.

A view can contain all rows of a table or select rows from a table. A view can be created from one or many tables which depends on the written SQL query to create a view.

Views, which are a type of virtual tables allow users to do the following −

Structure data in a way that users or classes of users find natural or intuitive.

- Restrict access to the data in such a way that a user can see and (sometimes) modify exactly what they need and no more.

- Summarize data from various tables which can be used to generate reports.

Creating Views:

Database views are created using the CREATE VIEW statement. Views can be created from a single table, multiple tables or another view.

To create a view, a user must have the appropriate system privilege according to the specific implementation.

The basic CREATE VIEW syntax is as follows −

CREATE VIEW view_name AS

SELECT column1, column2.....

FROM table_name

WHERE [condition];

You can include multiple tables in your SELECT statement in a similar way as you use them in a normal SQL SELECT query.

The WITH CHECK OPTION

The WITH CHECK OPTION is a CREATE VIEW statement option whose purpose is to ensure that all UPDATE and INSERTs satisfy the condition(s) in the view definition.

If they do not satisfy the condition(s), the UPDATE or INSERT returns an error.

The following code block has an example of creating same view CUSTOMERS_VIEW with the WITH CHECK OPTION.

```
CREATE VIEW CUSTOMERS_VIEW AS
```

```
SELECT name, age
FROM  CUSTOMERS
WHERE age IS NOT NULL
WITH CHECK OPTION;
```

The WITH CHECK OPTION in this case should deny the entry of any NULL values in the view's AGE column, because the view is defined by data that does not have a NULL value in the AGE column.

Updating a View

A view can be updated under certain conditions which are given below:

- The SELECT clause may not contain the keyword DISTINCT.
- The SELECT clause may not contain summary functions.
- The SELECT clause may not contain set functions.
- The SELECT clause may not contain set operators.
- The SELECT clause may not contain an ORDER BY clause.
- The FROM clause may not contain multiple tables.
- The WHERE clause may not contain subqueries.
- The query may not contain GROUP BY or HAVING.
- Calculated columns may not be updated.
- All NOT NULL columns from the base table must be included in the view in order for the INSERT query to function.

So, if a view satisfies all the above-mentioned rules then you can update that view. The following code block has an example to update the age of Ramesh.

```
SQL > UPDATE CUSTOMERS_VIEW
   SET AGE = 35
   WHERE name = 'Ramesh';
```

Sushant Thapa
2070 Batch
+977-98442-06228
Himalaya College of Engineering

This would ultimately update the base table CUSTOMERS and the same would reflect in the view itself. Now, try to query the base table and the SELECT statement would produce the following result.

```
+----+----------+-----+-----------+----------+
| ID | NAME     | AGE | ADDRESS   | SALARY   |
+----+----------+-----+-----------+----------+
|  1 | Ramesh   |  35 | Ahmedabad |  2000.00 |
|  2 | Khilan   |  25 | Delhi     |  1500.00 |
|  3 | kaushik  |  23 | Kota      |  2000.00 |
|  4 | Chaitali |  25 | Mumbai    |  6500.00 |
|  5 | Hardik   |  27 | Bhopal    |  8500.00 |
|  6 | Komal    |  22 | MP        |  4500.00 |
|  7 | Muffy    |  24 | Indore    | 10000.00 |
+----+----------+-----+-----------+----------+
```

Inserting Rows into a View

Rows of data can be inserted into a view. The same rules that apply to the UPDATE command also apply to the INSERT command.

Here, we cannot insert rows in the CUSTOMERS_VIEW because we have not included all the NOT NULL columns in this view, otherwise you can insert rows in a view in a similar way as you insert them in a table.

Deleting Rows into a View

Rows of data can be deleted from a view. The same rules that apply to the UPDATE and INSERT commands apply to the DELETE command.

Following is an example to delete a record having AGE = 22.

```
SQL > DELETE FROM CUSTOMERS_VIEW
```

```
   WHERE age = 22;
```

This would ultimately delete a row from the base table CUSTOMERS and the same would reflect in the view itself. Now, try to query the base table and the SELECT statement would produce the following result.

```
+----+----------+-----+-----------+----------+
| ID | NAME     | AGE | ADDRESS   | SALARY   |
+----+----------+-----+-----------+----------+
|  1 | Ramesh   |  35 | Ahmedabad |  2000.00 |
|  2 | Khilan   |  25 | Delhi     |  1500.00 |
|  3 | kaushik  |  23 | Kota      |  2000.00 |
|  4 | Chaitali |  25 | Mumbai    |  6500.00 |
|  5 | Hardik   |  27 | Bhopal    |  8500.00 |
|  7 | Muffy    |  24 | Indore    | 10000.00 |
+----+----------+-----+-----------+----------+
```

Dropping Views

Obviously, where you have a view, you need a way to drop the view if it is no longer needed. The syntax is very simple and is given below –

```
DROP VIEW view_name;
```

Following is an example to drop the CUSTOMERS_VIEW from the CUSTOMERS table.

```
DROP VIEW CUSTOMERS_VIEW;
```

Importance of View:

- Views are created for the security purposes for the database. It is always helpful for us to restrict the people from the outside to see the information like columns and data in the columns for the sake of the security of the data access.
- We can use view for abstraction of the data in the table which is not known by the end user.

- The data accessible through view is not stored anywhere in the database as a distinct object.
- A view can join two or more tables and show it as one object to user.

Triggers:

A database trigger is special stored procedure that is run when specific actions occur within a database. Most triggers are defined to run when changes are made to a table's data. Triggers can be defined to run *instead of* or *after* DML (Data Manipulation Language) actions such as INSERT, UPDATE, and DELETE.

Triggers help the database designer ensure certain actions are completed regardless of which program or user makes changes to the data.

The programs are called triggers since an event, such as adding a record to a table, fires their execution.

Events

The triggers can occur AFTER or INSTEAD OF a DML action. Triggers are associated with the database DML actions INSERT, UPDATE, and DELETE. Triggers are defined to run when these actions are executed on a specific table.

AFTER triggers

Once the DML actions, such as an INSERT completes, the AFTER trigger executes. Here are some key characteristics of AFTER triggers:

- After triggers are run after a DML action, such as an INSERT statement and any ensuing referential cascade actions and constraint checks have run.
- You can't cancel the database action using an AFTER trigger. This is because the action has already completed.

- One or more AFTER triggers per action can be defined on a table, but to keep things simple I recommend only defining one.
- You can’t define AFTER triggers on views.

INSTEAD OF triggers

INSTEAD OF triggers, as their name implies, run in place of the DML action which caused them to fire. Items to consider when using INSTEAD OF triggers include:

- An INSTEAD OF trigger overrides the triggering action. If an INSTEAD OF trigger is defined to execute on an INSERT statement, then once the INSERT statement attempt to run, control is immediately passed to the INSTEAD OF trigger.
- At most, one INSTEAD OF trigger can be defined per action for a table.
- Triggers use two special database objects, INSERTED and DELETED, to access rows affected by the database actions. Within the scope of a trigger the INSERTED and DELETE objects have the same columns as the trigger’s table.
- The INSERTED table contains all the new values; whereas, the DELETED table contains old values. Here is how the tables are used:
- INSERT – Use the INSERTED table to determine which rows were added to the table.
- DELETE – Use the DELETED table to see which rows were removed from the table.
- UPDATE – Use the INSERTED table to inspect the new or updated values and the DELETED table to see the values prior to update.

Creating a trigger:

The syntax for creating a trigger is:

```
CREATE OR REPLACE TRIGGER trigger_name
{BEFORE | AFTER |INSTEAD OF}
{INSERT [OR]|UPDATE[OR]|DELETE}
[OF col_name]
ON table_name
```

```
[REFERENCING OLD AS o NEW AS n]
[FOR EACH ROW]
WHEN(condition)
DECLARE
declaration_statements
BEGIN
executable_statements
EXCEPTION
exception-handling-statements
END;
```

Where,

- CREATE [OR REPLACE] TRIGGER trigger_name: Creates or replaces an existing
- trigger with the *trigger_name*.
- {BEFORE | AFTER | INSTEAD OF}: This specifies when the trigger will be executed.
- The INSTEAD OF clause is used for creating trigger on a view.
- {INSERT [OR] | UPDATE [OR] | DELETE}: This specifies the DML operation.
- [OF col_name]: This specifies the column name that will be updated.
- [ON table_name]: This specifies the name of the table associated with the trigger.
- [REFERENCING OLD AS o NEW AS n]: This allows you to refer new and old values for various DML statements, such as INSERT, UPDATE, and DELETE.
- [FOR EACH ROW]: This specifies a row-level trigger, i.e., the trigger will be executed for each row being affected. Otherwise the trigger will execute just once when the SQL statement is executed, which is called a table level trigger.
  - WHEN (condition): This provides a condition for rows for which the trigger would fire. This clause is valid only for row-level triggers.

EXAMPLE:

```
Select * from customers;

+----+----------+-----+-----------+----------+
| ID | NAME     | AGE | ADDRESS   | SALARY   |
+----+----------+-----+-----------+----------+
|  1 | Ramesh   |  32 | Ahmedabad |  2000.00 |
|  2 | Khilan   |  25 | Delhi     |  1500.00 |
|  3 | kaushik  |  23 | Kota      |  2000.00 |
|  4 | Chaitali |  25 | Mumbai    |  6500.00 |
|  5 | Hardik   |  27 | Bhopal    |  8500.00 |
|  6 | Komal    |  22 | MP        |  4500.00 |
+----+----------+-----+-----------+----------+
```

This trigger will display the salary difference between the old values and new values:

```
CREATE OR REPLACE TRIGGER display_salary_changes
    BEFORE DELETE OR INSERT OR UPDATE ON customers
    FOR EACH ROW
    WHEN (NEW.ID > 0)
    DECLARE
    sal_diff number;
BEGIN
    sal_diff := :NEW.salary - :OLD.salary;
    dbms_output.put_line('Old salary: ' ||
:OLD.salary);
    dbms_output.put_line('New salary: ' ||
:NEW.salary);
    dbms_output.put_line('Salary difference: ' ||
sal_diff);
END;    /
```

Triggering a Trigger

Let us perform some DML operations on the CUSTOMERS table. Here is one INSERT statement, which will create a new record in the table:

INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)

```
VALUES (7, 'Kriti', 22, 'HP', 7500.00 );
```

When a record is created in the CUSTOMERS table, the above create trigger, display_salary_changes will be fired and it will display the following result:

```
Old salary:
New salary: 7500
Salary difference:
```

Because this is a new record, old salary is not available and the above result comes as
null. Let us now perform one more DML operation on the CUSTOMERS table. The UPDATE
statement will update an existing record in the table:

```
UPDATE customers
SET salary = salary + 500
WHERE id = 2;
```

When a record is updated in the CUSTOMERS table, the above create trigger, display_salary_changes will be fired and it will display the following result:

```
Old salary: 1500
New salary: 2000
Salary difference: 500
```

Benefits of Triggers

Triggers can be written for the following purposes:

- Generating some derived column values automatically.
- Enforcing referential integrity.
- Event logging and storing information on table access.
- Auditing.
- Synchronous replication of tables.
- Imposing security authorizations.

- Preventing invalid transactions.

Unit 4

DB Backup and Recovery:

- Backup and Recovery Overview
- Database backup, restoration and recovery
- defining of backup and recovery procedure
- Testing the backup and recovery plan
- parallel instance recovery
- recovering from non-critical loses

Database must be protected from failures to protect crucial data from losing. There are two major categories of database failures:

- non-media failures: less critical in nature (statement failures, process failures, instance failures, and user errors), and
- media (disk) failures: more critical in nature—the inability to read or write from a database file.

Non Media Failure

- In most cases, statement, process, and instance failures are automatically handled by Oracle and require no DBA intervention. User error can require a manual recovery performed by the DBA.
- *Statement failure consists of a syntax error in the statement, and Oracle usually returns an* error number and description.
- *Process failure occurs when the user program fails for some reason, such as when there is an* abnormal disconnection or a termination. The process monitor (PMON) process usually handles cleaning up the terminated process.

- *Instance failure occurs when the database instance abnormally terminates due to a power* spike or outage. Oracle handles this automatically upon start-up by reading through the current online redo logs and applying the necessary changes back to the database.
- *User error occurs when a table is erroneously dropped or data is erroneously removed.*

Media or Disk Failure

A media failure occurs when the database fails to read or write from a file that it requires.

For example, a disk drive could fail, a controller supporting a disk drive could fail, or a database file could be removed, overwritten, or corrupted. Each type of media failure that occurs requires a different method for recovery.

The basic step to perform media recovery are:

1. Determine which files will need to be recovered: data files, control files, and/or redo logs.
2. Determine which type of media recovery is required: complete or incomplete, opened database, or closed database.
3. Restore backups of the required files: data files, control files, and offline redo logs (archived logs) necessary to recover.
4. Apply offline redo logs (archived logs) to the data files.
5. Open the database at the desired point, depending on whether you are performing a complete or an incomplete recovery.
6. Perform frequent testing of the process. Create a test plan of typical failure scenarios.

Backup and Recovery Overview

Backup and **recovery** is one of the most important aspect of database administration.

A backup is a representative copy of data. This copy can include important parts of a database such as the control file, redo logs and datafiles. A backup protects data from application error and acts as safeguard against unexpected data loss, by providing a way to restore original data.

Backups are divided into:

- Physical backups, and
- logical backups

Physical backups are copies of physical database files. The phrase "backup and recovery" usually refers to the transfer of copied files from one location to another, along with the various operations performed on these files.

In contrast, logical backups are those that contains data that is exported using SQL commands and stored in a binary file. Logical backups are used to supplement Oracle server. To recover a restored backup, data is updated using redo records from the transaction log.

Different type of backup strategy may include:

- **The entire database (whole).**
  A whole database backup includes all data files and at least one control file (remember that all control files within a database are identical).
- **A portion of the database (partial).**
  Partial database backups may include zero or more tablespaces, zero or more data files, and may or may not include a control file.

Backup type may be:

- **All information from all data files (full)**
  Full backups make a copy of every data block within the files being backed up that contains data.
- **Only information that has changed since some previous backup (incremental)**
  Incremental backups make a copy of all data blocks that have changed since some previous backup.

Backup mode may be:

- **Offline (consistent, cold)**
  Offline backups (also known as consistent backups) are taken while the database is not open. They are consistent because at the time of the backup, the SCN data file headers matches the SCN in the control files.
- **Online (inconsistent, hot)**
  Online backups (also known as hot or inconsistent backups) are taken while the database is open. The backups are inconsistent because with the database open there is no guarantee that the data files are synchronized with the control files. Inconsistent backups require recovery in order to be used.

Backups may be stored as:

- **Image copies:**
  Image copies are duplicates of data or archived log files (similar to simply copying the files using operating system commands).
- **Backup sets:**
  Backup sets are copies of one or more data or archived log files. With backup sets, empty data blocks are not stored, thereby causing backup sets to use less space on disk or tape. Backup sets can be compressed to further reduce the space requirements of the backup.

The advantage of creating a backup as an image copy is improved granularity of the restore operation. With an image copy only the file or files need to be retrieved from tape.

With backup sets the entire backup set must be retrieved from tape before you extract the file or files that are needed.

The advantage of creating backups as backup sets is better space usage. Most databases contain 20% or more empty blocks. Image copies back every single data block up, even if the data block is empty. Backup sets significantly reduce the space required by the backup.

Recovery

A major responsibility of the database administrator is to prepare for the possibility of hardware, software, network, process, or system failure. If such a failure affects the operation of a database system, you must usually recover the database and return to normal operation as quickly as possible. Recovery should protect the database and associated users from unnecessary problems and avoid or reduce the possibility of having to duplicate work manually.

Recovery has two part: rolling forward and rolling back. When Oracle rolls forward, it applies redo records to the corresponding data blocks. Oracle systematically goes through the redo log, which records every changes before the failure of system has occurred, to determine which changes it needs to apply to which blocks and then changes the block. Once Oracle has completed rolling forward stage, it begins rolling back where Oracle search through the rollback information stored in transaction table for uncommitted transactions, undoing any that it finds.

The Physical data Structures used in recovering data are:

- Datafiles and data blocks

- Control Files
- Rollback Segments
- Redo Log Files

Datafiles and Data Blocks:

A data file is a file which is part of an Oracle database that stores data - including user data and undo data and are collectively known as tablespaces.

Every Oracle database has one or more physical datafiles. The datafile is divided into smaller units called data blocks. Data blocks are the smallest units of storage that the database can use or allocate.

The first block of every datafile is header that contains important information such as file size, block size, tablespace, and creation timestamp. Whenever the database is opened, Oracle checks to see that the datafile header information matches the information stored in the control file. If it doesn't match then recovery is necessary.

Oracle reads data in a datafile during normal operation and stores it in the buffer cache. The Database Writer (DBWR) or DB Writer later writes from buffer cache to disk. The more data that accumulates in memory without being written in disk, the longer the recovery time it will take.

Control files:

The control file is the file that contains the record of the physical structures of the database and their status. Several types of information stored in the control file related to backup and recovery are:

- Database information (RESETLOGS SCN and time stamp)
- Tablespace and datafile records (filenames, datafile checkpoints, read/write status, offline ranges)
- Information about redo threads (current online redo log)

- Log records (log sequence numbers, SCN range in each log)
- A record of past RMAN backups
- Information about corrupt datafile blocks

Every time a user mounts database, its control file is used to identify the datafiles and online redo log files that must be opened for database operation. If physical structure of database changes, a new datafile or redo log file is created. Oracle then modifies the database's control file to reflect the change.

The control file should be backed up whenever the structure of database changes. Loss of the control file makes recovery from a data loss much more difficult.

Rollback Segments (Undo Segments):

Every database contains one or more rollback segments which is the logical structure contained in datafile that records the initial state of information before a transaction modifies a data block.

In general, the rollback segment of a database store the old values of data changed by uncommitted transactions which Oracle can use during database recovery to undo any uncommitted changes applied from the redo log to the datafiles, putting the data into a consistent state.

Redo Log Files:

An Oracle database requires at least two online redo log groups, and in each group there is at least one online redo log member, an individual redo log file where the changes are recorded.

Redo logs record all changes made to a database's data files. Each time data is changed in the database, that change is recorded in the online redo log first, before it is applied to the datafiles.

At intervals, the database rotates through the online redo log groups, storing changes in the current online redo log.

Because the redo log contains a record of all changes to the datafiles, if a backup copy of a datafile from some point in time and a complete set of redo logs from that time forward are available, the database can reapply changes recorded in the redo logs, in order to re-construct the datafile contents at any point between the backup time and the end of the last redo log. However, this is only possible if the redo log has been preserved.

Therefore, preserving the redo logs is a major part of most backup strategies. The first level of preserving the redo log is through a process called archiving. The database can copy online redo log groups that are not currently in use to one or more archive locations on disk, where they are collectively called the archived redo log. Individual files are referred to as archived redo log files. After a redo log file is archived, it can be backed up to other locations on disk or on tape, for long term storage and use in future recovery operations.

There are three basic types of recovery:

- instance recovery
- crash recovery
- media recovery

First two are performed by Oracle automatically at instance startup. Latter requires the user to issue command.

Instance recovery:

It occurs in an open database when one instance discovers that another instance has crashed. The surviving instance automatically uses the redo log to recover the committed data in database buffers that was lost when the instance failed.

Crash recovery:

It occurs when either a single-instance database crashes or all instance of multi-instance database crash. In crash recovery, an instance must first open the database and then execute recovery operation.

Media recovery:

It is executed on user's command, usually in response to media failure. In media failure, online or archived redo logs can be used to make a restored backup current or to update it to a specific point in time. Media recovery can restore the whole database, a tablespace, or a datafile and recover them to some non-current time, media recovery is being performed.

Database Backup, restoration and recovery:

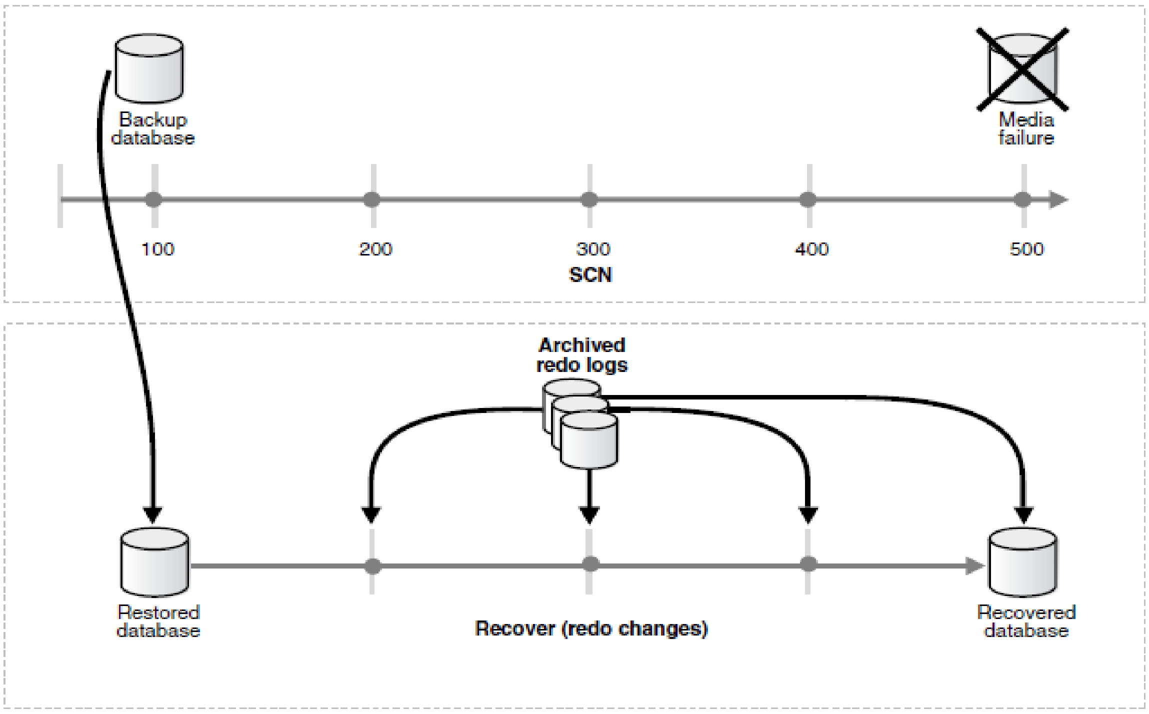
For normal functionality of a database, Oracle consists of set of physical structures that includes datafiles, redo logs, control files, and initialization files. If these files are not present, the database may not start up or it may halt during normal operations. So, all of these files must be backed up on a regular basis to disk, tape, or both. Such a backup can consist of a user-managed backup or Recovery Manager (RMAN)-based backup.

Both of these backups can be used to restore the necessary database files from disk or tape to the desired location.

The restore process consists of copying the database files from tape or disk to a desired location so that database recovery can begin.

The recovery process consists of starting the database and making it consistent using a complete or partial backup copy of some of the physical structures of the database. Recovery has many options depending on the type of backups that are performed.

Figure 1–1 Restoring and Recovering a Database



In this example a full backup of a database (copies of its datafiles and control file) is taken at SCN 100. Redo logs generated during the operation of the database capture all changes that occur between SCN 100 and SCN 500.

Along the way, some logs fill and are archived. At SCN 500, the datafiles of the database are lost due to a media failure.

The database is then returned to its transaction-consistent state at SCN 500, by restoring the datafiles from the backup taken at SCN 100, then applying the transactions captured in the archived and online redo logs and undoing the uncomitted transactions.

Defining a Backup and Recovery Strategy

To create a solid *backup and recovery strategy, you must keep in mind six major* requirements:

1. The amount of data that can be lost in the event of a database failure.
2. The length of time that the business can go without the database in the event of a database failure
3. Whether the database can be offline to perform a backup, and if so, the length of time that it can remain offline.
4. The types of resources available to perform backup and recovery.
5. The procedures for undoing changes to the database, if necessary.
6. The cost of buying and maintaining hardware and performing additional backups versus the cost of replacing or re-creating the data lost in a disaster.

Testing Backup and Recovery Strategies:

One of the most important (but also most overlooked) components of the recovery plan is testing. Testing should be done before and after the database that you are supporting is in production. Testing validates that your backups are working, and gives you the peace of mind that recovery will work when a real disaster occurs.

You should document and practice scenarios of certain types of failures so that you are familiar with them, and you should make sure that the methods to recover from these types of failures are clearly defined. At a minimum, you should document and practice the following types of failures:

- Loss of a system tablespace.
- Loss of a non-system tablespace.

- Loss of a current online redo log.
- Loss of the whole database.

Testing recovery should include recovering your database to another server, such as a test or development server.

Test servers are absolutely necessary, however, and businesses that fail to perform this requirement can be at risk of severe data loss or an unrecoverable situation.

Defining Backup and Recovery Procedure:

- Datafile Media Recovery: Restore Datafiles, Apply Redo.
- Complete, Incomplete and Point-In-Time Recovery.
- Automatic Recovery After Instance Failure: Crash Recovery.

Datafile Media Recovery: Restore Datafiles, Apply Redo

Datafile media recovery (often simply called media recovery) is the most basic form of user-initiated data recovery that can be used to recover from a lost or damaged current datafile, SPFILE or control file.

Datafile media recovery can be performed whether you use Recovery Manager or user-managed backup and recovery.

The need to restore a datafile from backup is not detected automatically. The first step in performing media recovery is to manually restore the datafile by copying it from a backup. Once a datafile has been restored from backup, however, the database does automatically detect that this datafile is out of date and must undergo media recovery.

Complete, Incomplete/Point-in-time Recovery:

Complete recovery is recovering a database to the most recent point in time, without the loss of any committed transactions. Generally, the term "recovery" refers to complete recovery.

In incomplete recovery, also known as point-in-time recovery, the goal is to restore the database to its state at some previous target SCN or time. Point-in-time recovery is one possible response to a data loss caused by, for instance, a user error or logical corruption that goes unnoticed for some time.

Automatic Recovery after Instance Failure: Crash Recovery

The crash recovery process is a special form of recovery, which happens the first time an Oracle database instance is started after a crash (or SHUTDOWN ABORT).

In crash recovery, the goal is to bring the datafiles to a transaction-consistent state, preserving all committed changes up to the point when the instance failed.

Crash recovery uses only the online redo log files and current online datafiles, as left on disk after the instance failure. Archived logs are never used during crash recovery, and datafiles are never restored from backup.

The database applies any pending updates in the online redo logs to the online datafiles of your database. The result is that, whenever the database is restarted after a crash, the datafiles reflect all committed changes up to the moment when the failure occurred.

(After the database opens, any changes that were part of uncommitted transactions at the time of the crash are rolled back.)

Parallel Instance Recovery:

The goal of the parallel recovery feature is to use computed and I/O parallelism to reduce the elapsed time required to perform crash recovery, single-instance recovery, or media recovery. Parallel recovery is most effective at reducing recovery time when several data files on several disks are being recovered concurrently.

Parallel recovery can speed up both instance recovery and media recovery.

<u>Parallel Recovery using RMAN</u>

For RMAN, the restore and application of incremental backups are parallelized using channel allocation. The RECOVERY_PARALLELISM parameter determines the number of concurrent processes that participates in recovery. Setting the parameter to 0 or 1 invokes serial recovery. With RESTORE and RECOVER statements, Oracle can automatically parallelize all three stages of recovery.

1. **Restoring Datafiles:** When restoring datafiles, the number of channels you allocate in the RMAN recover script effectively sets the parallelism RMAN uses.
2. **Applying Incremental Backups:** When you are applying incremental backups, the number of channels you allocate determines the potential parallelism.
3. **Applying Redo Logs:** Oracle applies redo logs using a specific number of parallel processes as determined by your setting for the RECOVERY_PARALLELISM parameter. The parameter specifies the number of redo application server processes that participates in the instance or media recovery.

During parallel recovery, one process reads the log files sequentially and dispatches redo information to several recovery processes that apply the changes from the log files to the datafiles.

>>Parallel Instance Recovery

Parallel execution can also improve recovery processing. For the parallel instance recovery, the parallel execution processes must be running when the instance starts up. Use PARALLEL_MIN_SERVERS to define the number of parallel execution servers available for parallel recovery and PARALLEL_MIN_SERVERS to set limit on number of parallel execution processes available for recovery.

>>Parallel Media Recovery

The PARALLEL clause in the RECOVER DATABASE statement determines the degree of parallelism in media recovery. If you do not give value for RECOVERY_PARALLELISM and if it has some non-zero value, it will be used as a default degree of parallelism for the media recovery.

<u>Parallel recovery using Operating System utilities:</u>

You can parallelize instance and media recovery in two ways by:

- Setting the RECOVERY_PARALLELISM parameter
- Specifying RECOVER statement options

Setting the RECOVERY_PARALLELISM parameter:

The RECOVERY_PARALLELISM parameter specifies the number of redo application server processes participating in instance or media recovery. One process reads log files sequentially and dispatches redo information to several recovery processes that apply the changes from the log files to the datafiles. A value of 0 or 1 indicates that recovery is performed serially by one process and the value cannon exceed the value of PARALLEL_MAX_SERVERS parameter.

Specifying RECOVER statement options:

When you specify RECOVER statement to parallelize instance and media recovery, the allocation of recovery processes to instance is operating system specific. The DEGREE keyword of the PARALLEL clause can either signify the number of processes on each instance of a parallel server or the number of processes to spread across all instances.

Recovering from Non-critical Losses:

Loss of file can be caused by:

- User error
- Application error
- Media failure

There are some files whose loss can be tolerated without going through the restore and recover process, known as "non-critical" loss. A noncritical file loss is one where the database can continue to function.

You fix the problem by taking one of these actions:

- Create a new file.
- Rebuild the file.
- Recover the lost or damaged file.

Damage to a TEMP file:

Datafiles are written by DBWn process and if it is not present or damaged, database will stay in mount state and will not upen. Unlike datafiles, tempfiles are written by server processes servicing the sessions that need some temporary space. So, even if a temp file is not available at startup time, the database will still open. DBWn will however, write a message to the alert log. The problem will only be apparent when a session require some temporary space.

Restoring a Temporary tablespace:

Temporary tablespaces, and the tempfiles that makes them up, cannot be backed up. The Recovery Manager (RMAN) ignores them completely if you run the REPORT SCHEMA command to show the files that make up the database; the tempfiles will not be listed.

Since you can't back up a tempfile, you cannot restore it. But you can recreate it as follows:

1. Add another tempfile to the damaged temporary tablespace
2. Take the damaged tempfile offline
3. Drop the damaged file

Eg:

```
alter tablespace temp_ts3 add tempfile 'C:\TEMPFILES\TS3-2.DBF' size 100m;

Tablespace altered.

alter database tempfile 'C:\TEMPFILES\TS3-1.DBF' offline;

database altered.

alter database tempfile 'C:\TEMPFILES\TS3-1.DBF' drop;

database altered.
```

An alternative approach would work at the tablespace level than file level as follows:

1. Create a new temporary tablespace
2. Switch your users over the new temporary tablespace via the ALTER DATABASE command
3. Drop the damaged tablespace

Eg:

```
create temporary tablespace temp_ts4 tempfile 'C:\TEMPFILES\TS4-1.DBF' size 100m;
```

```
tablespace created.

alter database default temporary tablespace temp_ts4;

database altered.

drop tablespace temp_ts3 including contents and datafiles;

Tablespace dropped.
```

Damage to an Online Redo Log File Member

Online redo log is the guarantee that the database will never be corrupted, provided that there is atleast one valid member of the current group (and of any previous groups that is still ACTIVE). If the redo log files are multiplexed, damage to an individual member is not critical but there is a risky position to be in. If there is still a valid member in group, the instance will remain open, and your users will not be aware of any problem—but there will be indications that something is wrong. First, the view V$LOGFILE will show the damaged or missing member as being INVALID. Second, the background processes will report the problem.

Re-creating a Damaged online redo log file member

You have two options: either drop the damaged member and add a replacement number, or clear the group.

In the first option, the damaged member is dropped and then the physical file is deleted from disk. Then a new member is added. This will fix the problem.

In second approach, we use the CLEAR LOGFILE command, which will create a new group by replacing all members and reusing the old member's filenames.

Choice between DROP/ADD and CLEAR depends on circumstances.

If there is no damage to disks and only to the files, CLEAR is simpler to use. Oracle will not let you CLEAR logfilegroup if that needs to be archived, or if one is still current of active. If there is damage to the disk, CLEAR will fail because Oracle will not be able to re-create the file in its original location. This is when one must DROP the original member and then ADD a replacement member in a different disk location.

Damage to Index Tablespace

Index data is real data that must be protected. Any transaction affecting a table will also affect the index on the table. If an index is not available because of media damage, end users will receive a message which is generated whenever any insert or delete operation is attempted against a table with an index in the damaged tablespaces. Apart from errors being reported to user sessions, damage to an index tablespace datafile will also show up through the usual message in the alert log and backgroundtrace files, and entries in the dynamic performance view.

Recovering an Index Tablespace:

An index tablespace can be restored and recovered like any other tablespace, provided that you are running database in archivelog mode. An alternative method rather than repairing damage through restore and recovery imply the dropping of tablespace, re-creating it and regenerating indexes.

The routine is as follows:

1. Take the damaged tablespace offline

2. Determine which indexes were in the damaged tablespace
3. Drop the tablespace and delete its datafiles
4. Create a new tablespace
5. Generate all the indexes in it

To see indexes in tablespace:

SQL> select owner, segment_name from dba_segments where tablespace_name='INDX' and segment_type='INDEX';

To drop tablespace:

SQL> drop tablespace indx including contents and datafiles;

To create a new tablespace:

SQL> create tablespace indx datafile 'C:\DATAFILES\INDX-1.DBF' size 100m extent management local uniform size 128k segment space management autonologging;

Damage to the Password file:

The password file is one of the files that cannot be backed up by RMAN. Loss of password file is not critical. It is always possible to connect to a database with the SYSDBA privilege with operating authentication. If the password file is damaged, the database will continue to work normally, except that the remote SYSDBA connections will not be possible. But if the database is shut down, then startup will fail when the instance tries to read the password file. To get around this problem, set the REMOTE_LOGIN_PASSWORDFILE instance parameter to NONE. This will prevent Oracle from looking for a passwordfile and the instance will open normally.

SQL> alter system set remote_login_passwordfile=none scope=spfile; and then restart.

Replacing the password file:

It is possible to backup the password file with normal operating system backup procedures. The system administrators' regular backup of ORACLE_HOME directory will have copies of it and so you can restore it with no problem. An alternative is to re-create it by re-running the command used to create it in the first place as.

orapwd file=<filename> password=<password> entries=<max_users>

Having re-created the password file, reset the REMOTE_LOGIN_PASSWORDFILE parameter to use it.

BACKUP PRINCIPLES

- **Physical and Logical Backups**
- **Whole database and partial database Backups**
- **Consistent and Inconsistent Backups**
- **Offline and Online Backups**
- **RMAN and USER-Managed Backups**

Physical Backups	**Logical Backups**
Physical backups are backups of physical database files: datafiles and control files.	Logical backups are exports of schema objects into a binary file
Physical backups are divided into two categories: image copies and backups in a proprietary format.	Logical backups have two utilities: Import and Export to move Oracle data in and out of system file.
Image copy is an exact duplicate of a datafile, control file or archived log using COPY command and restore using RESTORE command.	Import is the utility that reads export files and restores the corresponding data into an existing database.

The BACKUP command generates a backup set which is a logical object containing one or more backup pieces where each backup piece is a physical file in a proprietary binary format.	Export writes data from an Oracle database to binary operating system files. These export files store information about schema objects eg: tables and stored procedures

Whole Database Backups	**Partial Database Backups**
Whole database backup includes backups of the current control file along with all datafiles.	A partial database backup is a one which takes backup of some specific files while the database is open or shut down. Such as: Tablespace Backups, Datafile Backups, Control File Backups
Whole database backup must be done when backup is carried out for the first time.	Partial database backup is done only after whole database backup has been done once.
Whole database backup takes backup of each and every files in the database.	Partial database backup can be either incremental or differential which will take backup of those files where changes has been recorded.

Consistent Backup	**Inconsistent Backup**
It is a backup of one or more database file that you make after the database has been closed with a clean SHUTDOWN command.	It is a backup of one or more database files that you make while the database is open or after the database has shut down abnormally.

In this backup, read/write datafiles and control files are checkpointed with respect to the same **System Change Number (SCN).**	In this backup, all read/write datafiles and control files are not checkpointed with respect to same SCN.
A consistent backup is one in which database has been shut down. It is also called **cold backup**.	An inconsistent backup is taken when the database is up and running. This is also called a **hot backup.**
The database is guaranteed to be in a consistent state because no one can possibly be using the database; it is down.	While the backup is being performed, the database can still undergo change. A datafile in your backup can have old transactions and new transactions. This means the datafile is not consistent with a single point in time.

Offline Backup	**Online Backup**
offline backup is a backup of the database while it is not running.	an online backup allows you to backup the database while users are working.
to perform our backup we will shutdown the database from RMAN and then mount the database.	to perform online backup you will need to put your database in ARCHIVELOG mode
Also known as cold backup	also known as hot backup
Process to take backup: RMAN>shutdown immediate RMAN>startup mount RMAN>backup database; RMAN>sql ?alter database open?;	RMAN>backup database plus archivelog delete input; This command will backup your database. Along with the database backup, it will backup all the archived redo logs that have

	been generated by your database
While the database is "offline," the following files should be backed up: • All datafiles. • All controlfiles. • All archived redo log files. • The init.ora file or server parameter file (SPFILE).	While the database is open, the following files can be backed up: • All datafiles • All archived redo log files • One control file, via the **alter database backup controlfile** • The server parameter file (SPFILE

Unit 5

Oracle Recovery Manager (RMAN):

- Database corruption
- automatic storage management
- RMAN configuration
- Database Archival

Database Corruption:

A data block is corrupted when it is not in a recognized Oracle Database format, or its contents are not internally consistent. Data block corruption can damage internal Oracle control information or application and user data, leading to crippling loss of critical data and services. Block corruptions may affect only a single block or a large portion of the database (making it essentially unusable). While relatively rare, corruptions are inevitable, but Oracle provides a complete set of technologies to prevent and mitigate block corruptions.

A database block is corrupted when its content has changed from what Oracle Database expects to find. If not prevented or repaired, block corruption can bring down the database and possibly result in the loss of key business data.

While you cannot prevent all block corruptions, there is a comprehensive set of data protection solutions that you can implement to address most of them. Oracle offers sophisticated solutions—such as:

- Oracle Data Guard
- Data Recovery Advisor
- Oracle Flashback
- Oracle Recovery Manager (RMAN)
- Automatic Diagnostic Recovery (ADR)
- Oracle Secure Backup
- The MAA Advisor component of Oracle Enterprise Manager Grid Control
- Exadata Storage

These features offer database optimized ways to protect your data and ensure high availability of your data and hence, the application.

How Corruption Manifests Itself

When Oracle issues a write operation, it moves through the following I/O sequence:

$\Rightarrow$ to the file system

⇒to the volume manager

⇒to the device driver

⇒to the Host-Bus Adapter

⇒to the storage controller

⇒to the disk drive where data is written

Hardware failures or bugs in any layer can result in corrupt data being written to disk, or good data not written to disk (termed a “lost write”) yet reported as written to Oracle.

Physical and Logical Corruptions

Data corruption can manifest itself as a physical or a logical corruption:

● **Physical Corruption** of a block manifests as an invalid checksum or header, or when the block contains all zeroes. When that happens, the database will not recognize the block as a valid Oracle block, regardless of its content. A physical corruption is also called a media corruption.

● **Logical Corruption** happens when a data block has a valid checksum, etc., but the block contents are logically inconsistent. Logical block corruption can also occur when the structure below the beginning of the block (below the block header) is corrupt. In this case, the block checksum is correct but the block structures may be corrupt. Logical corruption can also result from a lost write in which a version of the block that was sent to disk somehow was not actually written. The result is that the version of that block on disk is older than the version in the buffer cache. Lost writes are usually caused by bugs in the operating system or hardware.

Intrablock and Interblock Corruptions

The data blocks to which we refer are Oracle data blocks, which are comprised of multiple operating system blocks that make up the database. The data blocks are stored on disk, but are also temporarily stored in the buffer cache in memory. Thus, corruptions do not always appear on disk and can be related to memory and transient in nature.

● for **intrablock corruption**, the corruption occurs in the block itself and can be either a physical or a logical corruption.

● for **interblock corruption**, the corruption occurs between blocks and can only be a logical corruption.

Automatic Storage Management (ASM):

Automatic Storage Management (ASM) is an integrated, high-performance database file system and disk manager. ASM is based on the principle that the database should manage storage instead of requiring an administrator to do it. ASM eliminates the need for you to directly manage potentially thousands of Oracle database files.

ASM groups the disks in your storage system into one or more disk groups each of which comprises of several physical disks that are controlled as a single unit. The physical disks are knows as ASM disks, while the files that reside on the disks are known as ASM files. The locations and names for the files are controlled by ASM.

ASM provides the following benefits:

- **Striping**—ASM spreads data evenly across all disks in a disk group to optimize performance and utilization. This even distribution of database files eliminates the need for regular monitoring and I/O performance tuning.

- **Mirroring**— Mirroring means keeping redundant copies, or mirrored copies, of each extent of the file, to help avoid data loss caused by disk failures. ASM can increase availability by optionally mirroring any file. ASM mirrors at the file level, unlike operating system mirroring, which mirrors at the disk level. The mirrored copy of each file extent is always kept on a different disk from the original copy. If a disk fails, ASM can continue to access affected files by accessing mirrored copies on the surviving disks in the disk group.

  ASM supports 2-way mirroring, where each file extent gets one mirrored copy, and 3-way mirroring, where each file extent gets two mirrored copies.

- **Online storage reconfiguration and dynamic rebalancing**—ASM permits you to add or remove disks from your disk storage system while the database is operating. When you add a disk, ASM automatically redistributes the data so that it is evenly spread across all disks in the disk group, including the new disk. This redistribution is known as **rebalancing**. It is done in the background and with minimal impact to database performance. When you request to remove a disk, ASM first rebalances by evenly relocating all file extents from the disk being removed to the other disks in the disk group.
- **Managed file creation and deletion**—ASM further reduces administration tasks by enabling files stored in ASM disk groups to be Oracle-managed files. ASM automatically assigns filenames when files are created, and automatically deletes files when they are no longer needed.

In summary ASM provides following functionalities:

- Manages group of disks, called disk groups.
- Manages disk redundancy within a disk group.
- Provides near-optimal I/O balancing without any manual tuning.

- Enables management of database objects without specifying mount points and filenames.
- Supports large files.

Recovery Manager (RMAN):

Recovery Manager is a client application that performs backup and recovery operation using the database server sessions. It is an Oracle utility that can back up, restore, and recover database files. It is a feature of the Oracle database server and does not require separate installation.

RMAN stores metadata about its operations in the control file of the target database and, optionally, in a recovery catalog schema in an Oracle database.

You can invoke RMAN as a command-line executable from the operating system prompt or use some RMAN features through the Enterprise Manager GUI.

RMAN was introduced in Oracle release 8.0 and is not compatible with Oracle databases prior to release 8.0.

Why use RMAN?

- It's FREE (with your Oracle license).
- Backups can be checked for corruption before it become a problem.
- Minimize downtime after failure.
- Recover single blocks of data.
- Backup directly to tape.
- Creating duplicate instances, including standbys for failover.

- Backups can be taken online without the additional overhead of traditional online backups.
- Automatically includes new datafiles and tablespaces without manual intervention.
- Only backs up used data blocks.
- Allows for compressed backups.
- Works with third-party media management.
- Allows for centralized management and reporting of backups.

RMAN processes most commands in two phases:

- **Compilation phase**
  During compilation phase, RMAN determines which objects the command will access. Then, RMAN constructs a sequence of remote procedure call (RPCs) that instruct the server sessions on the target database to perform the desired operation.
- **Execution phase**
  During the execution phase, RMAN sends the RPC calls to the target database, monitors their progress, and collects the results. If more than one channel is allocated, then RMAN can execute certain commands in parallel so that all of the channels' target database sessions are concurrently executing an RPC call.

Starting RMAN and Connecting to a Database

The RMAN client is started by issuing the rman command at the command prompt of your operating system. RMAN then displays a prompt for your commands as shown in the following example:

```
% rman

RMAN>
```

RMAN connections to a database are specified and authenticated in the same way as SQL*Plus connections to a database. The only difference is that RMAN connections to a target or auxiliary database require the SYSDBA privilege. The AS SYSDBA keywords are implied and cannot be explicitly specified. See Oracle Database Administrator's Guide to learn about database connection options for SQL*Plus.

You can connect to a database with command-line options or by using the CONNECT TARGET command. The following example starts RMAN and then connects to a target database through Oracle Net, AS SYSDBA is not specified because it is implied. RMAN prompts for a password.

```
% rman

RMAN> CONNECT TARGET SYS@prod

target database Password: password

connected to target database: PROD (DBID=39525561)
```

The following variation starts RMAN and then connects to a target database by using operating system authentication:

```
% rman

RMAN> CONNECT TARGET /

connected to target database: PROD (DBID=39525561)
```

To quit the RMAN client, enter EXIT at the RMAN prompt:

```
RMAN> EXIT
```

RMAN Environment

The RMAN environment consists of the utilities and databases that play a role in backing up your data. At a minimum, the environment for RMAN must include the following components:

- **A target database**

An Oracle database to which RMAN is connected with the TARGET keyword. A target database is a database on which RMAN is performing backup and recovery operations. RMAN always maintains metadata about its operations on a database in the control file of the database. The RMAN metadata is known as the RMAN repository.

- **The RMAN client**

An Oracle Database executable that interprets commands, directs server sessions to execute those commands, and records its activity in the target database control file. The RMAN executable is automatically installed with the database and is typically located in the same directory as the other database executables. For example, the RMAN client on Linux is located in $ORACLE_HOME/bin.

Some environments use the following optional components:

- **A fast recovery area**

A disk location in which the database can store and manage files related to backup and recovery. You set the fast recovery area location and size with the DB_RECOVERY_FILE_DIST and DB_RECOVERY_FILE_DEST_SIZE initialization parameters.

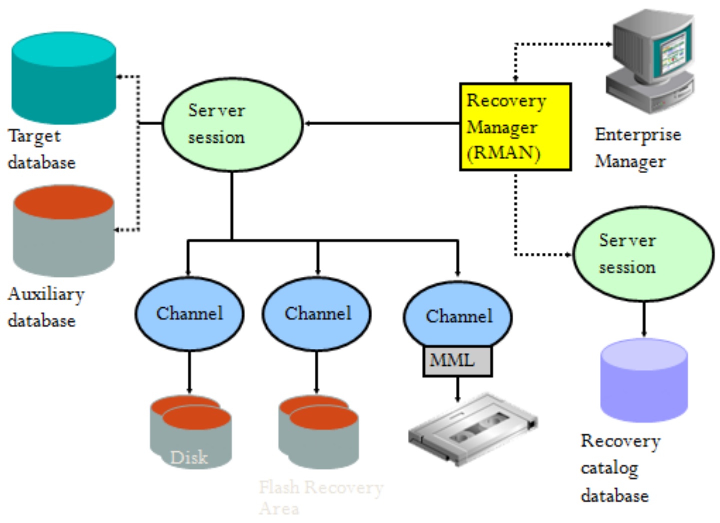
- **A media manager**

An application required for RMAN to interact 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. Media management devices are sometimes called SBT (system backup to tape) devices.

- **A recovery catalog**

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. The database may overwrite older records in the control file, but RMAN maintains records forever in the catalog unless the records are deleted by the user.

RECOVERY MANAGER COMPONENTS:



Target Database

The target database is the database that RMAN is backing up, restoring, or recovering.

Sushant Thapa
2070 Batch
+977-98442-06228
Himalaya College of Engineering

You can use a single recovery catalog in conjunction with multiple target databases. For example, assume that your data center contains 10 databases of varying sizes. You can use a single recovery catalog located in a different data center to manage the metadata from all of these databases.

RMAN Repository:

The RMAN repository is the collection of metadata about the target databases that RMAN uses for backup, recovery, and maintenance. RMAN always stores this information in records in the control file. The version of this information in the control file is the authoritative records of RMAN's backups of your database. This is one reason why protecting your control file is an important part of your backup strategy. RMAN can conduct all necessary backup and recovery operations using just the control file to store the RMAN repository information, and maintain all records necessary to meet your configured retention policy.

Among other things, RMAN stores information about:

- Backup sets and pieces
- Image copies (including archived redo logs)
- Proxy copies
- The target database schema
- Persistent configuration settings

You can access this metadata by issuing LIST, REPORT, and SHOW commands in the RMAN interface, or by using SELECT statements on the catalog views (only if you use a recovery catalog)

Media Management Interface:

To store backups on tape, RMAN requires a media manager. A media manager is a software program that loads, labels, and unloads sequential media such as tape drives used to back up and recover data.

Backup Types:

RMAN supports a number of different backup methods, depending on your availability needs, the desired size of your recovery window, and the amount of downtime you can endure while the database or a part of the database is involved in a recovery operation.

Consistent and Inconsistent Backup:

A physical backup can be classified by being a consistent or an inconsistent backup. In a consistent backup, all datafiles have the same SCN; in other words, all changes in the redo logs have been applied to the datafiles. Because an open database with no uncommitted transactions may have some dirty blocks in the buffer cache, it is rare that an open database backup can be considered consistent. As a result, consistent backups are taken when the database is shut down normally or in a MOUNT state.

In contrast, an inconsistent backup is performed while the database is open and users are accessing the database. Because the SCNs of the datafiles typically do not match when an inconsistent backup is taking place, a recovery operation performed using an inconsistent backup must rely on both archived and online redo log files to bring the database into a consistent state before it is opened. As a result, a database must be in ARCHIVELOG mode to use an inconsistent backup method.

Full and Incremental Backup:

Full backups include all blocks of every datafile within a tablespace or a database; it is essentially a bit-for-bit copy of one or more datafiles in the database. Either RMAN or an operating system command can be used to perform a full backup, although backups performed outside of RMAN must be cataloged with RMAN before they can be used in an RMAN recovery operation.

An alternative strategy to relying on full backups with archived redo logs is to use incremental backups along with archived redo logs for recovery. The initial incremental backup is known as a level 0 incremental backup. Each incremental backup after the initial incremental backup (also known as a level 1 incremental backup) contains only changed blocks and as a result takes less time and space.

Incremental level 1 backups can either be

- Cumulative Incremental or
- Differential Incremental

RMAN Recovery Catalog

The RMAN architecture revolves primarily around the recovery catalog. The recovery catalog is an Oracle database and associated objects in a schema that are created to manage RMAN backups. Although the recovery catalog is not required to use RMAN, some of the backup/recovery functions are not available without it being set up correctly. Generally, to set up a catalog, you identify the database in which the catalog will exist. This database should be separate from other Oracle databases. Once you have identified or created the database that will contain the recovery catalog, you create a user in that database and grant that user the

RECOVERY_CATALOG_OWNER role. Then you will create the recovery catalog, create any backup scripts and begin to use the recovery catalog.

So, what are the benefits of using the recovery catalog? RMAN does not require the recovery catalog in most cases. Without the recovery catalog in place however, these RMAN features are not available:

- Tablespace point-in-time recovery
- Stored scripts
- Recovery when the control file is lost or damaged

The bottom line is that Oracle strongly recommends that you use the recovery catalog.

Once you have set up the recovery catalog, you use the RMAN executable in concert with RMAN manual commands or stored scripts to back up, recover, and report on backups. The next few sections of this topic look in more detail at the setup, backup, recovery, and reporting functions of RMAN. Finally, the topic returns (in much more detail) to the recovery catalog and addresses maintenance issues and reporting from the views Oracle supplies for the DBAs. First, let's look at how to set up the catalog and use it to register, back up, and recover databases.

RMAN Channels

Before you can execute a backup or recovery using RMAN, you must allocate a channel between the backup processes and the operating system. The following operations in RMAN must have at least one channel allocated to operate correctly:

- BACKUP
- COPY
- RESTORE
- RECOVER

To allocate a channel, you use the RMAN command ALLOCATE CHANNEL. When you allocate the channel, you also specify the type of device that will be used for the operation that will occur through that channel. Multiple channels can be allocated, allowing for multiple backup sets or file copies to run in parallel by the execution of just a single RMAN command. Each time you issue an ALLOCATE CHANNEL command, a separate connection between the backup processes and the operating system is created.

RMAN Vs Traditional Backup Methods:

- **Skip unused blocks:**
  Blocks that have never been written to, such as blocks above the high water mark (HWM) in a table, are not backed up by RMAN when the backup is an RMAN backupset. Traditional backup methods have no way to know which blocks have been used.

- **True incremental backups:**
  For any RMAN incremental backup, unchanged blocks since the last backup will not be written to the backup file. This saves a significant amount of disk space, I/O time, and CPU time.

- **Block-level recovery:**
  To potentially avoid downtime during a recovery operation, RMAN supports *block-level recovery for recovery operations that only need to restore or repair a* small number of blocks identified as being corrupt during the backup operation.

- **Multiple I/O channels:**
  During a backup or recovery operation, RMAN can utilize many I/O channels, via separate operating system processes, to perform concurrent I/O. Traditional backup methods, such as a Unix **cp**

command or an Oracle export, are typically single-threaded operations.

- **Cataloging :**
  A record of all RMAN backups is recorded in the target database control file, and optionally in a recovery catalog stored in a different database.

- **Scripting capabilities :**
  RMAN scripts can be saved in a recovery catalog for retrieval during a backup session.

- **Encrypted backups:**
  RMAN uses backup encryption integrated into Oracle Database 11***g*** to store encrypted backups. Storing encrypted backups on tape requires the Advanced Security Option.

Backup Settings:

Default Device Type: Determines whether you will be backing up to disk or tape

- Options are mutually exclusive.
- Default may be overridden with the backup device type. parameter

Syntax:

- CONFIGURE DEFAULT DEVICE TYPE TO DISK;
- CONFIGURE DEFAULT DEVICE TYPE TO SBT;

Format:

Multiple channels may be allocated.

Various format options available

- %U: System generated unique filename (default).
- %F: DBID, day, month, year, sequence.
- %s: backup set.
- %p: backup piece.

Syntax:

- CONFIGURE CHANNEL DEVICE TYPE DISK FORMAT '/u01/backup/PPRD/rman/PPRD_%U';

Backup Method:

- Used to further define how backups will be carried out by default device type
- If compressed backup sets are chosen, backup files will be smaller, but it will take longer to restore from them.
- Syntax:
    - CONFIGURE DEVICE TYPE DISK BACKUP TYPE TO BACKUPSET;
    - CONFIGURE DEVICE TYPE DISK BACKUP TYPE TO COMPRESSED BACKUPSET;
    - CONFIGURE DEVICE TYPE DISK BACKUP TYPE COPY;

RMAN - Retention Policies

When making backups, it's a good idea to define how long you want to keep a given backup set. A retention policy is a policy that revolves around how long backup sets are kept. In Oracle, RMAN allows you to define a retention policy regarding backup sets. There are two types of retention policies available in RMAN:

- Recovery window
- Redundancy

These policies are mutually exclusive of each other, and only one policy can be defined at a time. Policies are established with the CONFIGURE RETENTION POLICY command.

Note: Media manager retention policies can cause havoc if they are not properly synchronized with RMAN retention policies! For example, when using optimization, you could have your media manager software remove a backup, and RMAN would not know about it. This might cause the database to be unrecoverable.

Recovery Window Retention Policies:

The recovery window retention policy is used to ensure that a backup is available that will facilitate a recovery within the defined time window. For example, if the recovery window is set to seven days, then no backup will be reported as obsolete unless it is older than 7 days, the defined recovery window retention time.

To set a recovery window retention policy, use the CONFIGURE RETENTION POLICY command, as shown in the following example:

```
CONFIGURE RETENTION POLICY TO RECOVERY WINDOW OF 7 DAYS;
```

In this case, we have configured a retention policy of seven days. When the DELETE OBSOLETE command is executed, only backups older than seven days will be reported as obsolete.

Note: If the control_file_record_keep_time database parameter value is less than the time set for the retention policy, you will need to use a recovery catalog.

Redundancy Retention Policies

The redundancy retention policy defines a fixed number of backups that need to be retained by RMAN. If there are a number of backups later than the number defined by the retention policy, then those backups can be deleted using the DELETE OBSOLETE command, which we will discuss in the next section.

To set a redundancy retention policy, use the configure retention policy command, as shown here:

```
CONFIGURE RETENTION POLICY TO REDUNDANCY 7;
```

Use care when choosing your retention policies. While the redundancy policy will work fine if you are sure you will do just a certain number of backups per day, it is possible that you could easily loose backups that you don't intend to if you should back up your database multiple times on the same day (say, for example, because you just upgraded the database). Generally a recovery window retention policy is best for production databases to assure you can recover to the time you want. Redundancy policies might be more appropriate for test and development environments, in which backup space might be at a premium.

Remove the Retention Policy

If you define a policy and you wish to remove it, you can issue the CONFIGURE RETENTION POLICY command. To remove the existing retention policy, issue the following command

```
CONFIGURE RETENTION POLICY TO NONE;
```

Optionally, you can return the retention policy to the default policy, which is one day, by issuing the command below:

```
CONFIGURE RETENTION POLICY TO DEFAULT;
```

DELETE OBSOLETE

The RMAN DELETE OBSOLETE command is used in conjunction with the defined retention policy to remove obsolete backups. The example below will remove all backups that are obsolete based on the currently existing policy:

```
DELETE OBSOLETE;
```

The delete obsolete command also can take parameters to define a recovery window or backup redundancy that is different from the existing policy. If your default recovery window is seven days and you want to remove backups that are five days old, then you could issue this command:

```
DELETE OBSOLETE RECOVERY WINDOW OF 5 DAYS;
```

Also, the delete obsolete command provides a method of removing orphaned backups that were created from a different incarnation of the database. An example of this command would look like this:

```
DELETE OBSOLETE ORPHAN;
```

Unit 6

DB Performance Tuning:

- Introduction
- Tuning methodology
- Tuning concepts
- AADM (Automatic Database Diagnostic Monitor)
- SQL Tuning Advisor

- AWR Report
- Virtual Private Database
- Policy types, selective columns, column masking

Performance Tuning:

One of the biggest responsibilities of a DBA is to ensure that the Oracle database is tuned properly. The Oracle RDBMS is highly tunable and allows the database to be monitored and adjusted to increase its performance.

One should do performance tuning for the following reasons:

- The speed of computing might be wasting valuable human time (users waiting for response);
- Enable your system to keep-up with the speed business is conducted; and
- Optimize hardware usage to save money (companies are spending millions on hardware).

As performance bottlenecks are identified during load testing and performance testing, these issues are commonly rectified through a process of performance tuning. Performance tuning can involve configuration changes to hardware, software and network components. A common bottleneck is the configuration of the application and database servers. Performance tuning can also include tuning SQL queries and tuning an applications underlying code to cater for concurrency and to improve efficiency.

Performance tuning can result in hardware changes being made. This is a last resort; ideally tuning changes will result in a reduction of resource utilization.

Instance Tuning

When considering instance tuning, take care in the initial design of the database to avoid bottlenecks that could lead to performance problems. In addition, you must consider:

- Allocating memory to database structures
- Determining I/O requirements of different parts of the database
- Tuning the operating system for optimal performance of the database

After the database instance has been installed and configured, you must monitor the database as it is running to check for performance-related problems.

Performance Principles

Performance tuning requires a different, although related, method to the initial configuration of a system. Configuring a system involves allocating resources in an ordered manner so that the initial system configuration is functional.

Tuning is driven by identifying the most significant bottleneck and making the appropriate changes to reduce or eliminate the effect of that bottleneck. Usually, tuning is performed reactively, either while the system is in preproduction or after it is live.

Tuning Methodology:

For best results, tune during the design phase, rather than waiting to tune after implementing your system. You can optimize performance by system tuning. This is the process of making adjustments to the way in which various system resources are used so that they correspond more closely to the overall usage patterns of the system. You can improve the overall response time of the system by locating and removing system bottlenecks.

Two different tuning methods are:

- Proactive Tuning While Designing and Developing Systems
- Reactive Tuning to Improve Production Systems

Proactive Tuning:

The proactive tuning begins with the initial phase of design and development of system. It helps developer to design and develop a system that can perform well and as a result minimize the initialization and on-going administrative cost.

In proactive tuning, the problems are fixed before they occur or at least before they are widely noticed. The main goal of proactive tuning is to detect and repair problems before they affect the system operation. Proactive tuning usually occurs on a regularly scheduled interval, where several performance statistics are examined to identify whether the system behavior and resource usage has changed.

Proactive tuning is done by following given steps:

1. Tune the business rules.
2. Tune the data design
3. Tune the application design
4. Tune the logical structure of database
5. Tune database operations
6. Tune the access paths

7. Tune memory allocation
8. Tune I/O and physical structures
9. Tune resource contention
10. Tune the underlying platforms

After completing these steps, reassess your database performance, and decide whether further tuning is necessary.

1. Tune the business rules

For optimal performance, you may need to adapt business rules. These concern the high-level analysis and design of an entire system. Configuration issues are considered at this level, such as whether to use a multi-threaded server system-wide. In this way, the planners ensure that the performance requirements of the system correspond directly to concrete business needs.

2. Tune the data design

In the data design phase, you must determine what data is needed by your applications. You must consider what relations are important, and what their attributes are. Finally, you need to structure the information to best meet performance goals.

The database design process generally undergoes a normalization stage when data is analyzed to eliminate data redundancy. With the exception of primary keys, any one data element should be stored only once in your database. After the data is normalized, however, you may need to denormalize it for performance reasons. You might decide that the database should retain frequently used summary values.

Another data design consideration is avoiding data contention.

3. Tune the application design

Business executives and application designers should translate business goals into an effective system design. Business processes concern a particular application within a system, or a particular part of an application.

At this level, you can also consider the configuration of individual processes.

4. Tune the logical structure of database

After the application and the system have been designed, you can plan the logical structure of the database. This primarily concerns fine-tuning the index design to ensure that the data is neither over- nor under-indexed. In the data design stage (Step 2), you determine the primary and foreign key indexes. In the logical structure design stage, you may create additional indexes to support the application.

Performance problems due to contention often involve inserts into the same block or incorrect use of sequence numbers. Use particular care in the design, use, and location of indexes, as well as in using the sequence generator and clusters.

5. Tune database operations

Before tuning the Oracle server, be certain that your application is taking full advantage of the SQL language and the Oracle features designed to enhance application processing. Use features and techniques such as the following, based on the needs of your application:

- Array processing
- The Oracle optimizer
- The row-level lock manager
- PL/SQL

Understanding Oracle's query processing mechanisms is also important for writing effective SQL statements.

Step 1: Find the Statements that Consume the Most Resources

Step 2: Tune These Statements To Use Fewer Resources

6. Tune the access paths

Ensure that there is efficient data access. Consider the use of clusters, hash clusters, B*-tree indexes, bitmap indexes, and optimizer hints. Also consider analyzing tables and using histograms to analyze columns in order to help the optimizer determine the best query plan.

Ensuring efficient access may mean adding indexes or adding indexes for a particular application and then dropping them again. It may also mean re-analyzing your design after you have built the database. You may want to further normalize your data or create alternative indexes. Upon testing the application, you may find that you are still not obtaining the required response time. If this happens, then look for more ways to improve the design.

7. Tune memory allocation

Appropriate allocation of memory resources to Oracle memory structures can have a positive effect on performance.

Although you explicitly set the total amount of memory available in the shared pool, the oracle dynamically sets the size of each of the following structures contained within it:

- The data dictionary cache
- The library cache
- Context areas (if running a multi-threaded server)

You can explicitly set memory allocation for the following structures:

- Buffer cache
- Log buffer

- Sequence caches

Proper allocation of memory resources improves cache performance, reduces parsing of SQL statements, and reduces paging and swapping.

Process local areas include:

- Context areas (for systems not running a multi-threaded server)
- Sort areas
- Hash areas

Be careful not to allocate to the system global area (SGA) such a large percentage of the machine's physical memory that it causes paging or swapping.

8. Tune I/O and physical structures

Disk I/O tends to reduce the performance of many software applications. The Oracle server, however, is designed so that its performance is not unduly limited by I/O. Tuning I/O and physical structure involves these procedures:

- Distributing data so that I/O is distributed to avoid disk contention.
- Storing data in data blocks for best access: setting an adequate number of free lists and using proper values for PCTFREE and PCTUSED.
- Creating extents large enough for your data, to avoid dynamic extension of tables. This adversely affects the performance of high-volume OLTP applications.
- Evaluating the use of raw devices.

9. Tune resource contention

Concurrent processing by multiple Oracle users may create contention for Oracle resources. Contention may cause processes to wait until resources are available. Take care to reduce the following types of contention:

- Block contention
- Shared pool contention
- Lock contention
- Pinging (in a parallel server environment)
- Latch contention

10.Tune the underlying platforms

See your platform-specific Oracle documentation for ways to tune the underlying system. For example, on UNIX-based systems you might want to tune the following:

- Size of the UNIX buffer cache
- Logical volume managers
- Memory and size for each process

Reactive Tuning:

Reactive tuning is a response to a known or reported problem. You may begin tuning performance metrics in reaction to user complaints about

- response time,
- instance failures, or
- errors found in the alert log.

Reactive tuning is inevitably necessary sometimes but your goal should be proactive approach to system maintenance.

It is possible to reactively tune an existing production system. To take this approach, start at the bottom of the method and work your way up, finding and fixing any bottlenecks. A common goal is to make Oracle run faster on the given platform. You may find, however, that both the Oracle server and the operating system are working well. To get additional performance gains, you may need to tune the application or add resources. Only then can you take full advantage of the many features Oracle provides that can greatly improve performance when properly used in a well-designed system.

Even the performance of well-designed systems can degrade with use. Ongoing tuning is, therefore, an important part of proper system maintenance.

Tuning concepts:

Understanding trade-offs:

Even with the application of a proven tuning methodology that utilizes extensive benchmarks, most tuning efforts involve some degree of compromise. This occurs because every Oracle server is constrained by the availability of three key resources: CPU, Disk (I/O), and memory.

CPU:

Tuning Oracle's memory and I/O activity will provide little benefit if the server's processor is already overburdened. However, even in high-end multi-CPU servers, considerations should be given to the impact that changing memory or device configurations will have on CPU utilization. Several Oracle Server Configuration parameter changes dynamically when CPUs are added or removed from the server.

Disk (I/O):

The more Oracle server activity occurs in memory, the lower the physical I/O will be. However, placing too many demands on the available memory by oversizing Oracle's memory structures can cause undesirable additional I/O in the form of Operating System paging and swapping. Modern disk-caching hardware and software also complicate database I/O tuning, since I/O activity on these systems may result in reads from the disk controller's cache and not directly from disk.

Memory:

The availability of memory for Oracle's memory structures is key to good performance. Managing that memory is important so that it is used to maximum benefit without wasting any that could be better used by other server processes. Oracle offers several memory tuning options that help you make the most efficient use of the available memory.

Common Tuning Problem Areas:

While examining your system for possible performance improvement, it is important to first examine those areas that are most likely the cause of poor performance.

These areas include:

- Poorly written application SQL
- Inefficient SQL execution plans
- Improperly sized System Global Area (SGA) memory structures
- Excessive file I/O
- Inordinate waits for access to database resources.

ADDM(Automatic Database Diagnosis Monitor)

For Oracle, the statistical data needed for accurate diagnosis of a problem is saved in the Automatic Workload Repository (AWR). The Automatic Database Diagnosis Monitor is the one that analyzes the AWR data on a regular basis and then locates the root causes of performance problems, provides recommendations for correcting any problems and identifies non-problem area of the system. Because AWR is a repository of historical performance data, ADDM can be used to analyze performance issues after the event, often saving time and resources reproducing a problem. The analysis is performed top down, first identifying symptoms and then refining them to reach the root cause of performance problem.

The goal of the analysis is to reduce a single throughput metric called DB time. DB time is the cumulative time spent by the database server in processing user requests. It includes wait time and CPU time of all non-idle user sessions.

ADDM doesn't target tuning of individual user response times; for that use tracing technique. By reducing DB time, the database server is able to support more user requests using the same resources, which increases throughput.

Some of the types of problem that ADDM considers includes are:

- **CPU bottlenecks-** Is the system CPU bound by Oracle or some other application?
- **Undersized Memory Structures**- Are the Oracle memory structures, such as the SGA, PGA, and buffer cache adequately sized?
- **I/O capacity issues**- Is the I/O subsystem performing as expected?
- **Concurrency Issues**- Are there buffer busy problem?

- **Database Configuration Issues**- Is there any evidence of incorrect sizing of log files, archiving issues, excessive checkpoints or sub-optimal parameter settings?

ADDM also documents the non-problem areas of the system. In addition to problem diagnostics, ADDM recommends possible solutions. When appropriate ADDM recommends multiple solutions for the DBA to choose from. ADDM considers variety of changes to a system while generating its recommendations that includes:

- Hardware Changes
- Database Configuration
- Schema changes
- Application changes
- Using other advisors

SQL Tuning Advisor

The automatic SQL Tuning capabilities are exposed through a server utility called the SQL Tuning Advisor that takes one or more SQL statements as an input and invokes the Automatic Tuning Optimizer to perform SQL tuning on the statement. The output of the SQL tuning advisor is in the form of advice or recommendations, along with a rationale for each recommendations and its expected benefit. The recommendation relates to collection of statistics on objects, creation of new indexes, restructuring of the statement, or creation of SQL profile. A user can choose to accept the recommendation to complete the tuning of SQL statements.

The SQL tuning advisor's input can be a single SQL statement or a set of statements. For tuning multiple statements, a SQL Tuning Set (STS) has to be first created. An STS is a database object that stores statements along with their execution context. It can be created manually using command line APIs or automatically using Oracle Enterprise Manager.

Input Sources:

The input for the SQL tuning advisor can come from several sources such as:

- **Automatic Database Diagnostic Monitor**
  It is the primary input source for SQL Tuning Advisor. By default, ADDM runs pro-actively once every hour and analyzes key statistics gathered by the AWR over last hour to identify any performance problem including High-load SQL statements. If high-load SQL statements are identified, ADDM recommends running SQL tuning advisor in SQL.

- **High-load SQL Statements**
  The AWR takes regular snapshots of system activities including high-load SQL statements ranked by relevant statistics, such as CPU consumption and wait time. A user can view the AWR and identify the high-load SQL of interest and run SQL tuning advisor on them.

- **Cursor Cache**
  This source is used for tuning recent SQL statements that are yet to be captured in the AWR. The cursor cache and AWR together provides the capability to identify and tune high-load SQL statements from the current time going as far back as the AWR retention allows, which is by default 7 days.

- **SQL Tuning Set**
  It is a user defined set of SQL statements that are yet to be deployed, with the goal of measuring their individual performance, or identifying the ones whose performance falls short of expectation. When a set of SQL statements are used as input, a SQL tuning set has to be first constructed and stored.

Sushant Thapa
2070 Batch
+977-98442-06228
Himalaya College of Engineering

<u>**Tuning Options:**</u>

SQL tuning advisor provides options to manage the scope and duration of a tuning task. The scope of tuning task can be set to **limited** or **comprehensive.**

- For limited, the SQL tuning advisor produces recommendations based on the statistics checks, access path analysis, and SQL structure analysis. SQL profile recommendations are not generated.
- For comprehensive, the SQL tuning advisor carries out all the analysis it performs under limited scope plus SQL profiling. You can also specify a time limit for the tuning task, which by default is 30 minutes.

SQL Tuning Advisor Output:

After analyzing the SQL statements, the SQL Tuning Advisor provides advice on optimizing the execution plan, the rationale for the proposed optimization, the estimated performance benefit, and the command to implement the advice. You simply have to choose whether or not to accept the recommendations to optimize the SQL statements.

Running the SQL tuning advisor:

The recommended interface for running the SQL Tuning Advisor is the Oracle Enterprise Manager. Whenever possible, you should run the SQL Tuning Advisor using Oracle Enterprise Manager. If Oracle Enterprise Manager is unavailable, you can run the SQL Tuning Advisor using procedures in the DBMS_SQLTUNE package. To use the APIs, the user must be granted specific privileges.

Running SQL Tuning Advisor using DBMS_SQLTUNE package is a multi-step process:

1. Create a SQL Tuning Set (if tuning multiple SQL statements)

2. Create a SQL tuning task
3. Execute a SQL tuning task
4. Display the results of a SQL tuning task
5. Implement recommendations as appropriate

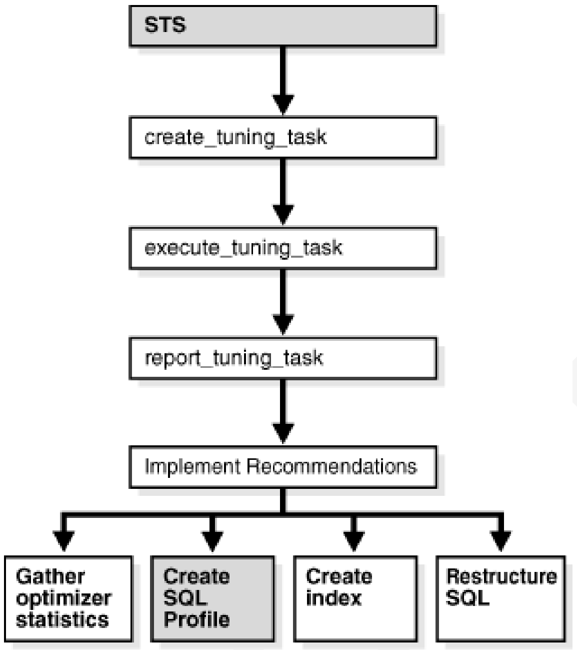


Fig: SQL Tuning Advisor APIs

Creating a SQL tuning task:

You can create tuning tasks from the text of a single SQL statement, a SQL Tuning Set containing multiple statements, a SQL statement selected by SQL identifier from the cursor cache, or a SQL statement selected by SQL identifier from the Automatic Workload Repository.

For example, to use the SQL Tuning Advisor to optimize a specified SQL statement text, you need to create a tuning task with the SQL statement passed as a CLOB argument. For the following PL/SQL code, the user HR has been granted the ADVISOR privilege and the function is run as user HR on the employees table in the HR schema.

```
DECLARE
```

```
 my_task_name VARCHAR2(30);

 my_sqltext   CLOB;

BEGIN

 my_sqltext := 'SELECT /*+ ORDERED */ * '
||

               'FROM employees e, locations l,
departments d ' ||

               'WHERE e.department_id =
d.department_id AND '  ||

                     'l.location_id = d.location_id
AND '      ||

                     'e.employee_id < :bnd';


 my_task_name := DBMS_SQLTUNE.CREATE_TUNING_TASK(

         sql_text    => my_sqltext,

         bind_list   =>
sql_binds(anydata.ConvertNumber(100)),

         user_name   => 'HR',

         scope       => 'COMPREHENSIVE',

         time_limit  => 60,

         task_name   => 'my_sql_tuning_task',

         description => 'Task to tune a query on a
specified employee');

END;

/
```

In this example, 100 is the value for bind variable :bnd passed as function argument of type SQL_BINDS, HR is the user under which the CREATE_TUNING_TASK function analyzes the SQL statement, the scope is set to COMPREHENSIVE which means that the advisor also performs SQL Profiling analysis, and 60 is the maximum time in seconds that the function can run. In addition, values for task name and description are provided.

The CREATE_TUNING_TASK function returns the task name that you have provided or generates a unique task name. You can use the task name to specify this task when using other APIs. To view the task names associated with a specific owner, you can run the following:

```
SELECT task_name FROM DBA_ADVISOR_LOG WHERE owner =
'HR';
```

Configuring a SQL Tuning Task

You can fine tune a SQL tuning task after it has been created by configuring its parameters using the SET_TUNING_TASK_PARAMETER procedure in the DBMS_SQLTUNE package:

```
BEGIN
  DBMS_SQLTUNE.SET_TUNING_TASK_PARAMETER(
    task_name => 'my_sql_tuning_task',
    parameter => 'TIME_LIMIT', value => 300);
END;
/
```

In this example, the maximum time that the SQL tuning task can run is changed to 300 seconds.

Executing a SQL Tuning Task

After you have created a tuning task, you need to execute the task and start the tuning process. For example:

```
BEGIN

  DBMS_SQLTUNE.EXECUTE_TUNING_TASK( task_name =>
'my_sql_tuning_task' );

END;

/
```

Like any other SQL Tuning Advisor task, you can also execute the automatic tuning task SYS_AUTO_SQL_TUNING_TASK using the EXECUTE_TUNING_TASK API. The SQL Tuning Advisor will perform the same analysis and actions as it would when run automatically. You can also pass an execution name to the API to name the new execution.

Automatic Workload Repository (AWR):

The Automatic Workload Repository (AWR) collects, processes, and maintains performance statistics for problem detection and self-tuning purposes. This data is both in memory and stored in the database. The gathered data can be displayed in both reports and views.

The statistics collected and processed by AWR include:

- Wait events to identify performance problems
- Time model statistics indicating the amount of DB Time associated with a process from the V$SESS_TIME AND V$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 access and usage statistics.
- Resource intensive SQL statements.

The STATISTICS_LEVEL initialization parameter must be set to the TYPICAL or ALL to enable the Automatic Workload Repository. If the value is set to BASIC, you can manually capture AWR statistics using procedures in the DBMS_WORKLOAD_REPOSITORY package. However, because setting the STATISTICS_LEVEL parameter to BASIC turns off in-memory collection of many system statistics, such as segments statistics and memory advisor information, manually captured snapshots will not contain these statistics and will be incomplete.

Snapshots:

Snapshots are sets of historical data for specific time periods that are used for performance comparisons by ADDM. By default, AWR automatically generates snapshots of the performance data once every hour and retains the statistics in the workload repository for 7 days. You can also manually create snapshots, but this is usually not necessary. The data in the snapshot interval is then analyzed by the Automatic Database Diagnostic Monitor (ADDM).

AWR compares the difference between snapshots to determine which SQL statements to capture based on the effect on the system load. This reduces the number of SQL statements that need to be captured over time.

Baselines:

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.

```
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.

Generating Automatic Workload Repository Reports

An AWR report shows data captured between two snapshots (or two points in time). The AWR reports are divided into multiple sections. The HTML report includes links that can be used to navigate quickly between sections. The content of the report contains the workload profile of the system for the selected range of snapshots.

The primary interface for generating AWR reports is Oracle Enterprise Manager. Whenever possible, you should generate AWR reports using Oracle Enterprise Manager. If Oracle Enterprise Manager is unavailable, you can generate AWR reports by running SQL scripts:

- The awrrpt.sql SQL script generates an HTML or text report that displays statistics for a range of snapshot Ids.
- The awrrpti.sql SQL script generates an HTML or text report that displays statistics for a range of snapshot Ids on a specified database and instance.
- The awrsqrpt.sql SQL script generates an HTML or text report that displays statistics of a particular SQL statement for a range of snapshot Ids. Run this report to inspect or debug the performance of a SQL statement.
- The awrsqrpi.sql SQL script generates an HTML or text report that displays statistics of a particular SQL statement for a range of snapshot Ids on a specified database and instance. Run this report to inspect or debug the performance of a SQL statement on a specific database and instance.
- The awrddrpt.sql SQL script generates an HTML or text report that compares detailed performance attributes and configuration settings between two selected time periods.
- The awrddrpi.sql SQL script generates an HTML or text report that compares detailed performance attributes and configuration settings between two selected time periods on a specific database and instance.

To run these scripts, you must be granted the DBA role.

Virtual Private Database:

Virtual Private Database (VPD) is a database security feature that is built into an Oracle database server, as opposed to being part of an application that is accessing the data. The user is only allowed to see the data they have been given permission to see. VPD enables you to create security policies to control database access at the row and column level. Essentially, Oracle VPD adds a dynamic WHERE clause to a SQL statement that is issued against the table, view, or synonym to which an Oracle Virtual Private Database security policy was applied.

You can apply Oracle VPD policies to SELECT, INSERT, UPDATE, INDEX, and DELETE statements.

For example:

Suppose a user performs following query:

SELECT * FROM OE.ORDERS;

The Oracle VPD policy dynamically appends the statement with a WHERE clause as:

SELECT * FROM OE.ORDERS WHERE SALES_REP_ID=149;

In this example, the user can view orders by Sales Representative 149;

Oracle Virtual Private Database doesn't support filtering for DDLs such as TRUNCATE OR ALTER TABLE statements.

Benefits of Using Oracle Virtual Private Database:

Oracle VPD policy provides following benefits:

- Basing security policies on database objects rather than applications
- Controlling how Oracle database evaluates policy functions

Basing Security Policies on Database Objects rather than Applications

Attaching Oracle VPD policies to database tables, views, or synonyms rather than applications, provides following benefits:

Sushant Thapa
2070 Batch
+977-98442-06228
Himalaya College of Engineering

- **Security**
  Associating a policy with a database table, view, or synonym can solve a potentially serious application security problem. Suppose a user is authorized to use an application, and then drawing on the privileges associated with that application wrongfully modifies the database by using an ad hoc query tool, such as SQL *Plus. By attaching security policies directly to tables, views or synonyms fine-grained access control ensures that the same security is in force, no matter how a user accesses the data.
- **Simplicity**
  You can add security policy to a table, view or synonym only once, rather than repeatedly adding it to each of your table-based, view-based or synonym-based applications.
- **Flexibility**
  You can have one security policy for SELECT statement, another for INSERT statement, and yet another for UPDATE AND DELETE statement.

Five Oracle VPD Policy Types:

Dynamic Policy:

The Dynamic Policy type runs the policy functions each time a user accesses the VPD protected database objects. If you do not specify a policy type in the DBMS_RLS.ADD_POLICY procedure, then by default, your policy will be dynamic.

This policy type doesn't optimize database performance as the static and context sensitive policy type. However, Oracle recommends that before you set policies as either static or context sensitive, you should test them as DYNAMIC policy types, which runs every time which enables you to observe how the policy function affects each query, because nothing is cached.

Static Policy and Shared Static Policy:

The static policy type enforces the same predicate for all users in the instance. Oracle database stores policy predicates in SGA, so policy functions do not re-run for each query. This results in faster performance. You can enable static policy by setting the policy_type parameter of the DBMA_RLS.ADD_POLICY procedure to either STATIC or SHARED_STATIC, depending on whether or not you want the policy to be shared across multiple objects.

Static policies are ideal for environments where every query requires the same predicate and fast performance is essential, such as hosting environments. For these situations, when the policy function appends the same predicate to every query, rerunning the policy finction each time adds unnecessary overhead to the system.

Context-Sensitive Policy:

In contrast to static properties, context sensitive policies do not always cache the predicate. With context-sensitive policies, the database assumes that the predicate will change after statement parse time. But if there is no change in local application context, Oracle database does not return the policy function within the user session. If there was a change in context, then the database reruns the policy function to ensure that it captures any change to the predicate since the initial parsing.

Context-Sensitive policies are useful when different predicates should be applied depending on which user is executing the query.

Shared Context-Sensitive Policy:

This policy operates in the same was as regular context-sensitive policies, except they can be shared across multiple database objects. For this policy type, all objects can share the policy function from the UGA, where the predicate is cached until the local session context changes.

When using shared context-sensitive policies, ensure that the policy predicate doesn't contain attributes that are specified to a particular database object, such as a column name.

Policy Types	**When the Policy Function Executes**	**Usage Example**	**Shared Across Multiple Objects?**
DYNAMIC	Policy function re-executes every time a policy-protected database object is accessed.	Applications where policy predicates must be generated for each query, such as time-dependent policies where users are denied access to database objects at certain times during the day	No
STATIC	Once, then the predicate is cached in the SGA.[Foot 1]	View replacement	No
SHARED_STATIC	Same as STATIC	Hosting environments, such as data warehouses where the same predicate must be applied to multiple database objects	Yes
CONTEXT_SENSITIVE	At statement parse time At statement execution time when the local application context changed since the last use of the	Three-tier, session pooling applications where policies enforce two or more predicates for different users or groups	No

	cursor		
SHARED_CONTEXT_SENSITIVE	First time the object is reference in a database session. Predicates are cached in the private session memory UGA so policy functions can be shared among objects.	Same as CONTEXT_SENSITIVE, but multiple objects can share the policy function from the session UGA	Yes

Foot 1: Each execution of the same cursor could produce a different row set for the same predicate because the predicate may filter the data differently based on attributes such as `SYS_CONTEXT` or `SYSDATE`.

Controlling the display of Column data with policies:

You can create policies that enforce row-level security when a security-relevant column is referenced in a query.

- Adding Policies for Column-Level Oracle Virtual Private Database
- Displaying Only the Column Rows Relevant to the Query
- Using Column Masking to Display Sensitive Columns as NULL Values

Adding Policies for Column-Level Oracle Virtual Private Database

Column-level policies enforce row-level security when a query references a security-relevant column. You can apply a column-level Oracle Virtual Private Database policy to tables and views, but not to synonyms.

To apply the policy to a column, specify the security-relevant column by using the sec_relevant_cols parameter of the DBMS_RLS.ADD_POLICYprocedure. This parameter applies the security policy whenever the column is referenced, explicitly or implicitly, in a query.

For example, users who are not in a Human Resources department typically are allowed to view only their own Social Security numbers. A sales clerk initiates the following query:

```
SELECT fname, lname, ssn FROM emp;
```

The function implementing the security policy returns the predicate ssn='my_ssn'. Oracle Database rewrites the query and executes the following:

```
SELECT fname, lname, ssn FROM emp

 WHERE ssn = 'my_ssn';
```

Example 1 shows a Oracle Virtual Private Database policy in which sales department users cannot see the salaries of people outside the department (department number 30) of the sales department users. The relevant columns for this policy are sal and comm. First, the Oracle Virtual Private Database policy function is created, and then it is added by using the DBMS_RLS PL/SQL package.

Example 1: Creating a Column-Level Oracle Virtual Private Database Policy

```
CREATE OR REPLACE FUNCTION hide_sal_comm (
 v_schema IN VARCHAR2,
 v_objname IN VARCHAR2)

RETURN VARCHAR2 AS
con VARCHAR2 (200);

BEGIN
 con := 'deptno=30';
 RETURN (con);
END hide_sal_comm;
/
```

Then you configure the policy with the DBMS_RLS.ADD_POLICY procedure as follows:

```
BEGIN
 DBMS_RLS.ADD_POLICY (
  object_schema      => 'scott',
  object_name        => 'emp',
  policy_name        => 'hide_sal_policy',
  policy_function    => 'hide_sal_comm',
  sec_relevant_cols => 'sal,comm');
END;
/
```

Displaying Only the Column Rows Relevant to the Query

The default behavior for column-level Oracle Virtual Private Database is to restrict the number of rows returned for a query that references columns containing sensitive information. You specify these security-relevant columns by using the sec_relevant_columns parameter of the DBMS_RLS.ADD_POLICY procedure as shown in the Example 1.

For example, consider sales department users with the SELECT privilege on the emp table, which is protected with the column-level Oracle Virtual Private Database policy created in Example 1. The user (for example, user SCOTT) runs the following query:

```
SELECT ENAME, d.dname, JOB, SAL, COMM
 FROM emp e, dept d
 WHERE d.deptno = e.deptno;
```

The database returns the following rows:

```
ENAME      DNAME          JOB              SAL
COMM
---------- -------------- --------- ---------- ----
------
ALLEN      SALES          SALESMAN        1600
300
WARD       SALES          SALESMAN        1250
500
```

Sushant Thapa
2070 Batch
+977-98442-06228
Himalaya College of Engineering

```


2 rows selected.
```

The only rows that are displayed are those that the user has privileges to access all columns in the row.

Using Column Masking to Display Sensitive Columns as NULL Values

If a query references a sensitive column, then the default action of column-level Oracle Virtual Private Database restricts the number of rows returned.

In contrast to the default action of column-level Oracle Virtual Private Database, column-masking displays all rows, but returns sensitive column values as NULL. To include column-masking in your policy, set the sec_relevant_cols_opt parameter of the DBMS_RLS.ADD_POLICY procedure to dbms_rls.ALL_ROWS.

Example below shows column-level Oracle Virtual Private Database column-masking. It uses the same VPD policy as Example 1, but with sec_relevant_cols_opt specified as dbms_rls.ALL_ROWS.

Example: Adding a Column Masking to an Oracle Virtual Private Database Policy

```
BEGIN

 DBMS_RLS.ADD_POLICY(

   object_schema          => 'scott',

   object_name            => 'emp',

   policy_name            => 'hide_sal_policy',

   policy_function        => 'hide_sal_comm',
```

Sushant Thapa
2070 Batch
+977-98442-06228
Himalaya College of Engineering

```
   sec_relevant_cols      =>' sal,comm',
   sec_relevant_cols_opt => dbms_rls.ALL_ROWS);
END;
/
```

Assume that a sales department user with SELECT privilege on the emp table (such as user SCOTT) runs the following query:

```
SELECT ENAME, d.dname, job, sal, comm
 FROM emp e, dept d
 WHERE d.deptno = e.deptno;
```

The database returns all rows specified in the query, but with certain values masked because of the Oracle Virtual Private Database policy:

```
ENAME      DNAME          JOB               SAL
COMM

---------- -------------- --------- ---------- ----
------

CLARK      ACCOUNTING     MANAGER
KING       ACCOUNTING     PRESIDENT
SCOTT      RESEARCH       ANALYST
WARD       SALES          SALESMAN         1250
500
JAMES      SALES          CLERK             950
MARTIN     SALES          SALESMAN         1250
1400
```

Sushant Thapa
2070 Batch
+977-98442-06228
Himalaya College of Engineering

```
6 rows selected.
```

The column-masking returned all rows requested by the sales user query, but made the sal and comm columns NULL for employees outside the sales department.

The following considerations apply to column-masking:

- Column-masking applies only to SELECT statements.
- Column-masking conditions generated by the policy function must be simple Boolean expressions, unlike regular Oracle Virtual Private Database predicates.
- For applications that perform calculations, or do not expect NULL values, use standard column-level Oracle Virtual Private Database, specifying sec_relevant_cols rather than the sec_relevant_cols_opt column-masking option.
- Column-masking used with UPDATE AS SELECT updates only the columns that users are allowed to see.
- For some queries, column-masking may prevent some rows from displaying. For example:

  ```
  SELECT * FROM emp

  WHERE sal = 10;
  ```

Because the column-masking option was set, this query may not return rows if the salary column returns a NULL value.

Model Question (Database Administration)
Tribhuvan University
Institute of Science and Technology

Bachelor Level/Fourth Year/Seventh Semester/Science Pass Marks: 24
Database Administration (CSC-406) Full Marks: 60
Time: 3 Hours.

Candidates are required to give their answers in their own words as far as practicable. The figures in the margin indicate full marks.

Group A
Long Answer Questions (Attempt any TWO questions) [2×10=20]

1. What do you mean by database management system? Mention the clear architecture of database management system including its different files.
2. You are working as DBA in Nepal Clearing House Limited which handle 24/7 online system. You have to design security and backup procedure for the organization.
3. What are the various kinds of performance report available in the oracle database management system mention the name and describe them one by one.

Group B
Short Answer Questions (Attempt any Eight questions) [8×5=40]
4. What is DML? Explain all the SML command with suitable example.
5. What is a view? Write an importance of view.
6. Explain the role of database administrator?
7. What is ASM? Mention the capabilities of ASM.
8. Write the command to add new redo group and drop group in the database.
9. Write DDL command for following.
 a. Change table product to product_org
 b. Add new check constraints in the status field of product_org table value of status field should always (00,01 and 09)
 c. Create backup table of sales
 d. Disable check constraint in the table.
10. What is AWR report? Write all the component of AWR report.
11. What do you mean by parallel instance recovery?
12. Write a trigger to create log of every update and delete statement of product table.
13. Write short note of followings:
a. Step to create backup device
b. Internet database connector

Sushant Thapa
2070 Batch
+977-98442-06228
Himalaya College of Engineering

Tribhuvan University
Institute of Science and Technology
2069

Bachelor Level/ Fourth Year/ Seventh Semester/ Science Full Marks: 60
Computer Science and Information Technology (CSC. 406) Pass Marks: 24
(Database Administration) Time: 3 hours.
Candidates are required to give their answers in their own words as far as practicable. Thefigures in the margin indicate full marks. (NEW COURSE)

Group A

Attempt All Questions (2*10=20)
1. Explain the database management system and also describe the database management system
architecture.
2. How can you design the plan of backup, recovery and security of an organization? Explain.
3. Explain the control and Redo log files with applications.

Group B

Attempt any eight questions. (8*5=40)
4. What do you mean by SQL and PLUS? Explain.
5. Explain the advanced stored procedures. Explain PL/SQL and its structure
6. Explain the testing the backup and recovery plan.
7. What do you mean by database corruption? How can you maintain the database corruption?
8. Describe the SQL Tuning Advisor with example.
9. What do you mean by virtual private database?
10. Explain the parallel instance recovery with example.
11. What do you mean by automatic storage management? Explain.
12. Explain the tuning methodology with example.
13. Write short notes on:
a. AADM (Automatic Diagnostic Monitor)
b. RMAN

Tribhuvan University
Institute of Science and Technology
2070

Bachelor Level/ Fourth Year/ Seventh Semester/ Science — Full Marks: 60
Computer Science and Information Technology (CSc. 406) — Pass Marks: 24
(Database Administration) — Time: 3 hours.

Candidates are required to give their answers in their own words as far as practicable. The figures in the margin indicate full marks.

(NEW COURSE)

Group A

Attempt All Questions (2*10=20)

1. Explain the database management system architecture and also explain the DBA roles and responsibilities.
2. What do you mean by Redo log files? Explain the maintaining and monitoring redo log files.
3. What is tuning methodology? Explain the Automatic database diagnostic monitor with its applications.

Group B

Attempt any Eight questions. (8*5=40)

4. Explain the database security and auditing.
5. How can you test the backup and recovery plan?
6. How can you manage the automatic storage?
7. Differentiate between database administrator and data administrator.
8. Explain the ASM and its capabilities.
9. Define the AWR. What are the components of AWR report?
10. What do you mean by parallel instance recovery?
11. What do you mean by database corruption? How can you recover in this case?
12. Explain the SQL tuning advisor.
13. Write short notes on:

a. Using ISQL Plus
b. RMAN

Tribhuvan University

Institute of Science and Technology

2071

??

Bachelor Level/ Fourth Year/ Seventh Semester/ Science Full Marks: 60

Computer Science and Information Technology (CSc. 406) Pass Marks: 24

(Database Administration) Time: 3 hours.

Candidates are required to give their answers in their own words as far as practicable. The figures in the margin indicate full marks.

(NEW COURSE)

Group A

Attempt any Two Questions (2*10=20)

1. Explain any detail about the DBMS architecture
2. What do you mean by database performance Tuning? Explain the role of ADDM and SQL Tuning Advisors in performance testing of oracle database.
3. As a DBA explain how you can manage user privileges, roles and profiles? Consider you got
a call from your backup DBA while you are on leave. He has corrupted all of the data files while playing with the ALTER DATABASE CONTROLFILE command. What do you do? Explain.

Group B

Attempt any Eight questions. (8*5=40)

4. Differentiate between logical and physical data independence.
5. What are the authentication methods for DBA? Explain.
6. What is redo log buffer and how do you monitor a waiting session in the
redo log buffer?
7. Differentiate between hot and cold backup. What is involved with
executing a hot backup?
8. Consider a situation that an alteration is made to the sp file which
results in the instance being unable to start. As DBA how can you fix this
problem?
9. What do you mean by a view? Explain its importance.
10. What do you mean by parallel instance recovery?
11. What is oracle virtual private database? Explain the benefit associated
with it.
12. Write the DBA queries for the following actions:
a. To display the size of your database

b. Lock or unlock a user
c. Show all connected users with their status
d. Show the current SQLs
e. Show what roles are granted to a user.

Tribhuvan University
Institute of Science and Technology
2073

Bachelor Level/ Fourth Year/ Seventh Semester/ Science Full Marks: 60
Computer Science and Information Technology (CSc. 406) Pass Marks: 24
(Database Administration) Time: 3 hours.
Candidates are required to give their answers in their own words as far as practicable. The figures in the margin indicate full marks.

Section A (2*10=20)

1. What do you mean by Database and database management system? Explain the SQL Plus overview.
2. Explain the managing maintaining and monitoring redo log files with appropriate example.
3. Explain the backup and recovery plan and strategy with practical example.

Section B

Attempt any eight questions. (8*5=40)
4.How can you manage the control files? Explain.
5.Explain the database security and auditing.
6.Explain the testing the backup and recovery plan.
7.What do you mean by backup and recovery strategy?
8.Explain the RMAN with example.
9.What do you mean by tuning methodology?
10.Explain the virtual private database with example.
11.Explain the parallel instance recovery with practical example.
12.What are the major difference between automatic database management and automatic storage management?
13.Write short notes on:
a. DBA roles
b. Column masking

Sushant Thapa
2070 Batch
+977-98442-06228
Himalaya College of Engineering