SMT L.P SAVANI SATUABABA BCA COLLEGE

B.C.A. SEM. - VI

Subject: RDBMS Using Oracle -II

SMT L.P SAVANI SATUABABA BCA COLLEGE

B.C.A. SEM-VI

Subject: RDBMS using Oracle-II

UNIT1

Basic PL/SQL Programming

- PL/SQL Block Structure.
- Control Structure.
- Implicit Cursor Programming.
- Explicit Cursor Programming.
- Parameterize Cursor and Cursor For Loop.

- **PL/SQL Block Structure**
- **Introduction of PL/SQL:**
- PL/SQL stands for Procedural Language / Structured Query Language.
- PL/SQL is an extension of SQL.
- It is a procedural language.
- The PL/SQL programming language was developed by Oracle Corporation in the late 1980s.
- PL/SQL is a combination of SQL along with the procedural features of programming languages.
- PL/SQL is specially designed for database oriented activities.
- PL/SQL is a completely portable, high-performance transaction-processing language. PL/SQL provides a built-in, interpreted and OS independent programming environment. PL/SQL can also directly be called from the command-line SQL*Plus interface.
- We can use PL/SQL to implement our business application by creating stored function, cursor and triggers.
- PL/SQL code is grouped into structures called as blocks.
- With the PL/SQL language, we can create schema objects, including following…
  - **Stored procedures & functions:**
  - A stored procedure is a PL/SQL program that can be enabled by an application, a trigger, or an Oracle tool.
  - Difference between procedure & function is that function returns a result but procedure cant.
  - **Packages:**
  - A package is a file that groups functions, cursors, stored procedures and variables in one place.
  - **Triggers:**
  - A trigger is a PL/SQL program that is stored in the database and executed immediately before or after the INSERT, UPDATE and DELETE commands.
  - **Cursors:**
  - Oracle uses workspaces to execute the SQL commands.
  - Through PL/SQL cursors, it is possible to name the workspace and access its information.
- **Advantages of PL/SQL:**
- **PL/SQL supports both static and dynamic SQL:**
- Static SQL supports DML operations and transaction control from PL/SQL block.
- In Dynamic SQL support DDL operations and transaction control from PL/SQL block.
- **Reduce network traffic:**
- PL/SQL allows sending an entire block of statements to the database at one time.
- This reduces network traffic and provides high performance for the applications.
- **Procedural language support:**
- PL/SQL is a development tools not only for data manipulation futures but also provide the conditional checking, looping or branching operations same as like other programming language.
- **Error handling:**
- PL/SQL is dealing with error handling, its permits the smart way handling the errors and giving user friendly error messages, when the errors are occur.
- **Declare variable:**
- PL/SQL gives you control to declare variables and access them within the block.
- The declared variables can be used at the time of query processing.

- **Intermediate Calculation:**
- Calculations in PL/SQL done quickly and efficiently without using Oracle engines.
- This improves the performance of transaction.
- **Portable application:**
- Applications are written in PL/SQL are portable in any Operating system.
- PL/SQL applications are independence program to run any computer.

- **Features of PL/SQL:**

(1)Block structure:
- PL/SQL is a block-structured language. Each program written in PL/SQL is written as a block.
- Blocks can also be nested. Each block is meant for a particular task.

(2)Variables and constants:
- PL/SQL allows you to declare variables and constants.
- Variables are used to store valuestemporarily.
- Variables and constants can be used in SQL and PL/SQL procedural statements just like anexpression.

(3)Control structures:
- PL/SQL allows control structures like IF statement, FOR loop, WHILE loop to be used in the block.
- Control structures are most important features in PL/SQL.

(4)Exception handling:
- An errors is to be detected and it is solve using exception.
- Whenever there is a predefined error PL/SQL raises an exception automatically.
- These exceptions can be handled to recover from errors.

(5)Modularity:
- PL/SQL allows process to be divided into different modules.
- Subprograms called as procedures and functions can be defined and invoked using the name.
- These subprograms can also take parameters.

(6)Cursors:
- A cursor is a private SQL area used to execute SQL statements and store processing information.
- PL/SQL implicitly uses cursors for all DML commands and SELECT command that returns only one row. And it also allows you to define explicit cursor to deal with multiple row queries.

(7)Built-in functions:
- Most of the SQL functions that we have seen so far in SQL are available in PL/SQL.
- These functionscan be used to manipulate variables of PL/SQL.

- **A Simple PL/SQL Block structure:**
- Each PL/SQL program consists of SQL and PL/SQL statements which makes a PL/SQL block. A PL/SQL Block consists of three sections:

  - **The Declaration section (optional).**
  - **The Execution section (mandatory).**
  - **The Exception (or Error) Handling section (optional).**

- **Declaration Section:**
- The Declaration section of a PL/SQL Block starts with the reserved keyword **DECLARE**.
- This section is optional and is used to declare any placeholders like variables, constants, records and cursors, which are used to manipulate data in the execution section.
- Placeholders may be any of Variables, Constants and Records, which stores data temporarily.
- Cursors are also declared in this section.
- **Execution Section:**
- The Execution section of a PL/SQL Block starts with the reserved keyword BEGIN and ends with END.
- This is a mandatory section and is the section where the program logic is written to perform any task.
- The programmatic constructs like loops, conditional statement and SQL statements are written into execution section.
- **Exception Section:**
- The Exception section of a PL/SQL Block starts with the reserved keyword EXCEPTION.
- This section is optional.
- Any errors in the program can be handled in this section, so that the PL/SQL Blocks terminates gracefully.
- If the PL/SQL Block contains exceptions that cannot be handled, the Block terminates abruptly with errors.
  - *Every statement in the above three sections must end with a semicolon;*
  - *PL/SQL blocks can be nested within other PL/SQL blocks.*
  - *Comments can be used to document code*.

- **Simple PL/SQL Block**

Structure:

DECLARE section (Optional)

Declaration of memory variables, Variable, Cursors, User-Defined Exception, constants,

BEGIN section (Mandatory)

SQL executable statements
PL/SQL executablestatements.

EXCEPTION section (Optional)

SQL or PL/SQL code to handle errors that may arise during the execution of the code block between BEGIN and EXCEPTION section.

END; (Mandatory)

Each & every PL/SQL block terminates with END section.

SAVANI

Command to Create PL/SQL block.

- **Syntax:**
  Edit program_name.sql
- **Example:**
  Edit first.sql

Command to execute PL/SQL block

- **Syntax:** Startprogram_name.sql
  OR
  @program_name.sql
- **Example:** Startfirst.sql
  OR
  @ first.sql

Example: PL/SQL block Program

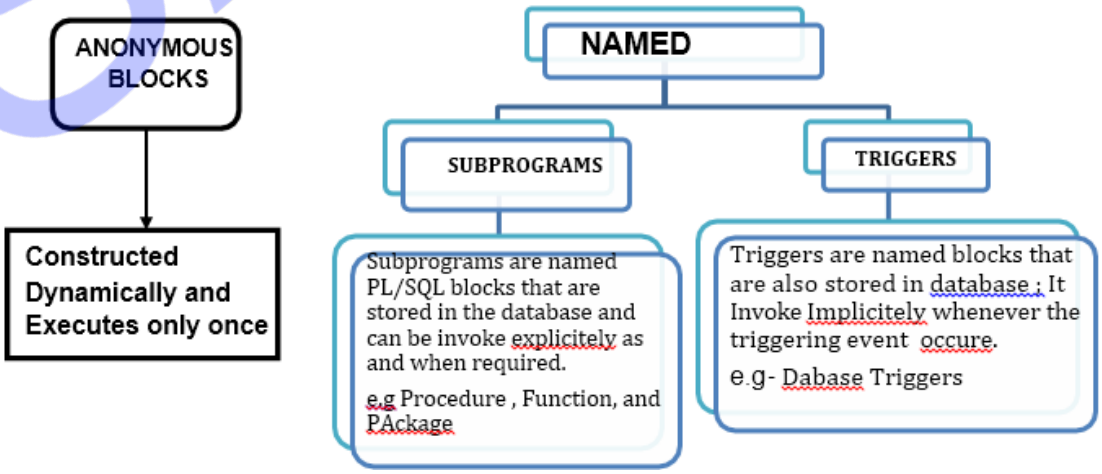
```
Edit
   first.sqlSETserverout
   put on BEGIN
            dbms_output.put_line('Welcome to Oracle');
   END;
```

Output:

Welcome to Oracle

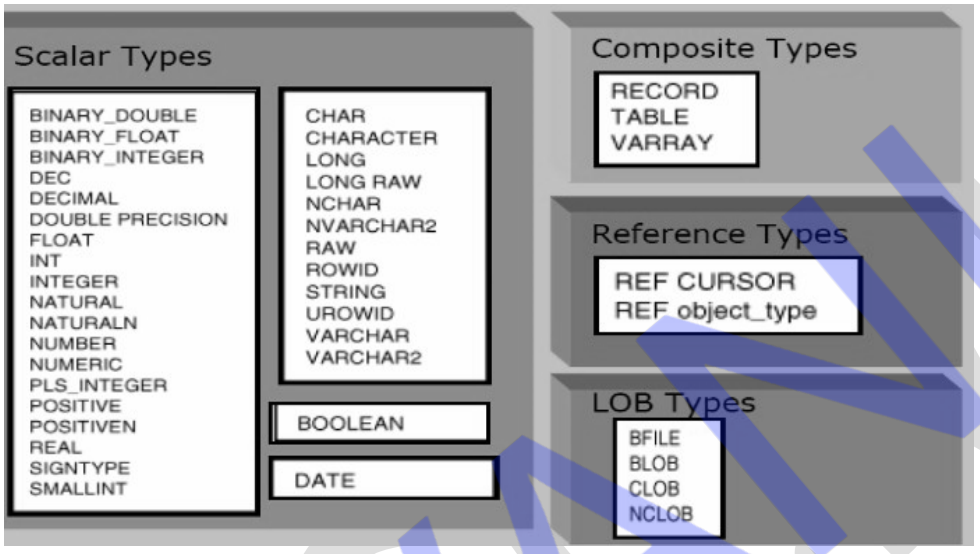
- **SET SERVEROUTPUT ON**
- By default it is off.
- To display any message on the oracle screen we have to first set **serveroutput**on.
- **dbms_output.put_line()**
- This function is used display the message as well as value of variable.
- dbms_output is a package that includes a number of procedures and functions that collect information in a buffer so it can be retrieved later.
- put_line puts a piece of information in the package buffer followed by an end-of-line marker.

- **What are the various types of PL/SQL blocks?**
- PL/SQL is a block structured language.
- It is composed of one or more blocks.



- **PL/SQL :-Data Types and Subtypes**

  "Data types means the type of value a variable can stored."
- **Number** – Store numeric values on which arithmetic operations are performed.
- **Char**- Store Alphanumeric values that represent single characters or strings of characters.
- **Date**- It is used to store Date and time type of value.
- **Boolean** - Logical values on which logical operations are performed (True/False).



- **%TYPE:**
- %TYPE is used for retrieving column value from table.
- PL/SQL can use the %TYPE attribute to declare variables based on definitions of columns in a table.
- If, a column's attribute change, the variable's attributes will change as well.
- %TYPE declares a variable or constant to have the same data type as that of a previously defined variable or of a column in a table or in a view.
- **Example: STUDENT.S_ID % type;**
- Here, STUDENT is table name and S_ID is field name (Column name).
- When referencing a table, a user may name the table and the column, or name the owner, the table and the column.
- **Declaring Variables:**
- Variables can have any data type, such as CHAR, DATE, NUMBER, BOOLEAN, etc.
- For example, assume that you want to declare variables for part Employee no, such as empno to hold 6-digit numbers and gender to hold the Boolean value TRUE or FALSE.
- Note that there is a semi-colon (;) at the end of each line in the declaration section. You can assign a value to a variable when it is declared.

```
DECLARE
        empnoNUMBER(6);
        empname VARCHAR2(20);
        gender BOOLEAN;
        hourly_salary NUMBER := 22.50;
        bonus  NUMBER := 150;
```

```
DECLARE
  num1 INTEGER;
  num2 REAL;
  num3 DOUBLEPRECISION;
  BEGIN
      null;
  END;
```

Control Structure:

- Following are three categories of control structure in PL/SQL.
  - **Conditional Control (If statement).**
  - **Iterative Control (Looping Structure).**
- **Conditional Control (If statement).**
- Conditional control is used for decision- making.
- Decision-making structures require that the programmer specify one or more conditions to be evaluated by the program.
- Statement are executed according to condition. There are following type of conditional statement:
  - **IF THEN Statement**
  - **IF THEN ELSE Statement**
  - **IF THEN ELSIF Statement (Else-if leader)**
  - **Nested IF THEN ELSE Statement**
  - **CASE Statement**
- **IF THEN Statement:**
- When an IF-THEN statement is executed, a condition is evaluated to either TRUE or FALSE.
- If the condition is TRUE, control is passed to the first executable statement of the IF-THEN construct.
- If the condition is FALSE, control is passed to the first executable statement after the END IF statement.

:: Syntax ::

```
IF (condition) THEN
   statement
END IF;
```

:: Example ::

```
DECLARE
 no INTEGER(2) := 10;
 BEGIN
 IF (no = 10) THEN
  DBMS_OUTPUT.PUT_LINE('Condition true');
 END IF;
END;
```

:: Output ::

```
Condition true
```

- **IF THEN ELSE Statement:**
- IF condition is TRUE then execute statement-1 part and cursor move next to END IF.
- IF condition is FALSE then execute statement-2 part and cursor move next to END IF.

:: Syntax ::

```
IF (condition) THEN
   Statement-1
ELSE
  Statement-2
END IF;
```

:: Example ::

```
DECLARE
  no INTEGER(2) := 10;
BEGIN
  IF (no = 10) THEN
      DBMS_OUTPUT.PUT_LINE('condition TRUE');
  ELSE
      DBMS_OUTPUT.PUT_LINE('condition FALSE');
  END IF;
END;
```

3) IF THEN ELSIF Statement:

- If you want to execute only one statement block out of multiple condition then it is used.
- Here we give another condition in ELSE part.
- We used ELSIF for another condition.
- The words CONDITION 1 through CONDITION N are a sequence of the conditions that evaluate to TRUE or FALSE. These conditions are mutually exclusive.
- In other words, if CONDITION 1 evaluates to TRUE, STATEMENT 1 is executed, and control is passed to the first executable statement after the reserved phrase END IF.
- The rest of the ELSIF construct is ignored. When CONDITION 1 evaluates to FALSE, control is passed to the ELSIF part and CONDITION 2 is evaluated, and so on.

:: Syntax ::

```
IF (condition-1) THEN
   Statement-1
ELSIF (Condition-2) THEN
  Statement-2
ELSIF (Conition-3) THEN
  Statement-3
ELSE
  Statement-N
END IF;
```

:: Output ::

```
No is 10
```

:: Example ::

```
DECLARE
  no INTEGER(2) := 10;
BEGIN
  IF (no = 5) THEN
        DBMS_OUTPUT.PUT_LINE ('No is 5');
  ELSIF (no=10)THEN
        DBMS_OUTPUT.PUT_LINE ('No is 10');
  ELSIF (no=15)THEN
        DBMS_OUTPUT.PUT_LINE ('No is 15');
   ELSE
         DBMS_OUTPUT.PUT_LINE ('No is'||no);
  END IF;
END;
```

4) Nested IF THEN ELSE Statement:

- If we want to use IF THEN ELSE inside another IF THEN ELSE statement then nested IF is used.
- We write IF condition into another IF statement block.
- According to condition particular block will be executed.

:: Syntax ::

```
IF (condition-1) THEN
        IF (condition-2)
                THEN
                Statement-1
        ELSE
                Statement-2
        END IF;
ELSE
 Statement-3
END IF;
```

:: Output ::

```
No is 10
```

:: Example ::

```
DECLARE
  no INTEGER(2) := 10;
   BEGIN
  IF (no > 5) THEN
        IF (no=10) THEN
           DBMS_OUTPUT.PUT_LINE ('No is 10');
        ELSE
           DBMS_OUTPUT.PUT_LINE ('No is not 10');
        END IF;
ELSE
    DBMS_OUTPUT.PUT_LINE ('No is'||no);
  END IF;
END;
```

- **CASE Statement:**
- If we require multiple IF then we used CASE statement.
- CASE statement selects one sequence of statement to execute.
- However, to select the sequence, the CASE statement uses selector rather than multiple Boolean expression.
- A selector is an expression, the value of which is used to select one of several alternatives.

:: Syntax ::

```
CASE selector
   WHEN value-1 THEN
      Statement-1;
  WHEN value-2 THEN
      Statement-2;
   ELSE
      Statement-3;
END CASE;
```

:: Output ::

```
value 3
```

:: Example ::

```
DECLARE
  no INTEGER := 3; BEGIN
   CASE no
     WHEN 1 THEN
        DBMS_OUTPUT.PUT_LINE ('value 1');
     WHEN 2 THEN
        DBMS_OUTPUT.PUT_LINE ('value 2');
     WHEN 3 THEN
        DBMS_OUTPUT.PUT_LINE ('value 3');
     ELSE
        DBMS_OUTPUT.PUT_LINE ('no matching CASE found');
   END CASE;
END;
```

- **<u>Iterative Control (Looping Structure).</u>**
- Looping structure is used to reduce number of common statements in a program for writing repeatedly.
- It means sometimes we want to execute certain statements to a fix number of times at that time we write those statements repeatedly then the code becomes too long which is not a good programming practice.
- Looping is a process that executes some of the common statements repeatedly up to finite number of times (Statement write only one time).
- In looping, set of statements are executed until some conditions for termination of the loop is satisfied.
- A program loop therefore consists of two parts one known as body of the loop and other is known
- as the test condition statement.
- The test condition statement checks certain conditions and then directs the repeated execution of the statements contained in the body of the loop.
- Following looping structure available in PL/SQL:
    - Simple LOOP
    - WHILE LOOP
    - FOR LOOP

(1) Simple LOOP:

- In this loop the sequence of statements encloses in between LOOP and END LOOP statement.
- When each iteration, the sequence of statements is executed and then control resumes at the top of the loop.

- Simple loop is also known as exit controlled loop.
- In this loop condition is check at the end of loop.
- Here, sequence of statements may be single statement or multiple statement.
- An EXIT statement or EXIT WHEN statement is required to break the loop.

:: Syntax ::

```
LOOP
        Statement-1
        Statement-2
   EXIT WHEN Condition;
END LOOP;
```

:: Output ::

1 2 3 4 5

:: Example ::

```
DECLARE
        i number (3);
BEGIN
        i:=1;
        loop
                dbms_output.put(i);
                i:=i+1;
            exit when i>5;
        end loop;

END;
```

(2) WHILE LOOP

- The reserved word WHILE marks the beginning of a loop construct.
- In this loop the condition is check first.
- If condition is true then only statement will be executed.
- Whenever condition is false then cursor move to the next executable statement after END LOOP.
- Statements 1 through N are a sequence of statements that is executed repeatedly.
- The END LOOP is a used to show the end of the loop.

:: Syntax ::

```
WHILE <condition>
   LOOP
        Statement-1
        Statement-2
        ……..……..
        Statement- N
   END LOOP;
```

:: Output :: 1 2 3 4 5

:: Example ::

```
DECLARE
        i number (3);
BEGIN
        i:=1;
        WHILE (i<=5)
        loop
                dbms_output.put(i);
                i:=i+1;
        end loop;
END;
```

(3) FOR LOOP:

- For loop is considered as Counter Loop.
- For loop is used when series of statement needs to be executed for specific number of times.
- For loop is an entry control loop.

SAVANI

- For loop is implemented on Number Datatype only.
- Scope of Variable is Private and it is within Loop.
- First we need to initialize, then need to perform increment / decrement.
- REVERSE Key word is used to execute Loop in Reverse order (Ex:- Print 10 to 1)
- For loop is execute as per the range you defined in the loop.
    - Variable in for loop need not be declared.
    - Increment value cannot be specified.
    - For loop variable is always incremented by 1.

:: Syntax ::

```
For <variable-name> in [REVERSE] <start-value>..<Stop-value>

Loop
        Statements;
End loop;
```

:: Example ::

```
DECLARE
    I number (3);
BEGIN
    For I in 1..5
    loop
        dbms_output.put(i);
    end
loop;
END;
```

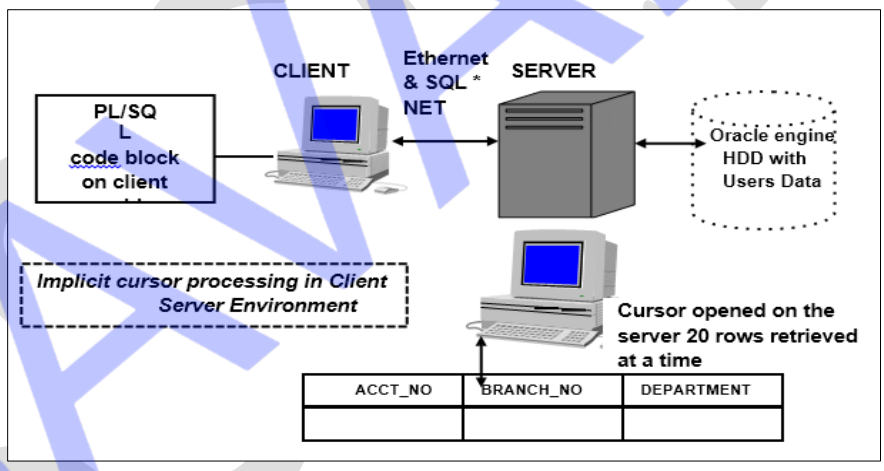
- **Implicit Cursor Programming**
    - **What is Cursor?**
- A cursor is a temporary work area created in the system memory when a SQL statement is executed.
- A cursor contains information of a select statement and the rows of data accessed by it.
- This temporary work area is used to store the data retrieved from the database, and manipulate this data.
- A cursor can hold multiple rows, but can process only one row at a time. The set of rows the cursor holds is called the **active data set**.
- The size of cursor is the sizes require for holding the number of Active data set. There are two types of cursors:
    - **Implicit cursor**
    - **Explicit cursor**

- **Implicit Cursor:**
- Implicit cursor is created by default when DML statements like, INSERT, UPDATE, and DELETE statements are executed.
- Implicit cursor is automatically open by Oracle Engine. There for we not require to open implicit cursor.
- They are also created when a SELECT statement that returns just one row is executed.
- Oracle provides few attributes known as implicit cursor attributes to check the status of DML operations.
- The cursor attributes available are %FOUND, %NOTFOUND, %ROWCOUNT, and %ISOPEN.
- For example, when you execute INSERT, UPDATE, or DELETE statements the cursor attributes tell us whether any rows are affected and how many have been affected.

- Implicit cursor attributes can be used to find out whether any row has been returned by the SELECT statement. PL/SQL returns an error when no data is selected.
- **Implicit Cursor Attributes:**

Attribute Name	Description
%ISOPEN	Returns TRUE if cursor is open, otherwise FALSE
%FOUND	Returns TRUE if record was fetched successfully, otherwise FALSE
%NOTFOUND	Returns TRUE if record was Not Fetched successfully, otherwise FALSE.
%ROWCOUNT	Returns number of records processed from the cursor.

- Since it is opened and managed by the Oracle engine internally, it fulfills the following tasks…
  - Reserving an area in memory.
  - Manipulating this area with appropriate data.
  - Processing the data in the memory area.
  - Releasing the memory area when the processing is complete etc.
  - Result data then passed to the client machine via the network.
- A cursor then opened in memory on the client machine to hold the rows returned by the oracle engine.
- Implicit cursor attributes can be used to access information about the status of the last insert, update, delete or single-row select statements.

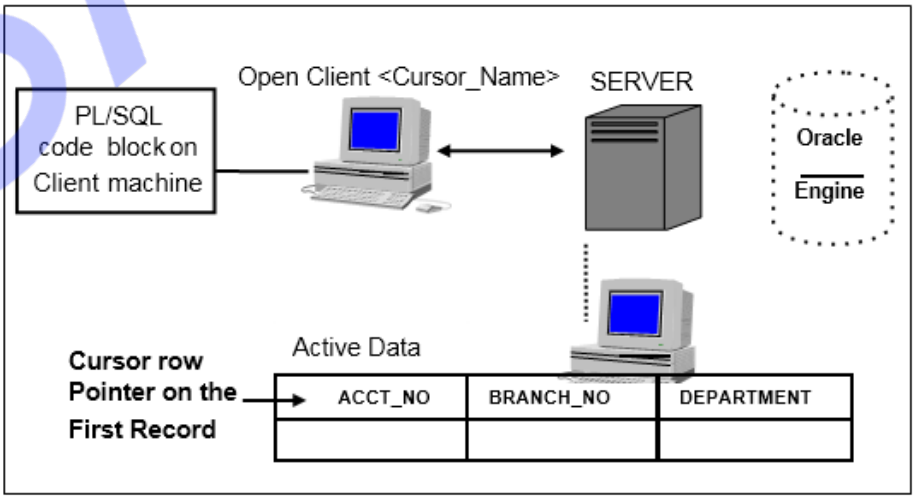


:: Example::

```
setserveroutput on
set verify off
set feedback off
begin
        update emp set sal=sal+500 where job='MANAGER';
if (sql%found) then
     dbms_output.put_line('Total '||sql%rowcount||' Records are updated');
else
     dbms_output.put_line('No records are Updated');
end if;
end;
```

- **Explicit Cursor Programming**
- Explicit cursor is needed when select statement fetch multiple records from tables.
- It has to be declared manually in the declaration section of the PL/SQL Block.
- Explicit cursor stores multiple records but only one record can be processed at a time, which is called as current row.
- When you fetch a row, the current row position moves to next row.
- Explicit cursor follows following steps…..
    1. **Declare a cursor**
    2. **Open the cursor**
    3. **Fetch data from the cursor one row at a time into memory variables.**
    4. **Process the data held in the memory variable as required using a loop.**
    5. **Exit from the loop after processing is complete.**
    6. **Close the cursor.**
- **1) Cursor Declaration (Declare a Cursor):**
- Cursor is defined in the declarative part of a PL/SQL block.
- In this section we give the name of cursor.
- Declaration is only virtual it not occupy any memory.
- After declaration three commands used to control the cursor are open, fetch and close.
- **Syntax**: cursor <cursor_name> is <select statement>;
- **Example**: cursor emp_update is select empno, deptno, sal from emp;

- **2) Opening a cursor:**
- It first executes the query and creates the active set that contains all rows which meets the query search criteria.
- When a cursor is opened, the first row becomes the current row.
- Open statement retrieves records from a database table and places the records in the cursor.
- A cursor is opened in the Server's memory.
- **Syntax**: open <cursor_name>;
- **Example**: open emp_update;



- **3) Fetching a record from the Cursor:**
- Fetch statement retrieves the rows from the active set opened in the server into memory variables declared in the PL/SQL code block on the client one row at a time.
- Memory variables are opened at client machine.
- Each time FETCH is executed, the cursor pointer is advanced to the next row in Active Data Set.
- Standard loop structure (loop…End loop) is used to fetch records from the cursor into memory variables one row at a time.
- **Syntax**: FETCH <cursor_name> into <variable1>,<variable2>,….<variableN>;
- **Example**: fetch emp_update into eno, dno, salary;

- **4) Closing a Cursor:**
- Close statement disables the cursor and the active set becomes undefined.
- This will release the memory occupied by the cursor and its data set both on the client and server.
- **Syntax**: close <cursor_name>;
- **Example**: close emp_update;
- **Attributes of Explicit cursor:**

Attribute Name	Description
Cur_name%ISOPEN	Result is TRUE, if an explicit cursor is open else FALSE. Toaccess this attribute.
Cur_name%FOUND	Result is TRUE, if the last fetch succeeded because a row wasavailable or to FALSE if the last fetch failed because no more rows were available.
Cur_name%NOTFOUND	It is logical opposite of %FOUND. It is TRUE, if the last fetchhas failed because no more rows were available or to FALSE,
Cur_name%ROWCOUNT	Returns the number of rows fetched from the active set. It isset to zero when the cursor is opened.

- **State the Difference between Implicit & Explicit cursor:**

S_No.	Implicit Cursor	Explicit Cursor
1	It is opened by the oracle enginefor its internal processing.	It is declared & opened by the user for dataProcessing.
2	Default name of the cursor is SQL.	User can create explicit cursor with anyname & also change the name.
3	Use SQL prefix before usingattributes of implicit cursor. Such as…SQL%FOUND	Use cursor name which is given by the userat declaration time before using attributes of explicit cursor. Suchas….cursor_name%FOUND
4	User has no required opening & closing the implicit cursor.	User must have to open the cursor forfetching the data & close the cursor after completion of operation.
5	It has not any type.	It has two types.Normal cursors.Parameterized cursors.
6	We cannot use the FETCHcommand with implicit cursor.	We must have to use the FETCH command.
7	Query which returns single rowimplicit cursor are used.	Query which returns multiple rows explicitcursor are used.

- **Parameterize Cursor and Cursor For Loop**
- **Parameterized Cursors:**
- A cursor can be declared with parameters, which allows you to pass values to the cursor.
- Values are passed to the cursor when it is opened, and they are used in the query when it is executed.
- With the use of parameters we can open & close the cursor many times with different values.
- Cursor with different values returns different active data set each time.
- Opening a parameterized cursor & passing values to the cursor open cursor_name (value / variable / Expression);
- **Syntax**: cursor <cursor_name> (Parameter_namedata_type) is <select statement>;**Example:**

```
declare
   dnoemp.deptno%type;
   cursorcur_dept(no number) is select ename,dname,loc,sal from dept, emp where
   emp.deptno=dept.deptno and dept.deptno=dno;
begin
                dno:=&dno;
                dbms_output.put_line('Employee Name' || 'Department Name' || 'Location' ||
                'salary');
                forcur_record in cur_dept(dno)
                loop
                  dbms_output.put_line(cur_record.ename ||' ' ||cur_record.dname || ' ' ||
                  cur_record.loc || ' '||cur_record.sal);
                end loop;
end;
```

- **Cursor For loops:**
- It is used to process multiple records.
- Advantage of cursor for loop is the loop itself will open a cursor, fetches the records into the cursor from the table until EOF is encounters then it closes the cursor.
- Variable used in cursor for loop is automatically created by the cursor for loop.
- A cursor can be closed even when an exit or a jumping statement is used to leave the loop in advance, of if an exception is raised inside the loop.
- **Syntax:**
- for <variable> in <cursor_name> loop statements
- end loop;**Example:**

```
declare
          cursorcur_emp is select empno,ename,job,sal from emp;
          v_total number(5) :=0;
begin
          forcur_record in cur_emp
          loop
                dbms_output.put_line(cur_record.empno|| ' ' || cur_record.ename|| '  '  ||
                cur_record.job || '        ' ||cur_recorde.sal);
                v_total:=V_total+cur_recor.sal;
          end loop;
                dbms_output.put_line('total salary is :- ' || v_total);
end;
```

SMT L.P SAVANI SATUABABA BCA COLLEGE

B.C.A. SEM-VI

Subject: RDBMS using Oracle-II

UNIT-2

Advance PL/SQL Programming

- Exception Handling
- Stored Procedure and Function
- Trigger
- Data Concurrency and locking

SAVANI

Q-1 Explain Exception Handling in Oracle.

❖ **What isException?**

- Exception is a one type of error.
- Exception can be user defined or System Exceptions.
- Exceptions are automatically generated by PL/SQL.

❖ **SystemExceptions:**

- **Cursor_Already_Open(ORA –06511)**
  - When user tries to open a cursor which is already open this error isgenerated.
- **Invalid_Cursor(ORA –01001):**
  - Whenuserperformsanyinvalidcursoroperationthiserrorisgenerated.
- **Dup_Val_On_Index(ORA –00001)**
  - When user tries to enter a duplicate value in a unique column this error israised.
- **Invalid_Number(ORA –01722)**
  - When user tries to enter other then numeric value in numeric field this error isgenerated.
- **Login_Denied(ORA –01017)**
  - When user tries to login with invalid User name & Password this exception willgenerated.
- **No_Data_Found(ORA –01403)**
  - When single row select statement found no data then this error willgenerated.
- **Not_Logged_On(ORA –01012)**
  - This exception is generated when user is not connected tooracle.
- **Program_Error(ORA –06501)**
  - When PL/SQL block has an internal error this erroris generated.
- **Storage_Error(ORA –06500)**
  - When PL/SQL block have insufficient memory to execute this error isgenerated.
- **Too_Many_Rows(ORA –01422)**
  - This error will be generated when single rowselect statement generates more than one row.
- **Value_Error(ORA –06502)**
  - This exception is raised whenever the user encounters an arithmetic, conversion or constraints error.
- **Zero_Divide(ORA –01476)**
  - This error is raised whenever PL/SQL block tries to divide any value byzero.

:: Example of System Generated Exception:

```
declare
        salary number;
begin
        selectsal into salary from emp where empno=100;
exception
        whenno_data_found then
        dbms_output.put_line("No records are found for empno 100");
end;
```

❖ **User DefinedExceptions:**

- User defined exceptions must be declared in the DECLARE section with the reserve wordEXCEPTION.

  <EXCEPTION_NAME> EXCEPTION;
- This exception can be brought into action by the following command in executablesection.

  RAISE <EXCEPTION_NAME>;
- When the exception is raised, processing control is passed to the EXCEPTION section of the PL/SQLblock.
- Therefore, the code for the exception must be defined in the EXCEPTION part of the PL/SQLblock.

  WHEN <EXCEPTION_NAME> THEN
  <ACTION>;

:: Example of User Defined Exceptions::

```
Declare
comission number (5,2);
        null_comission exception;
begin
    selectemp_comission into comission from emp where emp_no = 100;
    ifcomission is null then
        raisenull_commission;
    end if;
exception
    whennull_comission then
    dbms_output.put_line('no comission found for given employee');
end;
```

Q-2 Explain Stored Procedure and Function with example.

- A stored procedure or functions is a logically grouped set of sql and pl/sql statements that perform a specifictask.
- This is similar to a procedure/function in other programming languages.
- A stored procedure/function has a headerand a body.
- The header consists of the name of the stored procedure/function and the parameters or variables passed to the stored procedure/function.
- The body includes declaration section, execution section and exception section similar to a general PL/SQL Block.
- There are important difference between anonymous block and procedure is that:
  - ✓ Anonymous blocks stored as file & must be compiled and then run each time the script is loaded while stored procedure/function is stored as object in database in binary form so it is compiled only once when it iscreated.
  - ✓ Anonymous block cannot accept parameter as well as cannot return any value to and from external application while stored procedure/function can accept parameter as well as return value to and from externalapplication.

❖ **Advantage of using procedure andfunction:**

- **Security:**
  - ➢ It provides datasecurity.
  - ➢ Such as by giving permission to a procedure or function that can query a table and granting the procedure to users, not need permission to manipulate the tableitself.
- **Performance:**
  - ➢ Amount of information sent over a network isless.
  - ➢ No compilation is required to execute thecode.
  - ➢ Once the procedure is present in the shared pool of the SGA retrieval from disk is not required every time different users call the procedures or function. Such as reduction in disk I/O.
- **Memoryallocation:**
  - ➢ It provides shared memory facility.
  - ➢ Only one copy of procedure needs to be loaded for execution by multiple users.
  - ➢ Once a procedure is opened in the Oracle engine's memory, other users who have appropriate permission may access it whenrequired.
- **Productivity:**
  - ➢ By writing procedures repetitive coding can be avoided so increasing the productivity.
- **Accuracy:**

:: Syntax to create stored procedure ::

```
CREATE OR REPLACE PROCEDURE < procedure name >
  (<argument> {IN/ OUT/ IN OUT} <DATA TYPE>...) IS.AS}
  <variable> declarations:
  <constant> declarations:
 BEGIN
  <PL/SQL subprogram body>:
 EXCEPTION
  <exception PL/SQL block>:
  END:
```

- Aprocedureneedstobetestedonlyonceto guaranteethatitreturnsanaccurateresult.

:: Example of stored procedure::

```
create or replace procedure delete_emp(eno in number) is

 BEGIN
 delete from emp where empno= eno;
 END;
```

:: How to execute procedure? ::

```
start procedure1.sql;
execute delete_emp(10);
```

:: To Drop Procedure::

Syntax: drop procedure <procedure_name>;
Example: drop procedure delete_emp;

❖ **Keywords and parameters used in Procedure /Function:**

Keywords	Descripti on
Replace	Recreates the procedure if already exists. If a procedure is redefined the Oracle engine recompiles it.
Schema	Give the schema name which contains the procedure. The oracle engine takes the default schema to be the current schema, if it is omitted.
Procedure	It Is the name of the procedure to be created.
Argument	It Is the name of an argument to the procedure. Parentheses can be omitted if no arguments are present.
IN	Indicates that the parameter will accept value from user.
OUT	Indicates that the parameter will return a value to the user.
IN OUT	Indicates that the parameter will either accept a value from the user or return a value to the user.
Data type	It Is the data type of an argument. It supports any data type supported by PL/SQL.

:: Example of stored procedure ::

```
create or replace procedure find_name(v_empno in number) as
        v_enameemp.ename%type;
        v_jobemp.job%type;
        v_salemp.sal%type;
 begin
  select ename,job,sal into v_ename,v_job,v_sal from emp where empno=v_empno;
 dbms_output.put_line(' Name ' || v_ename || ' designation ' || v_job || ' salary ' || v_sal);
 exception
    when NO_DATA_FOUND then
    dbms_output.put_line(' Record not found, please try again');
 end;
```

Sample stored procedure that has in & out parameter

```
create or replace procedure addvalue(num1 in number,num2 in number, answer out number) is
begin
        answer:=num1+num2;
        dbms_output.put_line('answer of addition is :- ' || answer);
end;
```

- If stored procedure has out parameter, then it should be called from either other stored procedure/function or anonyms block.
- Above created "addvalue" stored procedure can be called from anonyms block in following way

```
        Set serveroutput on;
        declare
                ans number(5,2);
        begin
                addvalue(10,20,ans);
                dbms_output.put_line('answer of addition is ' || ans);
```

```
end;
```

:: Syntax to create Function ::

```
CREATE OR REPLACE FUNCTION < Function name >
  (<argument> {IN/ OUT/ IN OUT} <DATA TYPE>...) return <data type>{IS.AS}
  <variable> declarations:
  <constant> declarations:
 BEGIN
  <PL/SQL subprogram body>:
 EXCEPTION
  <exception PL/SQL block>:
```

:: Example of function that returns value ::

```
create or replace function factorial(num in integer) return integer as
        ans integer(5) :=1;
begin
        fori in 1..num loop
                ans:=ans*i;
        end loop;
        returnans;
end;
```

:: Function called from another anonyms block ::

```
setserveroutput on;
declare
       answer integer(5) :=1;
begin
        answer:= factorial(5); dbms_output.put_line('answer is :-' || answer);
end;
```

❖ **Difference between procedure &function:**

Procedure	Function
Not require to return value.	Must require to return value.
Using multiple OUT parameters we can return multiple values.	It returns only single value at a time.
Procedure keyword is used.	Function keyword is used.

Q-3 Whar is Trigger? Explain components and types of trigger.

- A database trigger is a named PL/SQL block stored in a database and executed implicitly (automatically) when a specific eventoccurs.
- An act of executing a trigger is referred to as firing a trigger.
- A triggering event is a DML (INSERT, UPDATE, or DELETE) statement executed against a database table.

- A trigger is a compiled procedure stored in the database.
- The language you use is PL/SQL. You code and compile a trigger in the same manner you code stored procedures.
- A trigger can fire before or after a triggering event.
- For example, if you have defined a trigger to fire before an INSERT statement on the STUDENT table, this trigger fires each time before you insert a row in the STUDENT table.
- The general syntax for creating a trigger is as follows (the reserved words and phrases surrounded by brackets are optional):

❖ **Components ofTriggers:**

- **Triggering event orstatement:**
  - It is a SQL statement that causes a trigger to befired.
  - INSERT, UPDATE, or DELETE statement for a specifictable.
- **TriggerRestriction:**
  - It specifies a Boolean expression that must be TRUE for the trigger tofire.
  - It is option available for triggers that are fired for eachrow.
  - Its function is to conditionally control the execution oftrigger.
  - Trigger restriction is specified using a WHENclause.
- **TriggerAction:**
  - Trigger action is the PL/SQL code to be executed when a triggering statement is encountered and any trigger restriction evaluates toTRUE.
  - It contains SQL & PL/SQL statements.

❖ **Types ofTriggers**

- **Rows triggers:**
  - It is fired each time a row in the table is affected by the triggering statement.
  - For example if update statement updates multiple rows of a table, a row trigger is fired once for each row affected by update statement.
  - If triggering statement affects no rows, the trigger is not executed atall.
  - It is used when some processing is required whenever a triggering statement affects a single row in atable.
- **Statementtriggers:**
  - Statement level triggers are triggered only once.
  - For example, when an update command updates 15 rows, the commands contained in the trigger are executed only once, and notwith every processed row.
- **Beforetriggers:**
  - It executes the trigger action before the triggering statement.
  - It is used in the following situation….
    - ✓ User can eliminate unnecessary processing of the triggering statement.
    - ✓ It is used to derive specific column values before completing a triggering INSERTor UPDATEstatement.
- **Aftertriggers:**
  - After trigger executes the trigger action after the triggering statement is executed.
  - It is used in the followingsituation…
    - ✓ After triggers are used when the triggering statement should complete before executing the trigger action.
    - ✓ If a before triggeris already present,an after trigger can perform differentactionson

the same triggering statement.

❖ **Point toremember:**

- There can be only one trigger of particular type that is for UPDATE, for INSERT or for DELETE.
- Only one table isspecified in the triggering event.
- Triggers cannot include commit, rollback and savepoint statement.
- Inside the trigger the correlation name :NEW and :OLD can be made use of to refer to data on the command lineand data in the table respectively.

:: **Syntax to create a Trigger ::**

```
create or replace trigger <trigger_name>
{before/after}
{insert/update/delete} of column on <table_name>
[For each row]
[When condition]
[PL/SQL block];
```

SAVANI

:: Create Table emp10 ::

```
Create table emp10(Eno number (7),Name varchar2 (30), Salary number (7,2) );

Table creterd…
```

:: Example of Trigger ::

```
editemp_delete.sql
set  serveroutput  on
set verify off
create or replace trigger emp_delete before delete on emp for each row
declare
        enoemp.empno % type;
        nameemp.ename  %  type;
        salary emp.sal % type;
begin
        eno:=:old.empno;
        name:=:old.ename;
        salary:=:old.sal;
        insert into emp10 values(eno,name,salary);
end;
```

:: Execution of trigger::

```
startemp_delete.sql; trigger
is created…..
delete from emp where ename='TURNER';
select * from emp10;
```

Q-4 Explain Data Concurrency and Locking in detail.

- Locks are mechanisms used to ensure data integrity while allowing maximum concurrent access ofdata.
- Oracle locking is fully automatic & requires no user intervention.
- The oracle engine (server machine) locks table data while executing SQL statement. This type of
- locking is called "implicit locking".
- Oracle default locking strategy is implicit locking.
- Since the oracle engine has a fully automatic strategy, it has to decide on two issues:-
  1) Types of lock to beapplied.
  2) Level of lock to beapplied.

❖ **Level ofLocks:**

1) Row Level:-If the Where clause evaluates to only one rowin thetable.
2) Page Level: - If the Where clause evaluates to a set of data.
3) TableLevel:-IfthereisnoWhereclause(i.e.thequeryaccessestheentiretable).

❖ **ExplicitLocking:**

- The technique of lock taken on a table or its resources by abusers called "Explicit Locking".
- Users can lock tables they own or any table on which they have been granted table privilege(select ,insert, update ,delete)
- Table or rows can be explicitly locked by using either the select …for update stmt. Or Lock tablestmt.

- **The select …. For Updatestatement:-**
- This clause is generally used to signal the oracle engine that data currently being used needs to beupdated.
- Ex:-Two client machines client A & client B are recording the transaction performed in a bank for a particular account no.simultaneously.
- Client A fires the following selectstatement:
- Client A>select * from acct_mstr where acct_no='Sb9' forupdate;
- When the above select statementis fired th2e oracle engine locks the record 'sb9'.
- This lock is released when a commit or rollback is fired by clientA.
- Now client B fires a select stmt.,which points to recordsb9.

Using Lock table statement:-

- Use the LOCK TABLE statement to lock one or more tables, table partitions, or table sub partitions in a specified mode. This lock manually overrides automatic locking and permits or denies access to a table or view by other users for the duration of your operation.
- A **LOCK** is a mechanism that prevents destructive interaction between two simultaneous transactions or sessions trying to access the same databaseobject.

Syntax:-

```
LOCK TABLE<TableName>[,<TableName>]…
IN{ROW SHARE|ROW
EXCLUSIVE|SHAREUPDATE|
        SHARE|SHARE ROW
        EXCLUSIVE|EXCLUSIVE} [NOWAIT]
```

Example:-

The following statement locks the employees table in exclusive mode but does not wait if another user already has locked the table:

```
LOCK TABLE employees IN EXCLUSIVE MODE NOWAIT;
```

Output:-TableLocked.

- **Lock modeClause:**

Specify one of the following modes:

- **ROW SHARE**:-ROW SHARE permits concurrent access to the locked table but prohibits users from locking the entire table for exclusive access. ROW SHARE is synonymous with SHARE UPDATE, which is included for compatibility with earlier versions of OracleDatabase.
- **ROW EXCLUSIVE**:-ROW EXCLUSIVE is the same as ROW SHARE, but it also prohibits locking in SHARE mode. ROW EXCLUSIVE locks are automatically obtained when updating, inserting, or deleting.SHARE UPDATE See ROWSHARE.
- **SHARE:-**SHARE permits concurrent queries but prohibits updates to the lockedtable.
- **SHARE**:-ROW EXCLUSIVE SHARE ROW EXCLUSIVE is used to look at a whole table and to allow others to look at rows in the table but to prohibit others from locking the table in SHARE mode or from updatingrows.
- **EXCLUSIVE**:- EXCLUSIVE permits queries on the locked table but prohibits any otheractivity onit.

- **NOWAIT :-** Specify NOWAIT if you want the database to return control to you immediatelyifthe specified table, partition, or table sub partition is already locked by another user. In this case, the database returns a message indicating that the table, partition, or sub partition is already locked by another user.
- If you omit this clause, then the database waits until the table is available, locks it, and returns control toyou.

❖ **Releasinglocks:**

All locks are released under the followingcircumstances:

1) The transaction is committedsuccessfully.
2) A rollback isperformed
3) A rollback to a savepointwill release locks set after the specifiedsavepoint.

Note: - commit: Save Work done.

Savepoint: Identify a point in a transaction to which you can later rollback.

Rollback: Restore database to original since the last COMMIT.

GRANT/REVOKE: Grant or back permission to or from the oracle users.

SAVANI

SMT L.P SAVANI SATUABABA BCA COLLEGE

[Bachelor of Computer Application]
BCA
Semester-VI

Subject: 605- RDBMS using Oracle-II

Unit - 3

Introduction to DBA

- Role of
- DBA.
- Users: Creating a new user, grant command, deleting user.
  Privileges: System privileges, Object privileges, Assigning object privileges to a user, Viewing User & privileges, revoking a system & an object privileges.
- Role: Creating a role, Grating privileges & roles to a role, granting role to a user, viewing the role of a user.

SAVANI

3.1 Role of DBA:

❖ **What is DBA?**

- DBA stands for Database Administrator.
- Database administrator is a person or group of persons under the supervision of a knowledgeable person called Administrator.
- DBA controls the database structure and sets up the definition for physical as well as logical creation of the database.

❖ **Role of DBA/Job of DBA:**

- Install and upgrade Oracle and its tools.
- Configure the Oracle instance and SQL * Net.
- Create a database.
- Create, alter, and remove database users and roles.
- Grant and restrict access rights.
- Allocate and manage physical and logical storage structures.
- Develop security strategies.
- Develop backup and recovery procedure.
- Monitor system performance.
- Analyze database performance and implement solutions to problem.
- Communicate with Oracle support service personnel.
- Troubleshoot locking problems.

3.2 Users: Creating a new user, grant command, deleting user:

- THE concept of users is very important in RDBMS for database administrators.
- Anyone who wants to access oracle must be registered as a user of the database, and must have privileges assigned to him or her for the tasks he or she will be able to execute.
- Roles are groups of privileges that can be assigned to users.

❖ **Users:**

- One of the most important tasks of an administrator is controlling access to the database.
- Oracle relies on a mechanism that allows you to register a person, called user.
- Each registered user has an access password, which must be provided in various situations.
- Each user is then assigned individual privileges or roles.

• **THE CREATE USER COMMAND:**

- The CREATE USER COMMAND is used for the creation of new users.
- It creates a user or an account through which you can connect to the database.
- You can assign the following optional properties to the user:
  - Standard tablespace.
  - Temporary tablespace.
  - Quotas to allocate space in the tablespace.
  - Profile containing resource limits.

SAVANI

:: Syntax to create new user::

create user <user_name> identified {by password | Externally |Globally as 'cn=user'} [default tablespace tablespace_name |Temporary tablespace tablespace_name| Quota { integer [K | M] | Unlimited} on tablespace]…
|profile profile_name |password expire |accout { Lock | Unlock }…

- **Syntax** :- create user <user_name> identified by <password>;
- **Example**: create user anil identified by anil123 default tablespace system profile default;

❖ **To assign privileges (rights) to new user:**

- To assign privileges, you must connect to the administrator using the SYSTEM name and the MANAGER password.
- To connect to the database as a new user, you must use the CONNECT command with following
  syntax: CONNECT user/password
  Example: CONNECT anil/anil123
- Both user name & password can be up to 30 characters long.
- If enter a user name that has already been registered, the system will issue an error message and will not allow you to register the new user.
- When user is created you have to assign him or her privileges.
- Before assigning privilege we will try to connect.
- So we have to assign the privileges using the GRANT command.

- **Explain GRANT command:**
  - The GRANT command has two distinct forms: one to distribute system privileges and other to distribute object privileges.
  - Only DBA can use the GRANT command to distribute systemprivileges.

- **Syntax:** GRANT system_privileges/ role TO user / role/ public with admin option;
- **Example:** GRANT DBA to anil;

- **Privilege:**-The name of the privilege to be assigned.
- **User/Role:**-The name of the user or role that is receiving the privileges.
- **With admin option:** Allows user or role that receives the privilege to assign it to other users, change it, or even delete it.

❖ **Explain DELETE USER:**

- We can remove a database user with the DROP USER command.
- This command removes both the user and all the objects contained in this user's schema.
- We will need to specify the CASCADE clause of the command.
- To see all users created in system the command is :- **select * from all_users;**

- **Syntax:** DROP user <user_name><cascade>;
- **Example**: DROP user anil cascade;

3.3 Privileges: System privileges, Object privileges, Assigning object privileges to a user, Viewing User & privileges, revoking a system & an object privileges.

❖ **What is a privilege?**

- A privilege is a permission given to the user to access and manipulate a database object in a certain way.
- For example, one user can be assigned the privilege to select tables but not change th em, while another user can read, update and even change the structure of tables and other objects.
- There are 2 types of privileges: System privilege and Object privilege.

❖ **System Privilege:**

- A system privilege is the permission to execute certain database actions in a specific type of database object.
- More than 70 types of privileges associated to the actions of a database.
- Name of the privilege is similar to the name of the action it executes.
- For example, a privilege called ALTER TABLE grants a user the right to modify a table.
- List of Some System Privileges:

System Privilege	Type of Action
CREATE	Create an object in user's own schema.
CREATE ANY	Create an object in another user's schema.
CREATE SESSION	Connect to the database.
DROP	Drop any object in user's own schema.
DROP ANY	Drop any object in another user's schema.
ALTER SYSTEM	Manipulate an Instance
ALTER DATABASE	Manipulate the database
ALTER USER	Change user roles or password.
ALTER/ CREATE/ DROP/ MANAGETABLESPACE	Manipulate a tablespace

- **Syntax:** GRANT system_privileges TO user with admin option;
- **Example:** GRANT CREATE SESSION to anil;

❖ **Object Privilege:**

- An object privilege is the right to perform certain actions in a specific object, such as the right to include a row in a table.
- It is not apply to all the objects of a database.
- Triggers, procedures, indexes, database links and clusters do not have any object privilege.
- List of Object Privileges:

Object Privilege	Object
ALTER	Table & Sequences
DELETE	Tables & Views
EXECUTE	Procedures
INDEX	Tables
INSERT	Tables & Views
REFERENCE	Tables
SELECT	Tables, Views, and Sequences
UPDATE	Tables and Views

- **Assigning object privileges to a user/role:**
  - A variation of the GRANT command allows you to assign object privileges to a user or role.
  - The privileges assigned can be: SELECT, ALTER, DELETE, EXECUTE, INSERT, INDEX, REFERNCE, AND UPDATE.
  - The following objects can be assigned privileges: tables, views, sequences, snapshots, and synonyms.
  - Any user who has authority can grant privileges on tables.

Syntax: GRANTT ALL
select / Insert / Delete / Index/ Alter /
Update [(Coumn_name)] on table_name/ view_name
TO user/public
with admin option

Example: CONNECT scott/tiger
GRANT select on emp to anil;

- So now emp table is also accessed by anil.
- Because owner of emp table scott gives privilege to the user anil.
- But anil can only see the data he could not perform any insert, update, delete, etc. operation.

- **Select**:-To select data in a table or view.
- **Insert**:-To insert rows in a table or view.
- **Delete**:-To eliminate rows in a table or view.
- **Update**:-To update a table and, optionally, update only the specified columns.
- **Index**:-To create or eliminate the indexes of a table.
- **Alter**:-To modify a table.
- **All**:-To perform all the above privileges.
- **Table_name**:- Name of the existing tables in the database.
- **View_name**:- Name of the existing views in the database.
- **Column_Name**:- This is optional & determines the columns of tables or views specified in the ON clause. The name of the columns must not be qualified.
- **User**:-Name of the user to whom the privilege is assigned.
- **Public**:-Means that all the current and new users have the privileges specified for the table or view.
- **With Admin Option**: - Allows the user or role that receives the privilege to grant it to other users, modify it, or even delete it.

❖ **Revoking a System Privileges:**
- We can use REVOKE command to revoke any system privileges.
- System privilege must have been granted with the Admin Option.

Example: REVOKE session from anil;

❖ **Revoking an Object privileges:**
- To revoke privilege we have to connect as a user who has the authority to grant privilege.
- A user can not grant or revoke privileges to himself or herself.

Example: connect scott/tiger
revoke select on emp from anil;

- So now privilege has been revoked, anil cannot access the table anymore.

3.4 Role: Creating a role, Grating privileges & roles to a role, Granting role to a user, viewing the role of a user.

❖ **What is Role?**

- Role is a group of privileges assigned to a particular user name.
- Instead of granting eight privileges to a user, we can create a role that is assigned those eight privileges to the user.
- Role simplifies the administration of users.
- Oracle has several predefined roles. Such as….
  - CONNECT
  - RESOURCE
  - DBA
  - EXP_FULL_DATABASE
  - IMP_FULL_DATABASE
- We can also create user defined roles that meet the needs of our application.

- **CONNECT** :
  - It allows access permission to the database object.
  - If user is created with CONNECT role then we can select, update or deletes the table rows.
  - User cannot create the object as well as destroy.
- **RESOURCE**:
  - Allows access to the database.
  - Allows the creation of tables, sequences, procedures, triggers, indexes etc.
  - It allows DDL & DML command operations on the object.
  - Most of the users have RESOURCE role.
- **DBA:**
  - Provides all the system privileges.
  - It allows granting privileges to other users.
  - DBA user can create & drop any user.

- Default Oracle users, password & their roles

User Name	Password	Role
SCOTT	TIGER	CONNECT & RESOURCE
SYSTEM	MANAGER	DBA
SYS	SYS1	CONNECT, RESOURCE, DBA, EXP_FULL_DATABASE, IMP_FULL_DATABASE
DEMO	DEMO	CONNECT & RESOURCE

❖ **How we can create a Role?**

- **Syntax:**

```
CREATE ROLE Role_name
[NOT IDENTIFIED | IDENTIFIED { BY password | EXTERNALLY | GLOBALLY
```

- **Role_name:**
  - Name of the role to be crated.
- **Not Identified:**
  - Indicates that this role is authorized by the database.
  - No password is required to enable it.
- **Identified:**
  - Indicates that the user must be authorized by the specified method before the role can be enabled with the SET ROLE command.

- **BY password:**
  - User must specify the password for Oracle to enable the role.
- **Externally:**
  - Indicates that the user must be authorized by an external service such as operating system or a third party software before the role can be enabled.
- **Globally:**
  - Indicates that the user must have authorization from oracle security service to use the role before it can be enabled with the SET ROLE command, or at the login.

- **Example:** CREATE ROLE basic1 identified by anil123;

❖ **Granting Privileges and Roles to a role:**

- After creating role, we can grant it privileges using the GRANT command.
- **Syntax:**

  GRANT role/Name_of_privilege TO user/role/ PUBLIC WITH ADMIN OPTION;
- **Example:**

  GRANT DBA to basic1;
  - Above example grants the DBA role to basic1.

❖ **Granting a Role to a User:**

- Now we assign the basic1 role to user anil.
- **Example:** connect anil/anil123;

  GRANT basic1 to anil;

❖ **Viewing the roles of a user:**

- Oracle has some special views that control the privileges and roles.
- One of these views is called USER_ROLE_PRIVS.
- It displays the roles assigned to the current user. Such as….
- **Example:** select * from USER_ROLE_PRIVS;

❖ **Deleting a Role:**

- DROP ROLE command is used to delete a role.
- User must have the system privilege DROP ANY ROLE to remove a role.
- **Syntax:** DROP ROLE <Role_name>;
- **Example:** DROP ROLE basic1;

Assignment-3

Q-1: What is DBA? Explain the role of DBA.
Q-2: Explain creating user, grant user and deleting user Q-3: write a detail note on privileges.
Q-4: How we can assign the privileges and revoking the privileges?
Q-5: What is Role? How we can create a role and giving privileges to role explain in detail.

SMT L.P SAVANI SATUABABA BCA COLLEGE

B.C.A. SEM-VI

Subject: RDBMS using Oracle-II

UNIT4

Data warehousing and Data Mining

- Data ware housing Definition, usage and trends
- DBMS vs. data warehouse, Data marts, Metadata
- Data warehouse architecture
- Design and construction of data warehouse
- Introduction to data mining
- Classification and Applications of data mining system

SAVANI

Q-1 "What is Data ware house?ExplainApplications and Features.

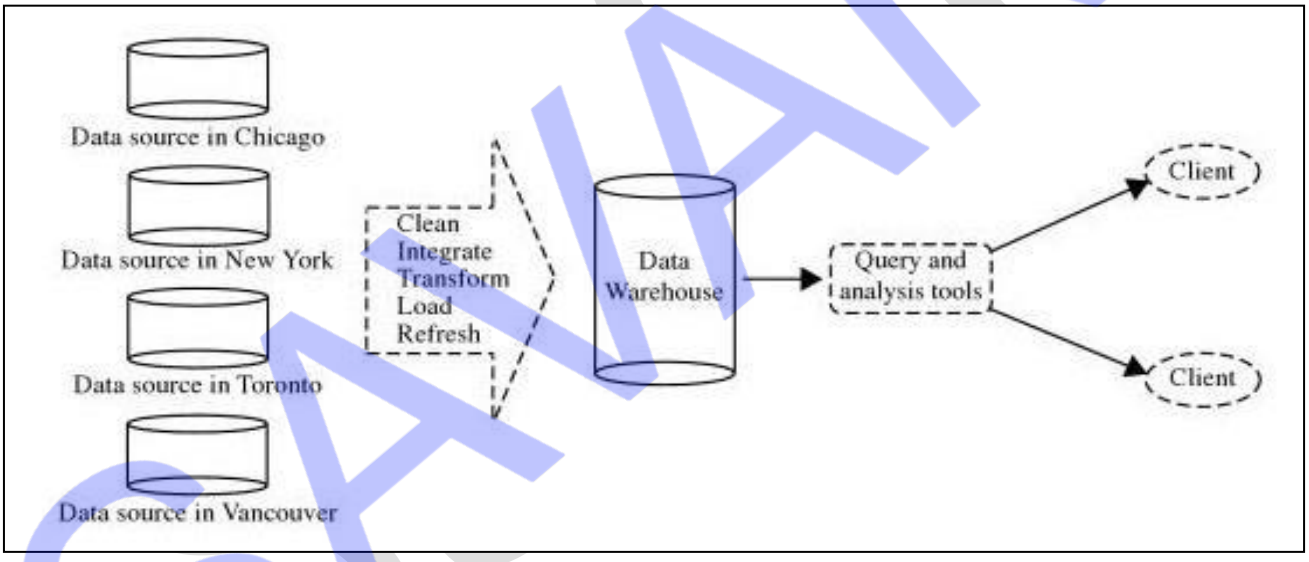
❖ **Datawarehouse:**

"The Data Warehouse is a database of unique data structure with data archiving capabilities that allows quick and easy performance of complex queries over large amounts of data"

OR

"A data warehouse is a subject-oriented, integrated, time-variant, and nonvolatile collection of data in support of management's decision making process"

- The term "Data Warehouse" was first coined by Bill Inmon in 1990.
- According to Inmon, a data warehouse is a subject oriented, integrated, time-variant, and non-volatile collection of data. This data helps analysts to take informed decisions in an organization.
- A data warehouse is a database, which is kept separate from the organizations operational Database.
- There is no frequent updating done in a data warehouse.
- It possesses consolidated historical data, which helps the organization to analyze its business.
- A data warehouse helps executives to organize, understand, and use their data to take strategic decisions.
- Data warehouse systems help in the integration of diversity of application
- systems. A data warehouse system helps in consolidated historical data analysis.



❖ **Application of Data warehouse (Usage):**

- A data warehouse helps business executives to organize, analyze, and use their data for decision making.
- A data warehouse serves as a sole part of a plan-execute-assess "closed-loop" feedback system for the enterprise management.
- Data warehouses are widely used in the following fields:
  - Financialservices.
  - Bankingservices.
  - Consumergoods.
  - Retailsectors.
  - Controlledmanufacturing.

Features of Data Warehouse.

(1) Subject-Oriented:

- A data warehouse is subject oriented because it provides information around a subject rather than the organization's ongoingoperations.
- These subjects can be product, customers, suppliers, sales, revenue,etc.
- A data warehouse does not focus on the ongoing operations, rather it focuses on modelling and analysis of data for decisionmaking.

(2) Integrated:

- A data warehouse is constructed by integrating data from heterogeneous sources such as relational databases, flat files,etc.
- This integration enhances the effective analysis of data.

(3) Time-Variant:

- Thedatacollectedinadatawarehouseisidentifiedwithaparticulartimeperiod.
- The data in a data warehouse provides information fromthe historical point of view.

(4) Non-Volatile:

- Non-volatile means the previous data is not erased when new data is addedto it.
- A data warehouse is kept separate from the operational database and therefore frequent changes in operational database is notreflected in the data warehouse.

SSVANI

Q-2DBMS vs Data Warehouse, Explain Data marts&Metadata.

❖ **Difference between DBMS and DataWarehouse:**

DBMS	Data Warehouse
DBMS is a software package.	Data warehouse is not software.
A database is designed to record data.	It is designed to analyze data.
A database is normally limited to asingle application.	It stores data from any number of applications.
Data is available real-time.	Data is refreshed from source systems when needed.
Is efficient in processing and storage.	Is efficient in analytics.
It is mostly used for storage.	Mostly support for reporting, visualization andanalysis.
Processing digital transactions.	Does not process digital transaction.
Not costly for single database.	Costly compared to a single database.
Entity – Relational modelingtechniques are used for RDMS database design.	Data – Modeling techniques are used for the Data Warehouse design.
Optimized for write operation.	Optimized for read operations.
Performance is low for analysisqueries.	High performance for analytical queries.

❖ **DataMarts:**

"A data mart is a subset or an aggregation of the data stored to a primary data warehouse."

- It includes a set of information pieces relevant to a specific business area, corporate department, or category of users.
- Data marts can be viewed as small, local data warehouses replicating the part of a primary data warehouse required for a specific application domain.
- The Data Marts populated from a primary DWH are often called dependent. Data marts are not strictly necessary, they are very useful for DWH systems in midsize to large enterprises because they are used as building blocks while incrementally developing data warehouses;
- They mark out the information required by a specific group of users to solve queries;
- They can deliver better performance because they are smaller than primary data warehouses.

❖ **Metadata:**

"Meta-data is the data used to define other data".

- It specifies source, values, usage, and features of DWH data and defines how data can be changed and processed at every architecture layer.
- Meta-data is stored in a meta-data repository which all the other architecture components can access it.
- According to Kelly, a tool for meta-data management should allow administrators to perform system administration operations, manage security,allow end users to navigate and queryMeta-Data.
- Allow end users to Uses a GUI.
- Allow end users to extend meta-data.
- Allow meta-data to be Imported / Exported into / from other standard tools and formats.

Q-3 Explain Data Warehouse Architecture in detail.

⌗ The Architecture Properties are essential for DWH system:

1. **Separation:** Analytical and transactional processing should be kept apart as much aspossible.
2. **Scalability:** Hardware and software architectures should be easy to upgrade as the data volume, which has to be managed and processed, and the number of users' requirements, which have to be met, progressivelyincrease.
3. **Extensibility:** The architecture should be able to host new applications and technologies without redesigning the wholesystem.
4. **Security:** Monitoring accesses is essential because of the strategic data stored in data warehouses.
5. **Administrator-ability:** Data warehouse management should not be overlydifficult.

❖ **Three-Tier data warehousearchitecture:**

(1) BottomTier:

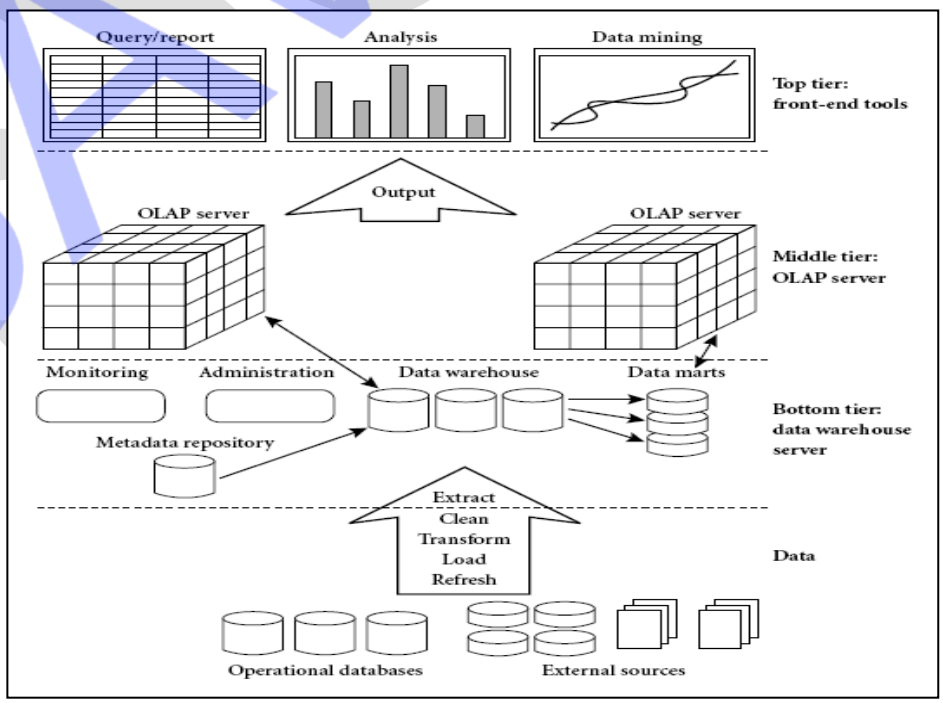
- Thebottomtierofthearchitectureisthedatawarehousedatabaseserver.
- It is the relational database system. We use the back end tools and utilities to feed data into the bottomtier.
- These back end tools and utilities perform the Extract, Clean, Load,and refresh functions.

(2) MiddleTier:

- In the middle tier, we have the OLAP Server that can be implemented in either of the followingways.
- By Relational OLAP (ROLAP), which is an extended relational database managementsystem. The ROLAP maps the operations on multidimensional data to standard relational operations.
- By Multidimensional OLAP (MOLAP) model, which directly implements the multidimensional data andoperations.

(3) Top-Tier:

- This tier is the front-end clientlayer.
- Thislayerholdsthequerytoolsandreportingtools,analysistoolsanddataminingtools.



[Figure: Three-Tier data warehouse architecture]

Q-4ExpainDesign and Construction of Data Warehouse.

- HavingDWmayprovide**competitive advantage**bypresentingrelevantinformationfromwhich to measure performance and make critical adjustments in order to win overcompetitors.
- DW can **enhance business productivity** because it is able to quickly and efficiently gather information that accurately describes the organization.
- DW facilitates **customer relationship management** (CRM) because it provides a consistent (constant or reliable) viewof customers and items across all lines of business, all departments, and all markets.
- DW may bring about **cost reduction** by tracking trends, patterns, and exceptions over long periods in a consistent and reliablemanner.
- The target of the design becomes how the record from multiple data sources should be extracted, transformed, and loaded (ETL) to be organized in a database as the data warehouse.

  - To design an effective DW we need to understand and analyze business needs and construct a **business analysisframework.**
  - The construction of large and complex information system can be viewed as construction of large and complex building, for which the owner, architect, and builder have different views.
  - These views are combined to form a complex framework that represents the **top-down**, (**business-driven** or **owner's perspective)**, and **bottom-up**, (**builder-driven)** or **implementer's view** of informationsystem.
  - 4differentviewsregardingthedesignofdatawarehousemustbeconsidered:**the*top-down view*, the *data source view*, the *data warehouse view*, and the*business query view***.

1) **The top-down view:**
   - It allows selection of relevant information necessary for DW.
   - This information matches current and future businessneeds.
2) **The data source view:**
   - Information being captured, stored, and managed by operational systems.
   - This information may be documented at various levels of detail and accuracy, from individual data source tables to integrated data source tables.
   - Data sources are often modeled by traditional data modeling techniques, such as the entity-relationship model or CASE (computer-aided software engineering)tools.
3) **The data warehouse view:**
   - It includes fact tables and dimension tables.
   - It represents the information that is stored inside the data warehouse, including pre-calculated totals and counts, as well as information regarding the source, date, and time of origin, added to provide historicalcontext.
4) **The business query view:**
   - It is the perspective of data in the data warehouse from the viewpoint of the enduser.

Q-6 What is Data mining? Also explain Applications of Data Mining.

❖ **Definition of Datamining:**

"Data mining is the task of discovering interesting patterns from large amounts of data where the data can be stored in databases, data warehouses or other information repositories."

- There is huge amount of data available in Information Industry.
- This data is of no use until converted into useful information.
- Analyzing this huge amount of data and extracting useful information from it is necessary.
- Data Mining is defined as extracting the information from the huge set of data.
- In other words we can say that data mining is mining the knowledge from data.
- This information can be used for any of the following applications:
  - MarketAnalysis, FraudDetection, CustomerRetention, ProductionControl, ScienceExploration.

❖ Classification and Applications of data miningsystem.

❖ **Data MiningApplications**

- Here is the list of applications of Data Mining:
  - Market Analysis and Management.
  - Corporate Analysis & RiskManagement.
  - FraudDetection.
  - OtherApplications.

(1) Market Analysis andManagement:

1) **Customer Profiling -** Data Mining helps to determine what kind of people buy what kind of products.
2) **Identifying Customer Requirements -** Data Mining helps in identifying the best products for different customers. It uses prediction to find the factors that may attract newcustomers.
3) **Cross Market Analysis -** Data Mining performs Association/correlations betweenproduct sales.
4) **Target Marketing -** Data Mining helps to find clusters of model customers who share thesame characteristics such as interest, spending habits, income etc.
5) **Determining Customer purchasing pattern -** Data mining helps in determining customer purchasingpattern.
6) **Providing Summary Information -** Data mining provide us variousmultidimensionalsummary reports.

(2) Corporate Analysis & RiskManagement:

1) **Finance Planning and Asset Evaluation -** It involves cash flow analysis and prediction, contingent claim analysis to evaluateassets.
2) **Resource Planning -** Resource Planning It involves summarizing and comparing the resources andspending.
3) **Competition -** It involves monitoring competitors and marketdirections.

(3) FraudDetection:

- Data Mining is also used in fields of credit card services and telecommunication to detectfraud.
- Infraudtelephonecall it helpstofinddestinationofcall,durationofcall,timeofdayorweek.
- It also analyze the patterns that deviate froman expected norms.

(4) OtherApplications:

- Data Mining also used in other fields such as sports, astrology, telecommunication industry, retail industry, financial data analysis, biological data analysis, etc.