Date ______
Page ______

Overview of PL/SQL

Introduction: Structured Query Language. (SQL)

SQL is powerful language to DBA (DataBase Administrator). But it suffers with several disadvantages when used as a conventional programming language.

SQL doesnot have procedural capabilities that means SQL doesnot provide the basic programming constructs for condition checking, looping and ~~braching~~ branching, that are essential for writing programs.

2) SQL (~~commands~~ statements) are passed to the oracle engiene one at a time. So, we cannot execute a group of SQL commands. This adds additional traffic on the network, therefore decreasing the speed of data processing, specially in a multi-users environment.

3) Generally, oracle engiene displays it's own error messages when processing SQL statements and we cannot customize error messages.

Introduction to PL/SQL

→ The PL/SQL programming language was developed by oracle corporation as a procedural extension of SQL commands. It supports the development of structures program and works as a bridge in between database and ~~to~~ front end tools.

mons
In t.

PL/SQL Block structure :

```
DECLARE
        <declaration-section>
BEGIN
        <executable commands>
EXCEPTION
        <exception-handling-statements>
END;
```

Fig:- A Generic Structure of PL/SQL Block.

~~The generic block of PL/SQL program consist of the following sections:~~

PL/SQL Block structure

→ The generic block of PL/SQL program consists of the following sections :

① DECLARE:

→ The declaration section that starts with the keyword declare, declares all variables, constants, cursors, sub programs and other elements to be used in a program.

② BEGIN :

→ The begin section is a compulsory section of all PL/SQL programs. It consists of executable SQL and PL/SQL Structures which describe the process that have to be applied to the tables. Actual data manipulation, data retrieval, looping, branching are statements are specified in this section.

③ EXCEPTION :

This section deals with the handling of e during the execution of data manipulation staten inside the begin section. It begins with the key exception and it is optional.

④ END :

Every PL/SQL block must be terminated END; (semicolon)

It indicates the end of the program.

Variables:

→ Variable in PL/SQL begin with a characte
maximum 30 characters variables are the mem
Locations used to hold values. In PL/SQL we
assign some value to the variable using assig
operator (:=) And, we can also fetch data
variables from data tables. Variables are declar

Variable Name, Data type ;

For eg:

```
balance FLOAT,
name VARCHAR(30);
balance = 75405.50;
SELECT S.name INO name →variable
FROM Student WHERE Reg.NO = 1001;
        field
```

Constant :

→ Declaring a constant is similar to decla
a variable except that the Keyword constant m
be added to the variable name and a value
assigned immediately. There after no further ass
to the constants are possible.

Syntax:

Constant_Name CONSTANT dataType = value

for eg:-

```
balance CONSTANT FLOAT = 100.00;
```

Condition Control Structure : IF - THEN - ELSE IF - ELSE - END IF

Syntax :

```
        If (condition) THEN
          action;
        ELSE condition THEN
            action;
         ELSE
            action;
         END If ;
```

For eg :
Assume that we have a table as :

EMP (EmpNo, EName, Post, Salary, Dept No) with few records
Now write a PL|SQL block to take employee No as input, fetch the salary of that employee and increase the salary by 2000 if salary <= 20,000 Otherwise, increase the salary by 5,000.

Program :

```
        DECLARE
                E no VARCHAR(10);
                Sal FLOAT;
               new_sal FLOAT;
                BEGIN
                e no : = & eno ;  OR eno : = : eno ; // To
  Take input from the user
```

Types of Cursor:

Cursor are Classified depending on the situations under which they are opened. There are two types of cursors as:

1) Implicit Cursor.
2) Explicit ~~ca~~ Cursor.

Implicit Cursor

→ This cursor is opened by Oracle engiene for it's internal processing, everytime when an SQL statement is processed. Since, the implicit cursor is opened and manged by oracle engiene internally, the functions of reserving an area in memory, populating this area with appropriate data, processing the data in memory area and releasing the memory area when the processing is complete is taken care by oracle engiene itself.

Attributes of Implicit Cursor

1) %ISOPEN	3) %NOTFOUND
2) %FOUND	4) %ROWCOUNT.

Implicit Cursor have the following attributes:

1) %ISOPEN: It returns true if cursor is open. Otherwise false. It is written as SQL%ISOPEN.

2) %FOUND : It evaluate true if an INSERT, DEL
UPDATE statement affect at least one record o
false. It is written as SQL%FOUND.

3) %NOTFOUND : It is the logical opposite of %FOUND
it evaluates true if insert, update and delete
doesnot affect any record. Otherwise, false. It is
written as SQL%NOTFOUND.

4) %ROWCOUNT : It returns the number of rows affe
by INSERT, UPDATE, DELETE or SELECT statement. It
written as SQL % ROWCOUNT.

For eg : A PL/SQL Block to change the department
number of a particular employee in Emp Table

Emp (EmpNo, E.Name, Post, Salary, ... dept no)

```
DECLARE
        eno Emp.EmpNo % TYPE ;
        dno Emp. DeptNo % TYPE;
BEGIN
        eno := : eno;
        d.no := . dno;
UPDATE Emp SET Dept no = d.no
WHERE Emp.No = e no ;
If SQL % FOUND THEN
dbms.output.put-line ("Employee Transfered Sucessfully
    ELSE
      dbms-output.put_line('Employee doesnot exist');
    END IF;
     END;
```

g:- A PL/SQL Block to input department number and increase the salary by 5,000 for the employee of given department number and counting the number of records affected.

```
DECLARE
        d.no Emp.Dept no % TYPE;
        rows VARCHAR 2 (5);
   BEGIN
           d.no := :dno;
   UPDATE Emp SET Salary = Salary + 5,000
     WHERE Dept.no = dno;
      rows = TO-CHAR(SQL % ROWCOUNT);
  IF SQL % ROWCOUNT >0 THEN
    dbms-output.put-line (rows || 'Rows affected');
 ELSE
   dbms-output.put-line ('No Employee in given departme
   END IF;
   END;
```

Unit-4.

Using PL/SQL Subprograms.

Subprogram

→ A Sub program is a logical group of SQL and PL/SQL statements that perform a specific task.

→ These sub programs are combined to form layer programs.

→ This is actually modular program design where a large and complex program is decomposed into multiple smaller and simple sub programs or modules.

→ In PL/SQL sub programs can be created as procedure and function.

→ These sub programs can be called by passing a set of parameters.

→ A PL/SQL sub program (either procedure or function) is made up of three parts as:

1) Declarative part.
2) Executable part.
3) Exception Handling (optional).

1) Declarative part:

→ The declarative part may contain the declaration of variables, constants, cursor, exception and other sub programs.

→ These part is optional and does not start with the keyword 'declare.'

#Executable Part :
→ This is the compulsory part and contains all SQL and PL/SQL statements necessary perform the desire task.

#Exception Handling Part: (optional)
→ This is an optional part and it contains the code that handles runtime errors of executable

Assignment
→ Write down the advantages of using subprograms PL/SQL.

① Improved Performance: The amount of information that application must send over a network in Small compa with issuing individual SQL statements or sending th text of an entire PL/SQL block to oracle database, because the information is sent only once and ~~therefore~~ thereafter invoked when it is used.

- The compiled form of a procedure is readily availab the database. So, no compilation is required at execu time.

* Memory allocation:

Because stored procedures take advantage of the shared memory capabilities of oracle database, it must load only a single copy of the procedure into memory for execution by multiple users sharing code among users results in a substantial reduction in database memory requirements for applications.

Passing Parameters: In PL/SQL we can pass Paramete
different ways as:

① Positional method

→ In positional notation, actual paramet
in function / procedure call must be in the Same or
function or procedure definition. They must have ide
datatypes. We cannot change the order of actual
formal arguments.

For Example:

```
        CREATE OR REPLACE FUNCTION fun (x Number,
Number, Z Number)
            END;
        function calling block
          BEGIN
            . . . .
            . . . .

      ~~function calling block~~
      ~~BEG~~   fun (b, c, a);
            . . . .

            END;
```

In this case, the values of actual argument b, c and
are copied to the formal arguments in the same or

② Named method:

→ In this method we use an array operat
while calling the function or procedure to pass the
parameters. In this method, there is no fixed positi
pass the parameters.

For example:

```
CREATE OR REPLACE FUNCTION fun(x Number, y Number,
Z Number):
      . . . . .
      . . . . .

      END;
    function calling block;
    BEGIN . .
        . . .

      fun(z ⇒ a, x ⇒ b, y ⇒ c);
       END;
```

(3) Mixed Method:

→ In this approach few parameters are passed as positional and few using named method.

```
    CREATE or REPLACE FUNCTION fun(x Number, y
Number, Z Number).
        . . . . .

        END;
        function calling block;
      BEGIN
        fun(a, z ⇒ b, y ⇒ c); // Legal
        fun(x ⇒ b, y ⇒ a, c); // illegal
        . . . .

      END;
```

In mixed method parameters on left of function call must be positional. So, we cannot write initial parameters as named followed by positional paramet

Function :

A PL/SQL function is same as procedure except that it returns a value to the calling block. It is mainly used to accept arguments, to perform certain calculations and return a single value at a time.

A function is created as :

```
    CREATE OR REPLACE FUNCTION function-name
  (argument [IN] Datatype .....)
   RETURN Datatype
  {IS, AS}
     Variable declaration;
       ........
   BEGIN
    PL/SQL Subprogram body ;
     EXCEPTION
   PL/SQL Exception Handling block;
     END;
```

where,
RETURN specifies the type of data returned by the function.

Private and Public objects in a Package

The subprograms cursors, variables and constants declared inside the package body are private. Such private body are private. Such private objects are only accessible within the same package and cannot be accessed in PL/SQL code outside the package.

Any subprograms, cursors, variables and constants declared within package specification are the public objects and become visible inside the same package as well as outside the package to access them outside the packages, we use

Packagename.objectName;

Overloading Procedure & function

In oracle package we can declare and define more than one functions or procedure with the same name but with different parameters or their datatypes. This concept is known as overloading functions or procedures. This ~~concept~~ is ~~known~~ as overloading approach simplifies the PL/SQL code blocks by providing similar name to a class of functions or procedure that can perform similar types of task but on different types of arguments.

Overview of Product Specific Packages

Oracle and various oracle tools are supplied with product specific packages that defines API's and we can call from PL/SQL JAVA, and other programming environment. Few commonly used packages are:

① DBMS_ALERT:

This package allows database triggers to call an application or errors when specific database value changed. The alerts are transaction based and they operate independently. This package includes various methods to deal with different alert messages that may arise while performing or executing transactions.

② DBMS_OUTPUT:-

This package allows us to display output from PL/SQL block and subprogram, which makes it easier to debug and test the PL/SQL blocks. The method put.line() is used with this package for displaying some message.

③ DBMS-PIPE:-

A pipe is an area of memory used by one process to pass information to other processes. This package allows different sessions to communicate over diffrent pipes. This package is useful in various operating systems to interact with multiple high level language programs.

④ UTL-FILE :

This package allows PL/SQL program read and write operating systems text file. provides different file related operations like o PUT(), GET(), CLOSE(), etc to interact u files for reading and writing operations.

⑤ URL-HTTP :

This package allows PL/SQL programs make hypertext transfer protocol call outs it can retrieve data from the web page Oracle web server as well as can provide to them. It exchanges data in the form of

⑥ UTL-SMTP :

This package allows PL/SQL programs to interact with e-mail services servers for variou e-mail related operations.

⑦ HTF and HTP :

Hypertext procedure (HTP) and Hyp function (HTF) packages generate HTML tags. f HTP procedure that generate HTML tags, there corresponding HTF functions with identical param