Experiment No. 9

Environment: Microsoft Windows
Tools/ Language: Oracle

Objective:- To implement the basics of PL/SQL.

Introduction – PL/SQL bridges the gap between database technology and procedural programming languages. It can be thought of as a development tool that extends the facilities of Oracles SQL database language. Via PL/SQL you can insert, delete, update and retrieve table data as well as use procedural techniques such as writing loops or branching to another block of code.

PL/SQL Block structure-

DECLARE
Declarations of memory variables
BEGIN
SQL executable statements for manipulating table data.
EXCEPTION
SQL and/or PL.SQL code to handle errors.
END;

Displaying user Messages on the screen –Any programming tool requires a method through which messages can be displayed to the user.

Dbms_output: is a package that includes a number of procedure and functions that accumulate information in a buffer so that it can be retrieved later. These functions can also be used to display message to the user.
put_line: put a piece of information in the buffer followed by an end of line marker. It can also be used to display message to the user.

SET SERVER OUTPUT ON: Setting the server output on.

Example: Write the following code in the PL/SQL block to display message to user
DBMS_OUTPUT.PUT_LINE('Display user message');

Conditional control in PL/SQL-
Syntax:
IF <condition> THEN
<Action>
ELSEIF<condition>
<Action>
ELSE
<Action>
ENDIF;

The WHILE LOOP:
Syntax:
WHILE <condition>
LOOP
<Action>
END LOOP;

The FOR LOOP statement:
Syntax:
FOR variable IN [REVERSE] start—end
LOOP
<Action>
END LOOP;

Cursor:

A cursor is a temporary work area created in the system memory when a SQL statement is executed. A cursor contains information on a select statement and the rows of data accessed by it.

There are two types of cursor in PL/SQL:
1. Implicit cursors: These are created by default when DML statements like, INSERT, UPDATE, and DELETE statements are executed. They are also created when a SELECT statement that returns just one row is executed.
Following are the cursor attributes available:

- **SQL%ROWCOUNT** – Number of rows returned/changed in the last executed query. Applicable for SELECT as well as DML statement
- **SQL%ISOPEN** – Boolean TRUE if the cursor is still open, else FALSE. For implicit cursor it is FALSE only
- **SQL%FOUND** – Boolean TRUE, if the cursor fetch points to a record, else FALSE
- **SQL%NOTFOUND** – Inverse of SQL%FOUND. The flag is set as FALSE when the cursor pointer does not point to a record in the result set.

2. Explicit cursors: They must be created when you are executing a SELECT statement that returns more than one row. Even though the cursor stores multiple records, 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.
Both implicit and explicit cursors have the same functionality, but they differ in the way they are accessed.

General Syntax for creating an explicit cursor is as given below:

```
CURSOR cursor_name IS select_statement;
```

How to access an Explicit Cursor?
These are the three steps in accessing the cursor.
1) Open the cursor.
2) Fetch the records in the cursor one at a time.
3) Close the cursor.

General Form of using an explicit cursor is:

```
DECLARE
   variables;
   records;
   create a cursor;
BEGIN
  OPEN cursor;
  FETCH cursor;
    process the records;
  CLOSE cursor;
END;
```

PL/SQL EXAMPLES:

1: Print

```
set serveroutput on;
BEGIN
dbms_output.put_line('Welcome');
END;
```

Output:

Welcome

2: For LOOP with IF

```
set serveroutput on;
DECLARE
i int;
n int;
BEGIN
n:=&n;
FOR i IN 1..n
Loop
IFmod(i,2) = 1
THEN
dbms_output.put_line('Welcome'||i);
END IF;
END Loop;
END;
```

Output:

```
Enter value for n: 5
Welcome1
Welcome3
Welcome5
```

3: For LOOP with IF ELSE

```
set serveroutput on;
DECLARE
i int;
n int;
BEGIN
n:=&n;
FOR i IN 1..n
Loop
IFmod(i,2) = 1
THEN
dbms_output.put_line('Wekcome'||i);
ELSE
dbms_output.put_line('PL/SQL');
END IF;
END Loop;
END;
```

Output:

```
Enter value for n: 5
Welcome1
PL/SQL
Welcome3
PL/SQL
Welcome5
```

4: basic LOOP

```
set serveroutput on;
DECLARE
n int;
BEGIN
N:=&n;
LOOP
dbms_output.put_line(‘Hi’);
if n<2
THEN dbms_output.put_line(‘Less than 2’);
elsif n>2
THEN dbms_output.put_line(‘Greater than 2’);
else
dbms_output.put_line(‘Zero’);
END IF;
n:=n-1;
EXIT WHEN n=0;
END LOOP;
END;
```

Output:

```
Enter value for n: 5
Hi
Greater than 2
```

Hi
Greater than 2
Hi
Greater than 2
Hi
Zero
Hi
Less than 2

5: While LOOP

```
set serveroutput on;
DECLARE
        n int;
        i int;
BEGIN
        n:= &n;
        WHILE n>=1 LOOP
                dbms_output.put_line('The value of a is '||n);
        n:= n-1;
END LOOP;
END;
```

Output:
Enter value for n: 5
The value of a is 5
The value of a is 4
The value of a is 3
The value of a is 2
The value of a is 1

Practical Assignment - 9

Department: Computer Engineering & Applications

Course: B.Tech. (CSE)

Subject: Database Management System Lab (ITE291)

Year: 2nd **Semester:**3rd

SQL Script for this experiment:

```
Drop table Student;
create table Student(sID number(5), sName varchar2(10), GPA number(3,1), sizeHS number(5));
insert into Student values (123, 'Amy', 3.9, 1000);
insert into Student values (234, 'Bob', 3.6, 1500);
insert into Student values (345, 'Craig', 3.5, 500);
insert into Student values (456, 'Doris', 3.9, 1000);
insert into Student values (567, 'Edward', 2.9, 2000);
insert into Student values (678, 'Fay', 3.8, 200);
insert into Student values (789, 'Gary', 3.4, 800);
insert into Student values (987, 'Helen', 3.7, 800);
insert into Student values (876, 'Irene', 3.9, 400);
insert into Student values (765, 'Jay', 2.9, 1500);
insert into Student values (654, 'Amy', 3.9, 1000);
insert into Student values (543, 'Craig', 3.4, 2000);
```

1. Write a PL/SQL code block to compute the factorial of a number.
2. Write a PL/SQL code block to determine whether the number is prime or not.
3. Write a PL/SQL code block to display n terms of a fibonacci series.
4. Write a PL/SQL code block to display the names and GPA of students from student table using an explicit cursor.
5. Write a PL/SQL code block that displays the names, GPA of students along with the grades of students after calculation from student table using an explicit cursor.
   Add a column grade to the student table; update the grades of students to the table after calculation. (The criteria of grade can be considered as grade = A if gpa>3.7; and grade = B, otherwise).

Pre-Experiment Question:

1. What are advantages of using PL/SQL in DBMS?
2. What are types of cursor?
3. How a record is declared in a PL/SQL block. Give syntax.

Post-Experiment Question:

1. What are the attributes of a cursor?
2. What are the advantages of using an explicit cursor