Database Session

Lecture - 6

Introduction To PL/SQL

The PL/SQL programming language was developed by Oracle Corporation in the late 1980s as procedural extension language for SQL and the Oracle relational database. Following are certain notable facts about PL/SQL −

- ❖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.
- ❖Direct call can also be made from external programming language calls to database.

PL/SQL Blocks

Every PL/SQL statement ends with a semicolon (;). PL/SQL blocks can be nested within other PL/SQL blocks using BEGIN and END. Following is the basic structure of a PL/SQL block −

```
DECLARE
<declarations section>
BEGIN
<executable command(s)>
EXCEPTION
<exception handling>
END;
```

Hello World Example

```
DECLARE

message varchar2(20):= 'Hello, World!';

BEGIN

dbms_output.put_line(message);

END;

/
```

The above code will display Hello, World! Message.

Simple Calculator Code In PL/SQL

```
DECLARE
a int;
b int;
BEGIN
a:=&a;
b:=&b;
dbms_ouput.put_line(‘Summation=’||(a+b));
dbms_ouput.put_line(‘Subtraction=’||(a-b));
dbms_ouput.put_line(‘Multiplication=’||(a*b));
dbms_ouput.put_line(‘Division=’||(a/b));
END;
/
```

Syntax Of If & If Else Statements

Syntax of If Statement:-

```
If condition then
/* If block code*/
End if;
```

Syntax of If Else Statement:-

```
If  condition then
/* If block code */
Else
/* Else block code*/
End if;
```

Example Application - 1

Develop a PL/SQL code to check given no. is even or odd.

```
DECLARE
n int;
BEGIN
n:=&n;
If n mod 2==0 then
dbms_output.put_line('The number is even');
Else
dbms_output.put_line('The number is odd');
End if;
END;
/
```

Syntax Of Ladder If Else

```
If condition1 then

/* Code 1 */

Elsif condition2 then

/* Code 2 */

Elsif condition3 then

/* Code 3*/

Else

/* Code 4 */

End If;
```

Example Application - 2

/*

Write a PL/SQL code to make a electricity bill calculator.

Unit	Bill
1-150	2.40/unit
For next 151-300	3.00/unit
For next more than 300	3.20/unit

*/

Example Application – 2 (cont..)

```
DECLARE
unit number(5,2);
bill number(10,2);
BEGIN
unit:=&unit;
If unit<=150 then
bill:=unit*2.40;
Elsif unit>150 and unit<=300 then
bill:=(150*2.40)+(unit-150)*3.00;
Else
bill:=(150*2.40)+(150*3.00)+(unit-300)*3.20;
End if;
dbms_output.put_line('Your bill is : '||bill);
END;
/
```

Loops In PL/SQL

PL/SQL Basic Loop:- In this loop structure, sequence of statements is enclosed between the LOOP and the END LOOP statements. At each iteration, the sequence of statements is executed and then control resumes at the top of the loop.

PL/SQL While Loop:- Repeats a statement or group of statements while a given condition is true. It tests the Condition before executing the loop body.

PL/SQL For Loop:- Execute a sequence of statements multiple times and abbreviates the code that manages the loop variable.

PL/SQL Basic Loop

Syntax of PL/SQL basic Loop:-

```
Loop

/* Code */

End Loop;
```

Example Application - 3

Develop a PL/SQL code to print 1-5 numbers using basic loop.

```
declare
a number(5):=1;
begin
dbms_output.put_line(‘Program started’);
Loop
dbms_output.put_line(a);
a:=a+1;
Exit when a>5;
End loop;
dbms_output.put_line(‘Program completed’);
End;
/
```

PL/SQL While Loop

It works like an entry-check loop in which execution block will not even be executed once if the condition is not satisfied, as the exit condition is checking before execution part. It does not require keyword 'EXIT' explicitly to exit from the loop since it is validating the condition implicitly each time of the loop.

Syntax Of While Loop:-

```
WHILE <EXIT condition>
LOOP
<execution block starts>
.
.
.
<execution_block_ends>
END LOOP;
```

Example Application - 4

```
DECLARE
a NUMBER :=1;
BEGIN
dbms_output.put_line('Program started');
WHILE (a <= 5)
LOOP
dbms_output.put_line(a);
a:=a+1;
END LOOP;
dbms_output.put_line(‘Program completed' );
END;
/
```

PL/SQL For Loop

"FOR LOOP" statement is best suitable when you want to execute a code for a known number of times rather than based on some other conditions.
In this loop, the lower limit and the higher limit will be specified and as long as the loop variable is in between this range, the loop will be executed.

```
FOR <loop_variable> in <lower_limit> .. <higher_limit>
LOOP <execution block starts>
.
.
.
<execution_block_ends>
END LOOP;
```

Example Application - 5

```
BEGIN
dbms_output.put_line(‘Program started.' );
FOR a IN 1 .. 5 LOOP
dbms_output.put_line(a);
END LOOP;
dbms_output.put_line('Program completed.');
END;
/
```