PL/SQL Loop (Iterative Statements)

The PL/SQL loops are used to repeat the execution of one or more statements for specified number of times. These are also known as iterative control statements.

Syntax for a basic loop:

```
LOOP
  Sequence of statements;
END LOOP;
```

Types of PL/SQL Loops

There are 4 types of PL/SQL Loops.

1. Basic Loop
2. Exit Loop
3. While Loop
4. For Loop

PL/SQL Exit Loop (Basic Loop)

PL/SQL exit loop is used when a set of statements is to be executed at least once before the termination of the loop. There must be an EXIT condition specified in the loop, otherwise the loop will get into an infinite number of iterations. After the occurrence of EXIT condition, the process exits the loop.

Syntax of basic loop:

```
LOOP
  Sequence of statements;
END LOOP;
```

Syntax of exit loop:

```
LOOP
   statements;
   EXIT;
```

```
  {or EXIT WHEN condition;}
END LOOP;
```

Example of PL/SQL EXIT Loop

Let's take a simple example:

```
DECLARE
i NUMBER := 1;
BEGIN
LOOP
EXIT WHEN i>10;
DBMS_OUTPUT.PUT_LINE(i);
i := i+1;
END LOOP;
END;
```

After the execution of the above code, you will get the following result:

```
1
2
3
4
5
6
7
8
9
10
```

Note: You must follow these steps while using PL/SQL Exit Loop.

- Initialize a variable before the loop body
- Increment the variable in the loop.
- You should use EXIT WHEN statement to exit from the Loop. Otherwise the EXIT statement without WHEN condition, the statements in the Loop is executed only once.

PL/SQL EXIT Loop Example 2

```
DECLARE
VAR1 NUMBER;
```

```
VAR2 NUMBER;
BEGIN
VAR1:=100;
VAR2:=1;
LOOP
DBMS_OUTPUT.PUT_LINE (VAR1*VAR2);
IF (VAR2=10) THEN
EXIT;
END IF;
VAR2:=VAR2+1;
END LOOP;
END;
```

Output:

```
100
200
300
400
500
600
700
800
900
1000
```

PL/SQL While Loop

PL/SQL while loop is used when a set of statements has to be executed as long as a condition is true, the While loop is used. The condition is decided at the beginning of each iteration and continues until the condition becomes false.

Syntax of while loop:

```
WHILE <condition>
 LOOP statements;
END LOOP;
```

Example of PL/SQL While Loop

Let's see a simple example of PL/SQL WHILE loop.

```
DECLARE
i INTEGER := 1;
BEGIN
WHILE i <= 10 LOOP
DBMS_OUTPUT.PUT_LINE(i);
i := i+1;
END LOOP;
END;
```

After the execution of the above code, you will get the following result:

```
1
2
3
4
5
6
7
8
9
10
```

Note: You must follow these steps while using PL/SQL WHILE Loop.

- Initialize a variable before the loop body.
- Increment the variable in the loop.
- You can use EXIT WHEN statements and EXIT statements in While loop but it is not done often.

PL/SQL WHILE Loop Example 2

```
DECLARE
VAR1 NUMBER;
VAR2 NUMBER;
BEGIN
VAR1:=200;
VAR2:=1;
WHILE (VAR2<=10)
LOOP
```

```
DBMS_OUTPUT.PUT_LINE (VAR1*VAR2);
VAR2:=VAR2+1;
END LOOP;
END;
```

Output:

```
200
400
600
800
1000
1200
1400
1600
1800
2000
```

PL/SQL FOR Loop

PL/SQL for loop is used when when you want to execute a set of statements for a predetermined number of times. The loop is iterated between the start and end integer values. The counter is always incremented by 1 and once the counter reaches the value of end integer, the loop ends.

Syntax of for loop:

```
FOR counter IN initial_value .. final_value LOOP
  LOOP statements;
END LOOP;
```

- initial_value : Start integer value
- final_value : End integer value

PL/SQL For Loop Example 1

Let's see a simple example of PL/SQL FOR loop.

```
BEGIN
FOR k IN 1..10 LOOP
-- note that k was not declared
```

```
DBMS_OUTPUT.PUT_LINE(k);
END LOOP;
END;
```

After the execution of the above code, you will get the following result:

```
1
2
3
4
5
6
7
8
9
10
```

Note: You must follow these steps while using PL/SQL WHILE Loop.

- You don't need to declare the counter variable explicitly because it is declared implicitly in the declaration section.
- The counter variable is incremented by 1 and does not need to be incremented explicitly.
- You can use EXIT WHEN statements and EXIT statements in FOR Loops but it is not done often.

PL/SQL For Loop Example 2

```
DECLARE
VAR1 NUMBER;
BEGIN
VAR1:=10;
FOR VAR2 IN 1..10
LOOP
DBMS_OUTPUT.PUT_LINE (VAR1*VAR2);
END LOOP;
END;
```

Output:

```
10
20
30
```

```
40
50
60
70
80
90
100
```

PL/SQL For Loop REVERSE Example 3

Let's see an example of PL/SQL for loop where we are using REVERSE keyword.

```
DECLARE
VAR1 NUMBER;
BEGIN
VAR1:=10;
FOR VAR2 IN REVERSE 1..10
LOOP
DBMS_OUTPUT.PUT_LINE (VAR1*VAR2);
END LOOP;
END;
```

Output:

```
100
90
80
70
60
50
40
30
20
10
```

PL/SQL Continue Statement

The continue statement is used to exit the loop from the reminder if its body either conditionally or unconditionally and forces the next iteration of the loop to take place, skipping any codes in between.

The continue statement is not a keyword in Oracle 10g. It is a new feature encorporated in oracle 11g.

For example: If a continue statement exits a cursor FOR LOOP prematurely then it exits an inner loop and transfer control to the next iteration of an outer loop, the cursor closes (in this context, CONTINUE works like GOTO).

Syntax:

```
continue;
```

Example of PL/SQL continue statement

Let's take an example of PL/SQL continue statement.

```
DECLARE
  x NUMBER := 0;
BEGIN
  LOOP -- After CONTINUE statement, control resumes here
    DBMS_OUTPUT.PUT_LINE ('Inside loop:  x = ' || TO_CHAR(x));
    x := x + 1;
    IF x < 3 THEN
      CONTINUE;
    END IF;
    DBMS_OUTPUT.PUT_LINE
      ('Inside loop, after CONTINUE:  x = ' || TO_CHAR(x));
    EXIT WHEN x = 5;
  END LOOP;

  DBMS_OUTPUT.PUT_LINE (' After loop:  x = ' || TO_CHAR(x));
END;
/
```

After the execution of above code, you will get the following result:

```
Inside loop:  x = 0
Inside loop:  x = 1
Inside loop:  x = 2
Inside loop, after CONTINUE:  x = 3
Inside loop:  x = 3
Inside loop, after CONTINUE:  x = 4
Inside loop:  x = 4
Inside loop, after CONTINUE:  x = 5
After loop:  x = 5
```