GALGOTIAS UNIVERSITY

School of Computing Science and Engineering

Program: BCA - IOP

Course Code: BCAS3031

Course Name: PL/SQL & Cursors and Triggers

Dr. T. Poongodi

Associate Professor

Simple Loop Statement, While Loop Statement, For Loop Statement, The Continue Statement

Looping:

Repeating a particular part of any program or any code statement as many times as required.

In PL/SQL, there are three different loop options,

- Basic Loop
- While Loop
- For Loop

PL/SQL Basic Loop

Basic loop or simple loop is preferred in PL/SQL code when there is no surety about how many times the block of code is to be repeated.

Basic loop: The code block will be executed at least once.

- Simple loop always begins with the keyword `LOOP` and ends with a keyword `END LOOP`.
- A basic/simple loop can be terminated at any given point by using the `exit` statement or by specifying certain condition by using the statement `exit when.`

There must be at least one executable statement between LOOP and END LOOP keywords.
The sequence of statements is executed repeatedly until it reaches a loop exit.

PL/SQL provides EXIT and EXIT-WHEN statements to allow you to terminate a loop.

- The EXIT forces the loop halts execution unconditionally and passes control to the next statement after the LOOP statement. You typically use the EXIT statement with the IF statement.

- The EXIT-WHEN statement allows the loop to terminate conditionally.
- When the EXIT-WHEN statement is reached, the condition in the WHEN clause is checked.
- If the condition evaluates to TRUE, the loop is terminated and control is passed to the next statement after the keyword END LOOP.
- If the condition evaluates to FALSE, the loop will continue repeatedly until the condition evaluates to TRUE.
- Do something inside the loop to make condition becomes TRUE

```
LOOP
     sequence of statements
END LOOP;
```

Sample Code:
Loop to print counting from 1 to 10 on the console and have used the `exit` statement to break out of the loop.

```
DECLARE
     i int;
BEGIN
     i := 1;
     LOOP
          if i>10 then
               exit;
          end if;
     dbms_output.put_line(i);
          i := i+1;
     END LOOP;
END;
```

1 2 3 4 5 6 7 8 9 10

PL/SQL procedure successfully completed

using the `exit when` statement to break out of the loop.

```
DECLARE
     i int;
BEGIN
     i := 0;
     LOOP
          i := i+2
          dbms_output.put_line(i);
          exit WHEN x > 10
     END LOOP;
END;
```

2 4 6 8 10

PL/SQL procedure successfully completed

PL/SQL: While Loop

- It is an entry controlled loop which means that before entering in a while loop first the condition is tested, if the condition is **TRUE** the statement or a group of statements get executed and if the condition is **FALSE** the control will move out of the while loop.

```
WHILE <test_condition> LOOP
    <action>
END LOOP;
```

To print the odd numbers between 1 to 10 using the while loop.

```
DECLARE
      num int:=1;
BEGIN
      while(num <= 10) LOOP
            dbms_output.put_line(''|| no);
            num := num+2;
END LOOP;
END;
```

1 3 5 7 9

PL/SQL procedure successfully completed.

Write the code to increment the value of the variable that you put in the condition otherwise the value of the variable will remain the same and the condition will always remain true.

PL/SQL: For Loop

- This loop is used when some statements in PL/SQL code block are to be repeated for a fixed number of times.
- When we use the for loop, define a counter variable which decides how many time the loop will be executed based on a starting and ending value provided at the beginning of the loop.
- The for loop automatically increments the value of the counter variable by 1 at the end of each loop cycle.
- The programmer need not have to write any instruction for incrementing or decrementing value.

for loop to print numbers from 1 to 10

```
DECLARE
      i number(2);
BEGIN
      FOR i IN 1..10 LOOP
            dbms_output.put_line(i);
      END LOOP;
END;
```

1 2 3 4 5 6 7 8 9 10

PL/SQL procedure successfully completed.

Print the number in reverse order.

```
DECLARE
     i number(2);
BEGIN
     FOR i IN REVERSE 1..10 LOOP
          dbms_output.put_line(i);
     END LOOP;
END;
```

10 9 8 7 6 5 4 3 2 1

PL/SQL procedure successfully completed.

PL/SQL CONTINUE Statement (Sequential Control Statements) are control your iteration loop, PL/SQL CONTINUE Statement to skip the current iteration with in loop.

- CONTINUE Statement : to skip the current iteration with in loop.
- CONTINUE WHEN Statement : to skip the current iteration with in loop when WHEN clauses condition true.

CONTINUE Statement

CONTINUE Statement unconditionally skip the current loop iteration and next iteration iterate as normal, only skip matched condition.

```
IF condition THEN
    CONTINUE;
END IF;
```

```
DECLARE
      no NUMBER := 0;
BEGIN
      FOR no IN 1 .. 5 LOOP
            IF i = 4 THEN
                  CONTINUE;
      END IF;
      DBMS_OUTPUT.PUT_LINE('Iteration : ' ||
no);
      END LOOP;
END;
/
```

Iteration # 1
Iteration # 2
Iteration # 3
Iteration # 5

PL/SQL procedure successfully completed

CONTINUE WHEN Statement unconditionally skip the current loop iteration when WHEN clauses condition true,

```
CONTINUE WHEN condition;
statement(s);
```

```
DECLARE
      no NUMBER := 0;
BEGIN
      FOR no IN 1 .. 5 LOOP
            DBMS_OUTPUT.PUT_LINE('Iteration : ' ||
no);         CONTINUE WHEN no = 4
                  DBMS_OUTPUT.PUT_LINE('CONTINUE WHEN
EXECUTE Iteration : ' || no);
      END LOOP;
END;
/
```

Iteration # 1
Iteration # 2
Iteration # 3
CONTINUE WHEN EXECUTE Iteration : 4
Iteration # 5

PL/SQL procedure successfully completed.


Thank You