Jump2Learn
PUBLICATION

www.jump2learn.com

CONCEPTS *of* RELATIONAL DATABASE MANAGEMENT SYSTEM

Jump2Learn - The Online Learning Place

Dr. RajeshKumar R. Savaliya | Ms. Sonal B. Shah | Mr. Vaibhav D. Desai

Unit 4 Iterative Statements

Title	P.No.
4.1 Introduction – Iterative Statements	03
4.1.1 Loop... End Loop	04
4.1.2 For... Loop	10
4.1.3 While Loop	12
4.1.4 EXIT Loop	14
4.1.5 Continue	14

4.1 Introduction

Iterative statements are the statement(s) which are executed zero or more times. The same set of statements are to be executed zero or more times.

If we want to understand iterative statement in day to day life, remember the instruction our English teachers used to give us during junior school: "Repeat spelling of bat five times". And we used to say

"B, A, T – Bat"
"B, A, T – Bat"
"B, A, T – Bat"
"B, A, T – Bat"
"B, A, T – Bat"

In above example the statement being repeatedly spoken (executed)is : "B, A, T – Bat" and number of times this statement spoken (executed) is, five times.

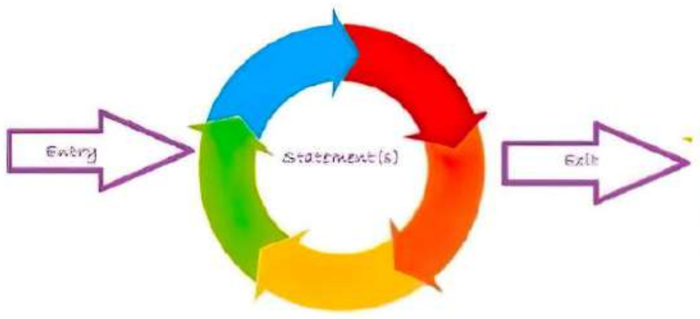

Fig 4.1 What is Iterative Statement?

In figure 4.1, the circle of arrows in the middle represents the same set of statements being executed multiple times. There is a specific point where an entry is made to these iterative statements and on a particular situations or conditions the execution of iterative statements are stopped or exited.

There are two types of iterative statements:

1. Count controlled

In this type of iterative statements, the number of times the statements are to be executed is fixed before starting execution of the iterative statements. Like in our earlier example, it was fixed to repeat spelling of BAT five times. Such type of iterative statements are relatively simple.

2. Condition controlled

In this type of iterative statements, the number of times the statements are to be executed is not really fixed, but its end of execution is based on a certain condition. As soon as this condition is satisfied, the execution of iterative statements is stopped or exited. This condition is known as exit condition. For example, repeat saying spelling of BAT until the teacher returns to the classroom. Now the spelling of BAT will be repeated for variable time based on the condition "When teacher returns to the classroom". If the teacher returns early, the spelling will be repeated lesser number times. But if the teacher returns late the spelling will be repeated more number of times. This totally depends on the condition, when the teacher returns to the class room.

We now have clear idea about what iterative statements are. Let us now discuss about various iterative statements provided by Oracle PL/SQL.

4.1.1 Loop..End Loop

"Loop-End Loop" is the most basic and common iterative statement provided by PL/SQL. This loop continues its execution unless it encounters statement "EXIT". It is logically mandatory to write "EXIT" statement inside this type of loop, otherwise the loop will get into an infinite number of iterations i.e. it will never come out of the loop.

The syntax of "Loop-End Loop" statement is as follows:

```
LOOP
      Statement(s);
      EXIT;
      { or EXIT WHEN condition; }
END LOOP;
```

Following is the flow diagram of "LOOP – END LOOP" statement.

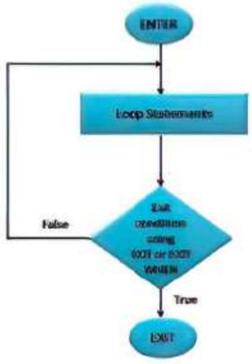

Fig 4.2 LOOP-END LOOP Flow Diagram

Let us take a very simple example of printing numbers between 1 to 10.

The PL/SQL code for the same is as follows (assuming that we have written the PL/SQL code in SQL script named "TEST41.SQL" using NOTEPAD or any other editor):

```
SET SERVEROUTPUT ON;
EXECUTE DBMS_OUTPUT.ENABLE(1000);

DECLARE
  N NUMBER(2) ;
BEGIN
 N := 1;
  LOOP
    DBMS_OUTPUT.PUT_LINE('CURRENT ITERATION IS : '||N);
    N := N + 1;
    IF ( N > 10) THEN
       EXIT;
    END IF;
  END LOOP;
END;
/
```

(TEST41.SQL)

Output of above program is:

```
SQL> @41


PL/SQL procedure successfully completed.


CURRENT ITERATION IS : 1

CURRENT ITERATION IS : 2

CURRENT ITERATION IS : 3

CURRENT ITERATION IS : 4

CURRENT ITERATION IS : 5

CURRENT ITERATION IS : 6

CURRENT ITERATION IS : 7

CURRENT ITERATION IS : 8

CURRENT ITERATION IS : 9

CURRENT ITERATION IS : 10


PL/SQL procedure successfully completed.


SQL>
```

In above example the "LOOP-END LOOP" is having "EXIT" statement which ensures the termination of loop after desired number of iterations i.e. 10 iterations in our case. Following points are to be understood:

1. A variable is to be initialized before loop body starts.

We have declared variable "N" in declaration section of PL/SQL and it is initialized by value 1 before "Loop" statement.

2. There must be an increment/decrement of this variable inside the loop;

In our case, we have incremented the value of N by 1 in every iterations using "N := N +1;" statement.

3. The loop must consist "EXIT" statement. If EXIT statement is written without any conditions, the loop will execute only once i.e. only single iteration will be executed.

We have written "EXIT" statement inside "IF..END IF" statement in our loop, which will be executed only of the value of N is greater than 10. This means that our loop will be exited only when value of N exceeds 10. Till such point of time the loop will keep on executed.

Note: TEST41.SQL is started with two statements before DECLARE statement, which are used for displaying the output on standard output using "DBMS_OUTPUT.PUT_LINE" statement.

SET SERVEROUTPUT ON;

The SET SERVEROUTPUT command is used to display the content of the Oracle buffer into your screen. The content is displayed on screen using DBMS_OUTPUT.PUT_LINE function.

DBMS_OUTPUT.ENABLE(1000);

It defines buffer of size of 1000 bytes. This buffers used to display the content on screen.

DBMS_OUTPUT.PUT_LINE (*Text*);

This function is used to display the content passed by an incoming parameter (Text) on the screen.

Now , we will understand "EXIT WHEN..." statement for the same example.

```
SET SERVEROUTPUT ON;
EXECUTE DBMS_OUTPUT.ENABLE(1000);

DECLARE
  N NUMBER(2) ;
BEGIN
  N := 1;
  LOOP
   DBMS_OUTPUT.PUT_LINE('CURRENT ITERATION IS : '||N);
   N := N + 1;
    EXIT WHEN N>10;
  END LOOP;
END;
/
```

(TEST42.SQL)

Output of TEST42.SQL is same as output of TEST41.SQL. The loop will be continued for iterations until value of N exceeds 10 i.e. the condition return after EXIT WHEN returns TRUE.

Oracle PL/SQl supports nested loops also. Nested loop means a loop inside loop.

Syntax of the same is:

```
LOOP
        Statement(s);
        LOOP
                Statement(s);
                EXIT;
                { or EXIT WHEN condition; }
        END LOOP;
        EXIT;
        { or EXIT WHEN condition; }
END LOOP;
```

Let us extend our example for nested loop scenario.

```
SET SERVEROUTPUT ON;
EXECUTE DBMS_OUTPUT.ENABLE(1000);

DECLARE
  N NUMBER(2) ;
  M NUMBER(2) ;
BEGIN
  N := 1;
  LOOP
   DBMS_OUTPUT.PUT_LINE('CURRENT OUTER ITERATION IS : '||N);
   N := N + 1;
        M := 1;
        LOOP
                DBMS_OUTPUT.PUT_LINE('------> CURRENT INNER ITERATION IS : '||M);
                M := M + 1;
                EXIT WHEN M>5;
        END LOOP;
   EXIT WHEN N>3;
  END LOOP;
END;
/
```

(TEST43.SQL)

The inner loop is executed five times for every outer iteration i.e. when outer iteration number is 1, inner iteration will be executed for 5 times. Then for outer iteration number 2, inner iteration again will be executed for 5 times. This continues till outer iteration number exceeds 3.

Output of TEST43.SQL is as follows:

```
SQL> @TEST43

PL/SQL procedure successfully completed.

CURRENT OUTER ITERATION IS : 1
------> CURRENT INNER ITERATION IS : 1
------> CURRENT INNER ITERATION IS : 2
------> CURRENT INNER ITERATION IS : 3
------> CURRENT INNER ITERATION IS : 4
------> CURRENT INNER ITERATION IS : 5
CURRENT OUTER ITERATION IS : 2
------> CURRENT INNER ITERATION IS : 1
------> CURRENT INNER ITERATION IS : 2
------> CURRENT INNER ITERATION IS : 3
------> CURRENT INNER ITERATION IS : 4
------> CURRENT INNER ITERATION IS : 5
CURRENT OUTER ITERATION IS : 3
------> CURRENT INNER ITERATION IS : 1
------> CURRENT INNER ITERATION IS : 2
------> CURRENT INNER ITERATION IS : 3
------> CURRENT INNER ITERATION IS : 4
------> CURRENT INNER ITERATION IS : 5

PL/SQL procedure successfully completed.

SQL>
```

4.1.2 FOR Loop

Another loop provided by PL/SQL is "For" loop. For loop is not infinite. It has a definite end. In the FOR loop header a counter variable is used which is compared between Lower Bound and Upper Bound values. The variable is initialized to the Lower bound value and is incremented with iterations. The statements block is executed as long as variable's value remains less than or equal to upper bound value.

Syntax of FOR loop is :

```
FOR counter IN val1..val2
LOOP
        statement(s);
END LOOP;
```

val1 - Start integer value.
val2 - End integer value.

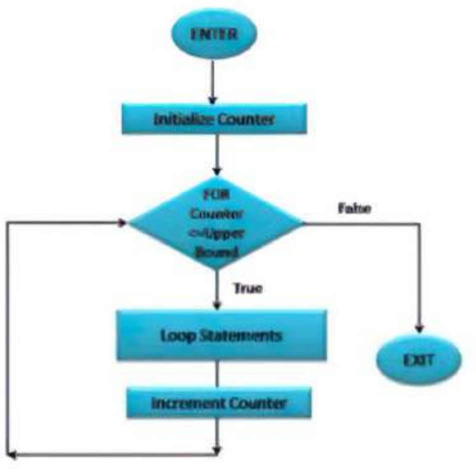

Fig 4.3 FOR LOOP Flow Diagram

Let us understand our earlier program using FOR loop.

```
SET SERVEROUTPUT ON;
EXECUTE DBMS_OUTPUT.ENABLE(1000);

BEGIN
  FOR N IN 1..10
  LOOP
   DBMS_OUTPUT.PUT_LINE('CURRENT ITERATION IS : '||N);
  END LOOP;
END;
/
```

(TEST44.SQL)

Output of TEST44.SQL is same as output of TEST41.SQL

There are some important and interesting observations to be made in above program:

1) The counter variable (N) is implicitly declared in the declaration section, so it's not necessary to declare it explicitly.

2) The counter variable is incremented by 1 in each iteration and does not need to be incremented explicitly.

3) EXIT WHEN statement and EXIT statements can be used in FOR loops but it's not done often.

Now, let us write the same SQL script for nested loop using FOR statement.

```
SET SERVEROUTPUT ON;
EXECUTE DBMS_OUTPUT.ENABLE(1000);

BEGIN
  FOR N IN 1..3
  LOOP
   DBMS_OUTPUT.PUT_LINE('CURRENT OUTER ITERATION IS :  '||N);
        FOR M IN 1..5
        LOOP
                DBMS_OUTPUT.PUT_LINE('------> CURRENT INNER  ITERATION IS : '||M);
        END LOOP;
  END LOOP;
END;
/
```

(TEST45.SQL)

So, it is very clear that if we have fix number of iterations, FOR loop is the simplest and shortest loop amongst all loops.

4.1.3 WHILE Loop

Another loop provided by PL/SQL is WHILE loop. The condition is evaluated at the beginning of each iteration. The iteration continues until the condition becomes false.

The General Syntax to write a WHILE LOOP is:

```
WHILE <condition>
        LOOP statement(s);
END LOOP;
```

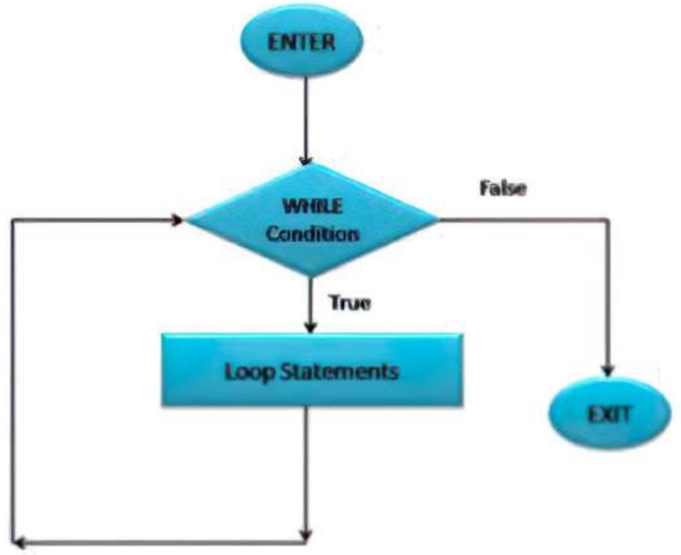

Fig 4.4 WHILE Loop Flow Diagram

Let us understand our earlier program using WHILE loop.

```
SET SERVEROUTPUT ON;
EXECUTE DBMS_OUTPUT.ENABLE(1000);

DECLARE
  N NUMBER(2) := 1;
BEGIN
```

```
  WHILE N <= 10
  LOOP
    DBMS_OUTPUT.PUT_LINE('CURRENT ITERATION IS : '||N);
    N := N + 1;
  END LOOP;
END;
/
```

(TEST46.SQL)

Few observations to be made in above program are:

1. Variable N is declared and initialized to value 1 in declaration section.

2. There is no EXIT or EXIT WHEN statements used in above program as the condition is being checked with WHILE statement. If the condition returns TRUE, the loop will be entered. Otherwise, in case it returns FALSE, the loop is exited.

3. As the condition is checked at the time of entering into the loop, WHILE loop is also known as entry controlled loop.

4. As the condition is checked before entering into the loop, it may be the case that the statements written inside WHILE loop may not be executed even once. (If the condition returns FALSE in the first iteration itself.) On the other hand, LOOP-END LOOP enters at least once and then the condition is checked for termination of the loop.

Let us write a program for nested loop using WHILE loop.

```
SET SERVEROUTPUT ON;
EXECUTE DBMS_OUTPUT.ENABLE(1000);

DECLARE
  N NUMBER(2) ;
  M NUMBER(2) ;
BEGIN
  N := 1;
  WHILE N <= 3
  LOOP
    DBMS_OUTPUT.PUT_LINE('CURRENT OUTER ITERATION IS : '||N);
    N := N + 1;
         M := 1;
      WHILE M <= 5
         LOOP
               DBMS_OUTPUT.PUT_LINE('------> CURRENT INNER ITERATION IS : '||M);
```

```
            M := M + 1;
        END LOOP;
  END LOOP;
END;
/
```

(TEST47.SQL)

In nutshell, WHILE is entry controlled loop and the loop will be entered for iterations if the condition with WHILE returns TRUE.

4.1.4 EXIT Loop

As we have seen earlier, EXIT statement is written inside loops. Generally it is used with LOOP..END LOOP or WHILE LOOPs. It is generally not used within FOR loop.
There are two ways EXIT statement is used inside a loop.

1. EXIT

The execution of loop is exited or terminated if EXIT statement is encountered. (Refer TEST41.SQL)

2. EXIT WHEN <Condition>

The execution of loop is exited or terminated if the condition written with EXIT EHEN statement returns TRUE. (Refer TEST42.SQL)

4.1.5 CONTINUE

CONTINUE is another statement generally written inside a loop. CONTINUE statement is used when you need to return back to start of next iteration of a loop. With CONTINUE, the loop terminates the current iteration and control passes to LOOP header to start next iteration. It means, when CONTINUE statement is encountered in a loop it forces the control to be passed to next iteration, without executing remaining statements written after CONTINUE statement. It can be combined with an IF- THEN-END IF statement to make it conditional.

Let us understand CONTINUE statement using following program to print all odd numbers between 1 to 10.

```
SET SERVEROUTPUT ON;
EXECUTE DBMS_OUTPUT.ENABLE(1000);
DECLARE
        N NUMBER:=0;
BEGIN
WHILE N < 10
LOOP
```

```
  N:=N+1;
  IF mod(N,2)=0 then
    CONTINUE;
  END IF;
  DBMS_OUTPUT.PUT_LINE('N ='||N);
END LOOP;
END;
/
```

(TEST48.SQL)

Output of above program is as follows:

```
SQL> @48

PL/SQL procedure successfully completed.

N =1
N =3
N =5
N =7
N =9

PL/SQL procedure successfully completed.

SQL>
```

To understand above program, we need to understand MOD function, which is used in this program.

Syntax of MOD is

MOD (argumet1, aergument2);

MOD returns the remainder (modulus) of argument 1 divided by argument 2.

e.g.

MOD(2,2) will return 0

MOD(3,2) will return 1

MOD(4,2) will return 0

MOD(5,2) will return 1

In our program (TEST48.SQL) if MOD function returns 0, it means the current value of variable N is even. As we want to print only odd numbers, for each even value of N, CONTINUE statement is used. It ignores execution of remaining statements after it. (in our case DBMS_OUTPUT.PUT_LINE statement). And it passes the control to header of the loop. But if MOD function returns 1, the IF statement returns FALSE and CONTINUE statement will not be executed. This means control will continue executing remaining statement (DBMS_OUTPUT.PUT_LINE). And all odd numbers will be printed.

****