Database Programming

3.Iterative Control (Loops)

Sagara Samarawickrama

CSD 4203 – 2021W

3.Loops

WHILE Loops

A WHILE loop has the structure as shown below

```
WHILE TEST CONDITION LOOP
  STATEMENT 1;
  STATEMENT 2;
  …
  STATEMENT N;
END LOOP;
```

The reserved word WHILE marks the beginning of a loop construct. The TEST CONDITION is the test condition of the loop that evaluates to TRUE or FALSE. The result of this evaluation determines whether the loop is executed. Statements 1 through N are a sequence of statements that is executed repeatedly. END LOOP is a reserved phrase that indicates the end of the loop construct.

As mentioned earlier, before the body of the loop can be executed, the test condition must be evaluated. The decision as to whether to execute the statements in the body of the loop is made prior to entering the loop. As a result, the loop will not be executed at all if the test condition yields FALSE

3.Loops

WHILE Loops

A WHILE loop has the structure as shown below

```
DECLARE
  v_counter NUMBER := 1;
BEGIN
  WHILE v_counter <= 5
  LOOP
     DBMS_OUTPUT.PUT_LINE ('v_counter = '||v_counter);

    IF v_counter = 2
    THEN
      EXIT;
    END IF;

    v_counter := v_counter + 1;
  END LOOP;
END;
```

Note ; According to the test condition, the loop should execute five times. However, the loop is executed only twice, because the exit condition is present inside the body of the loop. Therefore, the loop terminates prematurely.

3.Loops

Numeric FOR Loops

A numeric FOR loop is called numeric because it requires an integer as its terminating value. The structure of such a loop is shown below

```
FOR loop_counter IN [REVERSE] lower_limit..upper_limit
LOOP
STATEMENT 1;
STATEMENT 2;
…
STATEMENT N;
END LOOP;
```

The reserved word FOR marks the beginning of the FOR loop construct. The variable loop counter is an implicitly defined index variable. There is no need to define the loop counter in the declaration section of the PL/SQL block; instead, this variable is defined by the loop construct. The lower_limit and upper_limit are integer numbers or expressions that evaluate to integer values at run time, and the double dot (..) serves as the range operator.

3.Loops

Numeric FOR Loops

Consider the following example, which illustrates a numeric FOR loop that employs the IN option.

```
BEGIN
  FOR v_counter IN 1..5
  LOOP
    DBMS_OUTPUT.PUT_LINE ('v_counter = '||v_counter);
  END LOOP;
END;
```

Note ; According to the test condition, the loop should execute five times. However, the loop is executed only twice, because the exit condition is present inside the body of the loop. Therefore, the loop terminates prematurely.

Watch Out!

The loop counter is implicitly defined and incremented when a numeric FOR loop is used. As a result, it cannot be referenced outside the body of the FOR loop.

3.Loops

```
BEGIN
  FOR v_counter IN 1..5
  LOOP
    DBMS_OUTPUT.PUT_LINE ('v_counter = '||v_counter);
  END LOOP;
  DBMS_OUTPUT.PUT_LINE ('Counter outside the loop is
'||v_counter);
END;
```

When this example is run, it produces the following error message because the loop counter is declared implicitly by the loop, the variable v_counter cannot be referenced outside the loop

The next example demonstrates the usage of the IN REVERSE option for the loop.

```
BEGIN
  FOR v_counter IN REVERSE 1..5
  LOOP
    DBMS_OUTPUT.PUT_LINE ('v_counter = '||v_counter);
  END LOOP;
END;
```

3.Loops

Premature Termination of the Numeric FOR Loop

The EXIT statement can be used inside the body of a numeric FOR loop . If the exit condition evaluates to TRUE before the loop counter reaches its terminal value, the FOR loop is terminated prematurely. If the loop counter reaches its terminal value before the exit condition yields TRUE, there is no premature termination of the FOR loop.

```
FOR loop_counter IN lower_limit..upper_limit
LOOP
  STATEMENT 1;
  STATEMENT 2;
  IF EXIT CONDITION THEN
    EXIT;
  END IF;
END LOOP;
STATEMENT 3;
```

```
BEGIN
  FOR v_counter IN 1..5
  LOOP
    DBMS_OUTPUT.PUT_LINE ('v_counter = '||v_counter);
    EXIT WHEN v_counter = 3;
  END LOOP;
END;
```

According to the range specified, the loop should execute five times. However, the loop executes only three times because the exit condition appears inside the body of the loop. Thus, the loop terminates prematurely

3.Loops

Nested Loops

We have explored three types of loops: simple loops, WHILE loops, and numeric FOR loops. Any of these three types of loops can be nested inside one another

Following example, demonstrate how a simple loop can be nested inside a WHILE loop.

```
DECLARE
  v_counter1 BINARY_INTEGER := 0;
  v_counter2 BINARY_INTEGER;
BEGIN
  WHILE v_counter1 < 3
  LOOP
    DBMS_OUTPUT.PUT_LINE ('v_counter1: '||v_counter1);
    v_counter2 := 0;
    LOOP
      DBMS_OUTPUT.PUT_LINE ('  v_counter2: '||v_counter2);
      v_counter2 := v_counter2 + 1;
      EXIT WHEN v_counter2 >= 2;
    END LOOP;
    v_counter1 := v_counter1 + 1;
  END LOOP;
END;
```

3.Loops

Nested For Loop

Following is a nested For Loop.

```
BEGIN
  <<outer>>
  FOR v_counter IN 1..3
  LOOP
    <<inner>>
    FOR v_counter IN 1..2
    LOOP
      DBMS_OUTPUT.PUT_LINE ('outer.v_counter '||outer.v_counter);
      DBMS_OUTPUT.PUT_LINE ('inner.v_counter '||inner.v_counter);
    END LOOP inner;
  END LOOP outer;
END;
```

In this example, both the inner and outer loops use the same loop counter, v_counter. To reference both the outer and inner values of v_counter, loop labels are used . It should generate the following output

```
outer.v_counter 1
inner.v_counter 1
outer.v_counter 1
inner.v_counter 2
outer.v_counter 2
inner.v_counter 1
```

```
outer.v_counter 2
inner.v_counter 2
outer.v_counter 3
inner.v_counter 1
outer.v_counter 3
inner.v_counter 2
```

3.Loops

Using Loop Labels

Loops can be labeled it will improve the readability of the code.

```
BEGIN
  <<outer_loop>>
  FOR i IN 1..3
  LOOP
    DBMS_OUTPUT.PUT_LINE ('i = '||i);
    <<inner_loop>>
    FOR j IN 1..2
    LOOP
      DBMS_OUTPUT.PUT_LINE ('j = '||j);
    END LOOP inner_loop;
  END LOOP outer_loop;
END;
```

For both outer and inner loops, the statement END LOOP must be used. *If the loop label is added to each END LOOP statement, it becomes easier to understand which loop is being terminated.*