Database Programming with PL/SQL

Iterative Control: Basic Loops

Objectives

This lesson covers the following objectives:

- Describe the need for `LOOP` statements in PL/SQL
- Recognize different types of `LOOP` statements
- Create PL/SQL containing a basic loop and an `EXIT` statement
- Create PL/SQL containing a basic loop and an `EXIT` statement with conditional termination

Purpose

Looping constructs are the second type of control structure. Loops are mainly used to execute statements repeatedly until an `EXIT` condition is reached.

PL/SQL provides three ways to structure loops to repeat a statement or a sequence of statements multiple times. These are basic loops, `FOR` loops, and `WHILE` loops.

Purpose (cont.)

This lesson introduces the three loop types and discusses basic loops in greater detail.

Iterative Control: `LOOP` Statements

Loops repeat a statement or a sequence of statements multiple times. PL/SQL provides the following types of loops:

- Basic loops that perform repetitive actions without overall conditions
- `FOR` loops that perform iterative actions based on a counter
- `WHILE` loops that perform repetitive actions based on a condition

Basic Loops

The simplest form of a `LOOP` statement is the basic (or infinite) loop, which encloses a sequence of statements between the keywords `LOOP` and `END LOOP`.

Use the basic loop when the statements inside the loop must execute at least once.

Basic Loops Exit

Each time the flow of execution reaches the `END LOOP` statement, control is passed to the corresponding `LOOP` statement that introduced it.

A basic loop allows the execution of its statements at least once, even if the `EXIT` condition is already met upon entering the loop. Without the `EXIT` statement, the loop would be infinite.Syntax:

```
LOOP
  statement1;
  . . .
  EXIT [WHEN condition];
END LOOP;
```

Basic Loops Example

In this example, three new location IDs for the country code of CA and the city of Montreal are inserted.

```
DECLARE
  v_countryid    locations.country_id%TYPE := 'CA';
  v_loc_id       locations.location_id%TYPE;
  v_counter       NUMBER(2) := 1;
  v_new_city     locations.city%TYPE := 'Montreal';
BEGIN
  SELECT MAX(location_id) INTO v_loc_id FROM locations
    WHERE country_id = v_countryid;
  LOOP
    INSERT INTO locations(location_id, city, country_id)
    VALUES((v_loc_id + v_counter), v_new_city, v_countryid);
    v_counter := v_counter + 1;
    EXIT WHEN v_counter > 3;
  END LOOP;
END;
```

Basic Loops EXIT Statement

You can use the EXIT statement to terminate a loop. The control passes to the next statement after the END LOOP statement. You can issue EXIT either as an action within an IF statement or as a stand-alone statement within the loop.

```
DECLARE
  v_counter NUMBER := 1;
BEGIN
  LOOP
    DBMS_OUTPUT.PUT_LINE('The square of '
                  ||v_counter||' is: '|| POWER(v_counter,2));
    v_counter := v_counter + 1;
    IF v_counter > 10 THEN
      EXIT;
    END IF;
  END LOOP;
END;
```

Basic Loop EXIT Statement Rules

Rules:

- The EXIT statement must be placed inside a loop.
- If the EXIT condition is placed at the top of the loop (before any of the other executable statements) and that condition is initially true, then the loop exits and the other statements in the loop never execute.
- A basic loop can contain multiple EXIT statements, but you should have only one EXIT point.

Basic Loop EXIT WHEN Statement

Use the WHEN clause to allow conditional termination of the loop. When the EXIT statement is encountered, the condition in the WHEN clause is evaluated. If the condition yields TRUE, then the loop ends and control passes to the next statement following the loop.

```
DECLARE
  v_counter NUMBER := 1;
BEGIN
  LOOP
    DBMS_OUTPUT.PUT_LINE('The square of '
                ||v_counter||' is: '|| POWER(v_counter,2));
    v_counter := v_counter + 1;
    EXIT WHEN v_counter > 10;
  END LOOP;
END;
```

Terminology

Key terms used in this lesson included:

- Basic (infinite) loop
- `EXIT`

Summary

In this lesson, you should have learned how to:

- Describe the need for `LOOP` statements in PL/SQL
- Recognize different types of `LOOP` statements
- Create PL/SQL containing a basic loop and an `EXIT` statement
- Create PL/SQL containing a basic loop and an `EXIT` statement with conditional termination