
ORACLE
Academy

Database Programming with PL/SQL

4-3

Iterative Control: Basic Loops

ORACLE
Academy

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

ORACLE Academy

WARNING:

When a user executes a PL/SQL block in APEX that contains an infinite loop, there is no way for the user to stop the loop. It can only be stopped by the DBA who oversees the Academy database. If the loop does not contain DML statements, closing the browser window and reopening it will allow the user to continue coding. If the block contains a FOR UPDATE clause, the affected table(s) will remain locked until released by the DBA, which may take a day or more.

Be careful executing loops. All loops required by this course should have some type of limiting condition. NO infinite loops are required as part of this course.

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

Each loop is structured for a specific purpose. These loops are used to write code to handle all situations (problems). Loops can repeat one statement, a group of statements, and/or a block. Loops have a scope and loop variables have a life.

Basic Loops

- The simplest form of a LOOP statement is the basic 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 never end (an infinite loop)

```
BEGIN
  LOOP
    statements;
    EXIT [WHEN condition];
  END LOOP;
END;
```

Basic Loops Simple Example

- In this example, no data is processed
- We simply display the loop counter each time we repeat the loop

```
DECLARE
  v_counter   NUMBER(2) := 1;
BEGIN
  LOOP
    DBMS_OUTPUT.PUT_LINE('Loop execution #' || v_counter);
    v_counter := v_counter + 1;
    EXIT WHEN v_counter > 5;
  END LOOP;
END;
```

Basic Loops More Complex Example

- In this example, three new location IDs for Montreal, Canada, are inserted in the LOCATIONS table

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

ORACLE Academy

Basic Loops EXIT Statement

- You can use the EXIT statement to terminate a loop and pass control to the next statement after the END LOOP statement
- You can issue EXIT as an action within an IF statement

```
DECLARE
  v_counter NUMBER := 1;
BEGIN
  LOOP
    DBMS_OUTPUT.PUT_LINE('Counter is ' || v_counter);
    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

Basic Loop EXIT WHEN Statement

- Although the IF…THEN EXIT works to end a loop, the correct way to end a basic loop is with the EXIT WHEN statement
- If the WHEN clause evaluates to TRUE, the loop ends and control passes to the next statement following END LOOP

```
DECLARE
  v_counter NUMBER := 1;
BEGIN
  LOOP
    DBMS_OUTPUT.PUT_LINE('Counter is ' || v_counter);
    v_counter := v_counter + 1;
    EXIT WHEN v_counter > 10;
  END LOOP;
END;
```

ORACLE Academy

The statement, EXIT WHEN v_counter > 10, is logically identical to:

IF v_counter > 10 THEN EXIT;

END IF;

However, in PL/SQL, the EXIT WHEN statement is the appropriate syntax. From a programmer's perspective, it is the more elegant solution.

Terminology

- Key terms used in this lesson included:
  - Basic Loop
  - Counter
  - END LOOP
  - EXIT
  - LOOP

ORACLE Academy

- Basic Loop – A sequence of statements between the keywords LOOP and END LOOP; the statements must execute at least once.
- Counter – A counter is a variable that programmers use to keep track of the number of times a loop executes (or repeats). When a specific value is reached, it triggers the execution of the EXIT command which stops the loop from repeating, passing control to the first command following the END LOOP command.
- END LOOP – A PL/SQL command that marks the end of the statements within a loop and returns control to the corresponding LOOP statement that introduced it (the loop begins again).
- EXIT – Statement to terminate a loop.
- LOOP – A PL/SQL command that marks the beginning of the statements within a loop.

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


ORACLE
Academy