
ORACLE
ACADEMY

Database Programming with PL/SQL

5-4
Cursors with Parameters

ORACLE ACADEMY

Objectives

This lesson covers the following objectives:

- List the benefits of using parameters with cursors
- Create PL/SQL code to declare and use a cursor with a parameter

Purpose

- Consider a program which declares a cursor to fetch and process all the employees in a given department, and the department is chosen by the user at runtime.
- How would we declare the cursor?
- We don't know the department id when we write the code, but this won't work.

```
DECLARE
  CURSOR cur_emps IS
    SELECT * FROM employees
      WHERE department_id = ???;
```

Purpose

- There are several departments.
- Do we need to declare several cursors, one for each department, each with a different value in the WHERE clause?
- No. We can declare just one cursor to handle all departments by using parameters.

Cursors with Parameters

- A parameter is a variable whose name is used in a cursor declaration.
- When the cursor is opened, the parameter value is passed to the Oracle server, which uses it to decide which rows to retrieve into the active set of the cursor.

Cursors with Parameters

- This means that you can open and close an explicit cursor several times in a block, or in different executions of the same block, returning a different active set on each occasion.
- Consider an example where you pass a location_id to a cursor and it returns the names of the departments at that location.
- The next slide shows how.

Cursors with Parameters: Example

```
DECLARE
  CURSOR cur_country (p_region_id NUMBER) IS
    SELECT country_id, country_name
      FROM countries
      WHERE region_id = p_region_id;
  v_country_record    cur_country%ROWTYPE;
BEGIN
  OPEN cur_country (5);
  LOOP
    FETCH cur_country INTO v_country_record;
    EXIT WHEN cur_country%NOTFOUND;
    DBMS_OUTPUT.PUT_LINE(v_country_record.country_id || ' '
                         || v_country_record.country_name);
  END LOOP;
  CLOSE cur_country;
END;
```

Change to whichever region is required.

Defining Cursors with Parameters Syntax

- Each parameter named in the cursor declaration must have a corresponding value in the OPEN statement.
- Parameter data types are the same as those for scalar variables, but you do not give them sizes.
- The parameter names are used in the WHERE clause of the cursor SELECT statement.

```
CURSOR cursor_name
  [(parameter_name datatype, ...)]
IS
  select_statement;
```

Defining Cursors with Parameters Syntax

In the syntax:

- *cursor_name* Is a PL/SQL identifier for the declared cursor
- *parameter_name* Is the name of a parameter
- *datatype* Is the scalar data type of the parameter
- *select_statement* Is a SELECT statement without the INTO clause

```
CURSOR cursor_name
  [(parameter_name datatype, ...)]
IS
  select_statement;
```

Opening Cursors with Parameters

The following is the syntax for opening a cursor with parameters:

```
OPEN cursor_name(parameter_value1, parameter_value2, ...);
```

Cursors with Parameters

- You pass parameter values to a cursor when the cursor is opened.
- Therefore you can open a single explicit cursor several times and fetch a different active set each time.
- In the following example, a cursor is opened several times.

```
DECLARE
  CURSOR cur_countries (p_region_id NUMBER) IS
    SELECT country_id, country_name FROM countries
      WHERE region_id = p_region_id;
  v_country_record    c_countries%ROWTYPE;
BEGIN
  OPEN cur_countries (5);
  ...
  CLOSE cur_countries;
  OPEN cur_countries (145);
  ...
```

Open the cursor again and return a different active set.

Another Example of a Cursor with a Parameter

```
DECLARE
  v_deptid       employees.department_id%TYPE;
  CURSOR cur_emps (p_deptid NUMBER) IS
    SELECT employee_id, salary
      FROM employees
      WHERE department_id = p_deptid;
  v_emp_rec      cur_emps%ROWTYPE;
BEGIN
  SELECT MAX(department_id) INTO v_deptid
    FROM employees;
  OPEN cur_emps(v_deptid);
  LOOP
    FETCH cur_emps INTO v_emp_rec;
    EXIT WHEN cur_emps%NOTFOUND;
    DBMS_OUTPUT.PUT_LINE(v_emp_rec.employee_id || '  '
                         || v_emp_rec.salary);
  END LOOP;
  CLOSE cur_emps;
END;
```

Cursor FOR Loops with a Parameter

We can use a cursor FOR loop if needed:

```
DECLARE
  CURSOR   cur_emps (p_deptno NUMBER) IS
    SELECT  employee_id, last_name
      FROM    employees
      WHERE   department_id = p_deptno;
BEGIN
  FOR v_emp_record IN cur_emps(10) LOOP
    ...
  END LOOP;
END;
```

Cursors with Multiple Parameters: Example 1

In the following example, a cursor is declared and is called with two parameters:

```
DECLARE
  CURSOR cur_countries (p_region_id NUMBER, p_population NUMBER) IS
    SELECT  country_id, country_name, population
      FROM    countries
      WHERE   region_id = p_region_id
      OR     population > p_population;
BEGIN
  FOR v_country_record IN cur_countries(145,10000000) LOOP
    DBMS_OUTPUT.PUT_LINE(v_country_record.country_id ||' '
                         || v_country_record. country_name||' '
                         || v_country_record.population);
  END LOOP;
END;
```

Cursors with Multiple Parameters: Example 2

This cursor fetches all IT Programmers who earn more than $10000.

```
DECLARE
  CURSOR cur_emps (p_job VARCHAR2, p_salary NUMBER) IS
    SELECT  employee_id, last_name
      FROM    employees
      WHERE   job_id  = p_job
      AND     salary > p_salary;
BEGIN
  FOR v_emp_record IN cur_emps('IT_PROG', 10000) LOOP
    DBMS_OUTPUT.PUT_LINE(v_emp_record.employee_id ||' '
                         || v_emp_record.last_name);
  END LOOP;
END;
```

Summary

In this lesson, you should have learned how to:

- List the benefits of using parameters with cursors
- Create PL/SQL code to declare and use a cursor with a parameter


ORACLE
ACADEMY