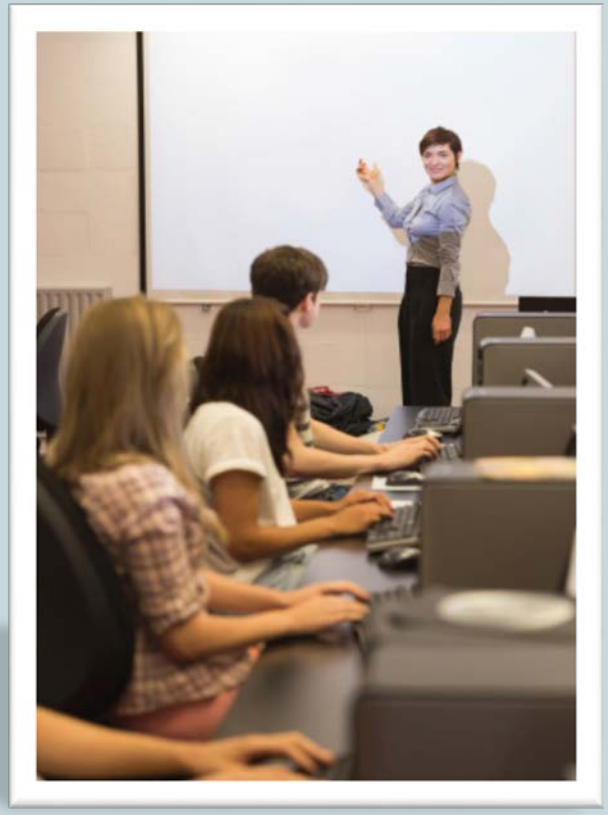

ORACLE
Academy

Database Programming with PL/SQL

5-6
Using Multiple Cursors



ORACLE Academy

Objectives

This lesson covers the following objectives:

- Explain the need for using multiple cursors to produce multi-level reports
- Create PL/SQL code to declare and manipulate multiple cursors within nested loops
- Create PL/SQL code to declare and manipulate multiple cursors using parameters

Purpose

- In real-life programs you often need to declare and use two or more cursors in the same PL/SQL block.
- Often these cursors are related to each other by parameters.
- One common example is the need for multi-level reports in which each level of the report uses rows from a different cursor.
- This lesson does not introduce new concepts or syntax.
- It shows more powerful uses for the concepts and syntax that you already know.

A Sample Problem Statement

- You need to produce a report that lists each department as a sub-heading, immediately followed by a listing of the employees in that department, followed by the next department, and so on.
- You need two cursors, one for each of the two tables.
- The cursor based on EMPLOYEES is opened several times, once for each department.

Problem Solution: Step 1

- Declare two cursors, one for each table, plus associated record structures.
- Why is cursor cur_emp declared with a parameter?

```
DECLARE
 CURSOR cur_dept IS
  SELECT department_id, department_name
   FROM departments
   ORDER BY department_name;
 CURSOR cur_emp (p_deptid NUMBER) IS
  SELECT first_name, last_name
   FROM employees
   WHERE department_id = p_deptid
   ORDER BY last_name;
 v_deptrec  cur_dept%ROWTYPE;
 v_emprec  cur_emp%ROWTYPE;
```

Problem Solution: Step 2

- Open the cur_dept cursor and fetch and display the DEPARTMENTS rows in the usual way.

```
DECLARE
 CURSOR cur_dept IS .....;
 CURSOR cur_emp (p_deptid NUMBER) IS .....;
 v_deptrec  cur_dept%ROWTYPE;
 v_emprec  cur_emp%ROWTYPE;
BEGIN
 OPEN cur_dept;
 LOOP
  FETCH cur_dept INTO v_deptrec;
  EXIT WHEN cur_dept%NOTFOUND;
  DBMS_OUTPUT.PUT_LINE(v_deptrec.department_name);
 END LOOP;
 CLOSE cur_dept;
END;
```

Problem Solution: Step 3

- After each DEPARTMENTS row has been fetched and displayed, you need to fetch and display the EMPLOYEES in that department.
- To do this, you open the EMPLOYEES cursor, fetch and display its rows in a nested loop, and close the cursor.
- Then, you do the same for the next DEPARTMENTS row.
- And so on.
- The next slide shows the code for this.

Problem Solution

```
DECLARE
 CURSOR cur_dept IS .....;
 CURSOR cur_emp (p_deptid NUMBER) IS .....;
 v_deptrec  cur_dept%ROWTYPE;
 v_emprec  cur_emp%ROWTYPE;
BEGIN
 OPEN cur_dept;
 LOOP
  FETCH cur_dept INTO v_deptrec;
  EXIT WHEN cur_dept%NOTFOUND;
  DBMS_OUTPUT.PUT_LINE(v_deptrec.department_name);
  OPEN cur_emp (v_deptrec.department_id);
  LOOP
   FETCH cur_emp INTO v_emprec;
   EXIT WHEN cur_emp%NOTFOUND;
   DBMS_OUTPUT.PUT_LINE(v_emprec.last_name || ' ' ||
      v_emprec.first_name);
  END LOOP;
  CLOSE cur_emp;
 END LOOP;
 CLOSE cur_dept;
END;
```

A Second Example

- You need to produce a report that lists each location in which your departments are situated, followed by the departments in that location.
- Again, you need two cursors, one for each of the two tables.
- The cursor based on DEPARTMENTS will be opened several times, once for each location.
- The next slide shows the code needed to produce this report.

A Second Example

```
DECLARE
 CURSOR cur_loc IS SELECT * FROM locations;
 CURSOR cur_dept (p_locid NUMBER) IS
  SELECT * FROM departments WHERE location_id = p_locid;
 v_locrec  cur_loc%ROWTYPE;
 v_deptrec cur_dept%ROWTYPE;
BEGIN
 OPEN cur_loc;
 LOOP
  FETCH cur_loc INTO v_locrec;
  EXIT WHEN cur_loc%NOTFOUND;
  DBMS_OUTPUT.PUT_LINE(v_locrec.city);
  OPEN cur_dept (v_locrec.location_id);
  LOOP
   FETCH cur_dept INTO v_deptrec;
   EXIT WHEN cur_dept%NOTFOUND;
   DBMS_OUTPUT.PUT_LINE(v_deptrec.department_name);
  END LOOP;
  CLOSE cur_dept;
 END LOOP;
 CLOSE cur_loc;
END;
```

Using FOR Loops with Multiple Cursors

- You can use FOR loops (and other cursor techniques, such as FOR UPDATE) with multiple cursors, just as you can with single cursors.

```
DECLARE
 CURSOR cur_loc IS SELECT * FROM locations;
 CURSOR cur_dept (p_locid NUMBER) IS
  SELECT * FROM departments WHERE location_id = p_locid;
BEGIN
 FOR v_locrec IN cur_loc
 LOOP
  DBMS_OUTPUT.PUT_LINE(v_locrec.city);
  FOR v_deptrec IN cur_dept (v_locrec.location_id)
  LOOP
   DBMS_OUTPUT.PUT_LINE(v_deptrec.department_name);
  END LOOP;
 END LOOP;
END;
```

A Final Example

- Which employees will receive a salary increase by running the code below?

```
DECLARE
 CURSOR cur_dept IS SELECT * FROM my_departments;
 CURSOR cur_emp (p_dept_id NUMBER) IS
  SELECT * FROM my_employees WHERE department_id = p_dept_id
   FOR UPDATE NOWAIT;
BEGIN
 FOR v_deptrec IN cur_dept LOOP
  DBMS_OUTPUT.PUT_LINE(v_deptrec.department_name);
  FOR v_emprec IN cur_emp (v_deptrec.department_id) LOOP
   DBMS_OUTPUT.PUT_LINE(v_emprec.last_name);
   IF v_deptrec.location_id = 1700 AND v_emprec.salary < 10000
    THEN UPDATE my_employees
     SET salary = salary * 1.1
     WHERE CURRENT OF cur_emp;
   END IF;
  END LOOP;
 END LOOP;
END;
```

Summary

In this lesson, you should have learned how to:

- Explain the need for using multiple cursors to produce multi-level reports
- Create PL/SQL code to declare and manipulate multiple cursors within nested loops
- Create PL/SQL code to declare and manipulate multiple cursors using parameters


ORACLE
Academy