Practices for Lesson 8: Using Explicit Cursors

Chapter 8

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Practice 8-1: Using Explicit Cursors

In this practice, you perform two exercises:

- First, you use an explicit cursor to process a number of rows from a table and populate another table with the results by using a cursor `FOR` loop.
- Second, you write a PL/SQL block that processes information with two cursors, including one that uses a parameter.

1. Create a PL/SQL block to perform the following:
   a. In the declarative section, declare and initialize a variable named `v_deptno` of type `NUMBER`. Assign a valid department ID value (see table in step d for values).
   b. Declare a cursor named `c_emp_cursor`, which retrieves the `last_name`, `salary`, and `manager_id` of employees working in the department specified in `v_deptno`.
   c. In the executable section, use the cursor `FOR` loop to operate on the data retrieved. If the salary of the employee is less than 5,000 and if the manager ID is either 101 or 124, display the message "<<*last_name*>> Due for a raise." Otherwise, display the message "<<*last_name*>> Not Due for a raise."
   d. Test the PL/SQL block for the following cases:

Department ID	Message
10	Whalen Due for a raise
20	Hartstein Not Due for a raise Fay Not Due for a raise
50	Weiss Not Due for a raise Fripp Not Due for a raise Kaufling Not Due for a raise Vollman Not Due for a raise. . . . . . OConnell Due for a raise Grant Due for a raise
80	Russell Not Due for a raise Partners Not Due for a raise Errazuriz Not Due for a raise Cambrault Not Due for a raise . . . Livingston Not Due for a raise Johnson Not Due for a raise

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

2. Next, write a PL/SQL block that declares and uses two cursors—one without a parameter and one with a parameter. The first cursor retrieves the department number and department name from the `DEPARTMENTS` table for all departments whose ID number is less than 100. The second cursor receives the department number as a parameter, and retrieves employee details for those who work in that department and whose `employee_id` is less than 120.

   a. Declare a cursor `c_dept_cursor` to retrieve `department_id` and `department_name` for those departments with `department_id` less than 100. Order by `department_id`.
   b. Declare another cursor `c_emp_cursor` that takes the department number as parameter and retrieves the following data from the `EMPLOYEES` table: `last_name`, `job_id`, `hire_date`, and `salary` of those employees who work in that department, with `employee_id` less than 120.
   c. Declare variables to hold the values retrieved from each cursor. Use the `%TYPE` attribute while declaring variables.
   d. Open `c_dept_cursor` and use a simple loop to fetch values into the variables that are declared. Display the department number and department name. Use the appropriate cursor attribute to exit the loop.
   e. Open `c_emp_cursor` by passing the current department number as a parameter. Start another loop and fetch the values of `emp_cursor` into variables, and print all the details retrieved from the `EMPLOYEES` table.

   Notes

   - Check whether `c_emp_cursor` is already open before opening the cursor.
   - Use the appropriate cursor attribute for the exit condition.
   - When the loop completes, print a line after you have displayed the details of each department, and close `c_emp_cursor`.

   f. End the first loop and close `c_dept_cursor`. Then end the executable section.

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

g. Execute the script. The sample output is as follows:

```
Script Output
Task completed in 0.008 seconds
PL/SQL procedure successfully completed.

Department Number : 10  Department Name : Administration
--------------------------------------------------------------------------------
Department Number : 20  Department Name : Marketing
--------------------------------------------------------------------------------
Department Number : 30  Department Name : Purchasing
Raphaely    PU_MAN   07-DEC-10   11000
Khoo    PU_CLERK   18-MAY-11   3100
Baida    PU_CLERK   24-DEC-13   2900
Tobias    PU_CLERK   24-JUL-13   2800
Himuro    PU_CLERK   15-NOV-14   2600
Colmenares    PU_CLERK   10-AUG-15   2500
--------------------------------------------------------------------------------
Department Number : 40  Department Name : Human Resources
--------------------------------------------------------------------------------
Department Number : 50  Department Name : Shipping
--------------------------------------------------------------------------------
Department Number : 60  Department Name : IT
Hunold    IT_PROG   03-JAN-14   9000
Ernst    IT_PROG   21-MAY-15   6000
Austin    IT_PROG   25-JUN-13   4800
Pataballa    IT_PROG   05-FEB-14   4800
Lorentz    IT_PROG   07-FEB-15   4200
--------------------------------------------------------------------------------
Department Number : 70  Department Name : Public Relations
--------------------------------------------------------------------------------
Department Number : 80  Department Name : Sales
--------------------------------------------------------------------------------
Department Number : 90  Department Name : Executive
King    AD_PRES   17-JUN-11   24000
Kochhar    AD_VP   21-SEP-09   17000
De Haan    AD_VP   13-JAN-09   17000
--------------------------------------------------------------------------------
```

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Solution 8-1: Using Explicit Cursors

In this practice, you perform two exercises:

- First, you use an explicit cursor to process a number of rows from a table and populate another table with the results by using a cursor `FOR` loop.
- Second, you write a PL/SQL block that processes information with two cursors, including one that uses a parameter.

1. Create a PL/SQL block to perform the following:
   a. In the declarative section, declare and initialize a variable named `v_deptno` of the `NUMBER` type. Assign a valid department ID value (see table in step d for values).

```
DECLARE
v_deptno NUMBER := 10;
```

   b. Declare a cursor named `c_emp_cursor`, which retrieves `last_name`, `salary`, and `manager_id` of employees working in the department specified in `v_deptno`.

```
CURSOR  c_emp_cursor IS
   SELECT     last_name, salary,manager_id
   FROM       employees
   WHERE      department_id = v_deptno;
```

   c. In the executable section, use the cursor `FOR` loop to operate on the data retrieved. If the salary of the employee is less than 5,000 and if the manager ID is either 101 or 124, display the message "<<*last_name*>> Due for a raise." Otherwise, display the message "<<*last_name*>> Not Due for a raise."

```
BEGIN
 FOR emp_record IN c_emp_cursor
  LOOP
   IF emp_record.salary < 5000 AND (emp_record.manager_id=101 OR
emp_record.manager_id=124) THEN
      DBMS_OUTPUT.PUT_LINE (emp_record.last_name || ' Due for a
raise');
   ELSE
      DBMS_OUTPUT.PUT_LINE (emp_record.last_name || ' Not Due for a
raise');
   END IF;
  END LOOP;
END;
```

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

d. Test the PL/SQL block for the following cases:

Department ID	Message
10	Whalen Due for a raise
20	Hartstein Not Due for a raise Fay Not Due for a raise
50	Weiss Not Due for a raise Fripp Not Due for a raise Kaufling Not Due for a raise Vollman Not Due for a raise. . . . . . OConnell Due for a raise Grant Due for a raise
80	Russell Not Due for a raise Partners Not Due for a raise Errazuriz Not Due for a raise Cambrault Not Due for a raise . . . Livingston Not Due for a raise Johnson Not Due for a raise

2. Next, write a PL/SQL block that declares and uses two cursors—one without a parameter and one with a parameter. The first cursor retrieves the department number and department name from the `DEPARTMENTS` table for all departments whose ID number is less than 100. The second cursor receives the department number as a parameter, and retrieves employee details for those who work in that department and whose `employee_id` is less than 120.

   a. Declare a cursor `c_dept_cursor` to retrieve `department_id` and `department_name` for those departments with `department_id` less than 100. Order by `department_id`.

```
DECLARE
  CURSOR c_dept_cursor IS
        SELECT department_id,department_name
        FROM  departments
        WHERE department_id < 100
        ORDER BY    department_id;
```

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

b. Declare another cursor `c_emp_cursor` that takes the department number as parameter and retrieves the following data from the `EMPLOYEES` table: `last_name`, `job_id`, `hire_date`, and `salary` of those employees who work in that department, with `employee_id` less than 120.

```
CURSOR c_emp_cursor(v_deptno NUMBER) IS
        SELECT last_name,job_id,hire_date,salary
        FROM  employees
        WHERE department_id = v_deptno
        AND employee_id < 120;
```

c. Declare variables to hold the values retrieved from each cursor. Use the `%TYPE` attribute while declaring variables.

```
  v_current_deptno departments.department_id%TYPE;
  v_current_dname  departments.department_name%TYPE;
  v_ename employees.last_name%TYPE;
  v_job employees.job_id%TYPE;
  v_hiredate employees.hire_date%TYPE;
  v_sal employees.salary%TYPE;
```

d. Open `c_dept_cursor` and use a simple loop to fetch values into the variables that are declared. Display the department number and department name. Use the appropriate cursor attribute to exit the loop.

```
BEGIN
   OPEN c_dept_cursor;
      LOOP
         FETCH c_dept_cursor INTO v_current_deptno,
           v_current_dname;
         EXIT WHEN c_dept_cursor%NOTFOUND;
         DBMS_OUTPUT.PUT_LINE ('Department Number : ' ||
           v_current_deptno || '  Department Name : ' ||
           v_current_dname);
```

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

e. Open `c_emp_cursor` by passing the current department number as a parameter. Start another loop and fetch the values of `emp_cursor` into variables, and print all the details retrieved from the `EMPLOYEES` table.

Notes

- Check whether `c_emp_cursor` is already open before opening the cursor.
- Use the appropriate cursor attribute for the exit condition.
- When the loop completes, print a line after you have displayed the details of each department, and close `c_emp_cursor`.

```
IF c_emp_cursor%ISOPEN THEN
  CLOSE c_emp_cursor;
END IF;
OPEN c_emp_cursor (v_current_deptno);
  LOOP
    FETCH c_emp_cursor INTO  v_ename,v_job,v_hiredate,v_sal;
    EXIT WHEN c_emp_cursor%NOTFOUND;
    DBMS_OUTPUT.PUT_LINE (v_ename || '    ' ||  v_job
                     || '   '  ||  v_hiredate || '   ' ||  v_sal);
  END LOOP;
  DBMS_OUTPUT.PUT_LINE('-------------------------------------------
-------------------------------------------');
         CLOSE c_emp_cursor;
```

f. End the first loop and close `c_dept_cursor`. Then end the executable section.

```
    END LOOP;
  CLOSE c_dept_cursor;
END;
```

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

g. Execute the script. The sample output is as follows:

```
Script Output
Task completed in 0.008 seconds
PL/SQL procedure successfully completed.

Department Number : 10  Department Name : Administration
--------------------------------------------------------------------------------
Department Number : 20  Department Name : Marketing
--------------------------------------------------------------------------------
Department Number : 30  Department Name : Purchasing
Raphaely    PU_MAN   07-DEC-10   11000
Khoo    PU_CLERK   18-MAY-11   3100
Baida    PU_CLERK   24-DEC-13   2900
Tobias    PU_CLERK   24-JUL-13   2800
Himuro    PU_CLERK   15-NOV-14   2600
Colmenares    PU_CLERK   10-AUG-15   2500
--------------------------------------------------------------------------------
Department Number : 40  Department Name : Human Resources
--------------------------------------------------------------------------------
Department Number : 50  Department Name : Shipping
--------------------------------------------------------------------------------
Department Number : 60  Department Name : IT
Hunold    IT_PROG   03-JAN-14   9000
Ernst    IT_PROG   21-MAY-15   6000
Austin    IT_PROG   25-JUN-13   4800
Pataballa    IT_PROG   05-FEB-14   4800
Lorentz    IT_PROG   07-FEB-15   4200
--------------------------------------------------------------------------------
Department Number : 70  Department Name : Public Relations
--------------------------------------------------------------------------------
Department Number : 80  Department Name : Sales
--------------------------------------------------------------------------------
Department Number : 90  Department Name : Executive
King    AD_PRES   17-JUN-11   24000
Kochhar    AD_VP   21-SEP-09   17000
De Haan    AD_VP   13-JAN-09   17000
--------------------------------------------------------------------------------
```

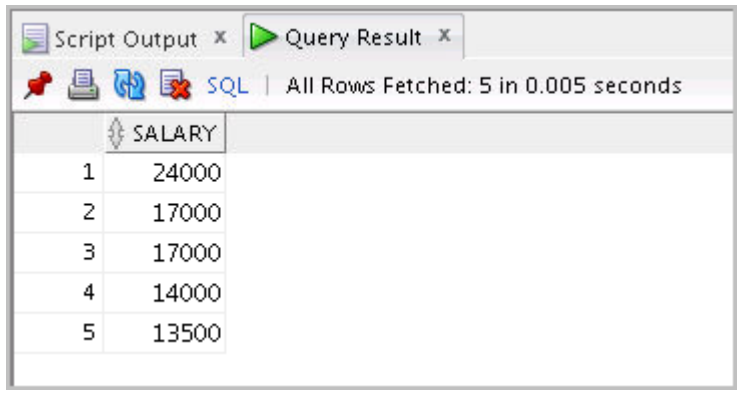
Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Practice 8-2: Using Explicit Cursors: Optional

If you have time, complete the following optional practice. Here, create a PL/SQL block that uses an explicit cursor to determine the top *n* salaries of employees.

1. Run the `lab_08-02.sql` script to create the `TOP_SALARIES` table for storing the salaries of the employees.
2. In the declarative section, declare the `v_num` variable of the `NUMBER` type that holds a number *n*, representing the number of top *n* earners from the `employees` table. For example, to view the top five salaries, enter `5`. Declare another variable `v_sal` of type `employees.salary`. Declare a cursor, `c_emp_cursor`, which retrieves the salaries of employees in descending order. Remember that the salaries should not be duplicated.
3. In the executable section, open the loop, fetch the top *n* salaries, and then insert them into the `TOP_SALARIES` table. You can use a simple loop to operate on the data. Also, try and use the `%ROWCOUNT` and `%FOUND` attributes for the exit condition.

   Note: Make sure that you add an exit condition to avoid having an infinite loop.
4. After inserting data into the `TOP_SALARIES` table, display the rows with a `SELECT` statement. The output shown represents the five highest salaries in the `EMPLOYEES` table.



	SALARY
1	24000
2	17000
3	17000
4	14000
5	13500

5. Test a variety of special cases such as `v_num = 0` or where `v_num` is greater than the number of employees in the `EMPLOYEES` table. Empty the `TOP_SALARIES` table after each test.

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Solution 8-2: Using Explicit Cursors: Optional

If you have time, complete the following optional exercise. Here, create a PL/SQL block that uses an explicit cursor to determine the top *n* salaries of employees.

1. Execute the `lab_08_02.sql` script to create a new table, `TOP_SALARIES`, for storing the salaries of the employees.
2. In the declarative section, declare a variable `v_num` of type `NUMBER` that holds a number *n*, representing the number of top *n* earners from the `EMPLOYEES` table. For example, to view the top five salaries, enter `5`. Declare another variable `v_sal` of type `employees.salary`. Declare a cursor, `c_emp_cursor`, which retrieves the salaries of employees in descending order. Remember that the salaries should not be duplicated.

```
DECLARE
  v_num         NUMBER(3) := 5;
  v_sal         employees.salary%TYPE;
  CURSOR        c_emp_cursor IS
    SELECT      salary
    FROM        employees
    ORDER BY    salary DESC;
```

3. In the executable section, open the loop, fetch the top *n* salaries, and then insert them into the `TOP_SALARIES` table. You can use a simple loop to operate on the data. Also, try and use the `%ROWCOUNT` and `%FOUND` attributes for the exit condition.

   Note: Make sure that you add an exit condition to avoid having an infinite loop.

```
BEGIN
  OPEN c_emp_cursor;
  FETCH c_emp_cursor INTO  v_sal;
  WHILE c_emp_cursor%ROWCOUNT <= v_num AND c_emp_cursor%FOUND LOOP
             INSERT INTO top_salaries (salary)
             VALUES (v_sal);
    FETCH c_emp_cursor INTO v_sal;
  END LOOP;
  CLOSE c_emp_cursor;
END;
```

4. After inserting data into the `TOP_SALARIES` table, display the rows with a `SELECT` statement. The output shown represents the five highest salaries in the `EMPLOYEES` table.

```
/
SELECT * FROM top_salaries;
```

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

The sample output is as follows:

```
SALARY
----------------------
24000
17000
17000
14000
13500
```

5. Test a variety of special cases such as `v_num = 0` or where `v_num` is greater than the number of employees in the `EMPLOYEES` table. Empty the `TOP_SALARIES` table after each test.

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.