**Quiz**

Implicit cursors are declared by PL/SQL implicitly for all DML and PL/SQL SELECT statements. The Oracle Server implicitly opens a cursor to process each SQL statement that is not associated with an explicitly declared cursor.

1. True

2. False

**Answer: True**

**Practice 7-1: 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 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).

**DECLARE**

**v\_deptno NUMBER := 20;**

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.

**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;**

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

Изображение выглядит как текст, квитанция, снимок экрана

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

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 the 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;**

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\_dept\_id departments.department\_id%TYPE;**

**v\_dept\_name departments.department\_name%TYPE;**

**v\_lname employees.last\_name%TYPE;**

**v\_job\_id employees.job\_id%TYPE;**

**v\_hire\_date 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 declared. Display the department number and department name. Use the appropriate cursor attribute to exit the loop.

**OPEN c\_dept\_cursor;**

**LOOP**

**FETCH c\_dept\_cursor INTO v\_dept\_id, v\_dept\_name;**

**EXIT WHEN c\_dept\_cursor%NOTFOUND;**

**DBMS\_OUTPUT.PUT\_LINE('Department Number : ' || v\_dept\_id ||**

**' Department Name: ' || v\_dept\_name);**

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.

**Note**

* 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\_dept\_id);**

**LOOP**

**FETCH c\_emp\_cursor INTO v\_lname, v\_job\_id, v\_hire\_date, v\_sal;**

**EXIT WHEN c\_emp\_cursor%NOTFOUND;**

**DBMS\_OUTPUT.PUT\_LINE(v\_lname || ' ' || v\_job\_id || ' ' || v\_hire\_date || ' ' || v\_sal);**

**END LOOP;**

**DBMS\_OUTPUT.PUT\_LINE('---------------------------------------------------');**

**CLOSE c\_emp\_cursor;**

**END LOOP;**

**CLOSE c\_dept\_cursor;**

f)  End the first loop and close c\_dept\_cursor. Then end the executable section.

**END LOOP;**

**CLOSE c\_dept\_cursor;**

**END;**

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

Изображение выглядит как текст

Автоматически созданное описаниеИзображение выглядит как стол

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

**Practice 7-2: 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\_07\_02.sql script to create the top\_salaries table for storing the salaries of the employees.

**CREATE TABLE top\_salaries (**

**salary NUMBER**

**);**

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 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 := 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 and fetch the top n salaries, and then insert them into 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;**

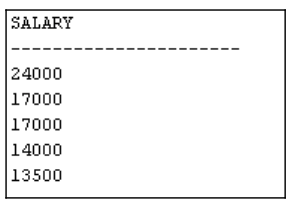
**CLOSE c\_emp\_cursor;**

**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;**



Изображение выглядит как текст

Автоматически созданное описание

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.

**DELETE FROM top\_salaries;**