Experiment No 21: Create Implicit and Explicit Cursors.

I. Practical Significance:

A cursor is a pointer to this context area. PL/SQL controls the context area through a cursor. A cursor holds the rows (one or more) returned by a SQL statement. The set of rows the cursor holds is referred to as the active set. You can name a cursor so that it could be referred to in a program to fetch and process the rows returned by the SQL statement, one at a time. There are two types of cursors namely: Implicit cursors, Explicit cursors.

II. INDUSTRY / EMPLOYER EXPECTED OUTCOME:

To implement PL/SQL programs to implement cursors.

III. COURSE LEVEL LEARNING OUTCOMES (COS):

CO4 - Implement PL/SQL codes for given application.

IV. LABORATORY LEARNING OUTCOME:

Implement PL/SQL program based on cursors.

V. Relevant Affective Domain related outcome(s)

a. Follow precautionary measures.
b. Follow installation steps.
c. Follow ethical practices.

VI. Relevant Theoretical Background:

Cursors provide a way to iterate through the rows returned by a query, one at a time, and perform various operations on the data. With cursors, we can access records individually, control them as needed, or display them on the console accordingly.

The PL/SQL programming language supports two kinds of cursors.

Implicit Cursor

Explicit Cursor

I. Implicit Cursors

An implicit cursor is generated by Oracle when DML statements such as INSERT, UPDATE, and DELETE are executed. Oracle offers some attributes through which programmers can perform operations on these cursors. Implicit cursors help to handle database transactions without requiring to manually create cursors every time. The attributes offered by Oracle are as follows:

a. **%FOUND:** This function returns a Boolean value of True,if INSERT, UPDATE, and DELETE, affect single or multiple rows. Similarly, it returns True if any SELECT statement also returns single or multiple rows. If neither of these conditions are met, the function will return False.

b. **%NOTFOUND:** This attribute operates in reverse to its counterpart, %FOUND. If a DML statement has not affected any rows or a SELECT statement does not return any results, then the %NOTFOUND attribute will be evaluated as True. Alternatively, it will return False when rows have been impacted by a DML statement or when the SELECT DML statement returns at least one result.

c. **%ISOPEN:** Regarding Implicit cursors, Oracle shut off the cursor immediately following the SQL statement's execution, resulting in a False return value.

d. **%ROWCOUNT:** The function returns the rows affected by DML statements such as INSERT, UPDATE, and DELETE. It can also provide the count of rows returned from SELECT INTO statements in PL/SQL code.

II. Explicit Cursor

An explicit cursor is declared in a program's declaration block and are particularly useful when working with SQL statements that produce multiple rows of results. To use an explicit cursor, many steps need to be implemented, including careful definition and initialization of the cursor within the program. If used correctly, explicit cursors can improve program performance and data accuracy.

Explicit Cursor Syntax

Cursor Cursor-Name

IS

Select-Statement;

Steps to Utilize Explicit Cursors

Step 1: Cursor Declaration

Syntax: CURSOR cursor_name IS SELECT statement;

Step 2: Open Cursor

Syntax: OPEN cursor_name;

Step 3: Fetching Cursor

Syntax: FETCH cursor_name INTO variable;

Step 4: Close Cursor

Syntax: CLOSE cursor_name;

VII. Required Resources/apparatus/equipment with specifications

Sr.No	Equipment Name with Broad Specifications	Relevant LLO Number
1	Computer system with all necessary components like; motherboard, random access memory (RAM), read-only memory (ROM), internal hard disk drives, Mouse, Keyboard, and RDBMS applications such as Oracle Express Edition,MySql,SQLite,Oracle Apex etc.	All

VIII. Procedure

1. Write program in notepad, save the program with .sql extension.
2. Take the program to SQL command prompt.
3. Execute and check the program output.

IX. Result(s)

In this Practical, we studied to create Implicit & Explicit Cursors.

X. Practical related questions (Provide space for answers)

Note: Below are a few sample questions for reference. Teacher must design more such questions to ensure the achievement of identified CO.

1. Distinguish between Implicit and Explicit cursors in oracle.
2. List advantages of using cursors in SQL programming.

(Space for answer)

* practical related questions

1] Distinguish between Implicit & explicit cursor in oracle.

Implicit cursor	Explicit cursor
1] A parameter is always a function of knowns and unknowns	1] A parameter is always a function of knowns
2] Solves Simultaneous Equations	2] Low convergence
3] Highly convergent	3] Very small time steps.
4] Large time steps	4] uses algebric differences equations
5] Large time steps	5] Computationally expensive

2] ⟶

Advantages of using cursor in SQL programming.

1] Row-by-Row Processing:
Cursors allow you to process individual rows in a result set one at a time, which can be beneficial for complex row-level operations.

2] Handling Large Data Sets:
They are useful for handling large datasets where batch processing might not be feasible

3] Enhanced Control: with cursors, you get more control over the result set navigation such as moving forward & backward through the rows.

4] Dynamic Sql:
They are well-suited for executing dynamic Sql queries that can change based on the data.

*Exercise

1] →

```
DECLARE
CURSOR comp_dept_cursor IS
SELECT Student_id, Student_name, Course
 FROM  Students
WHERE department = 'Computer';
V_Student_id Students.student_id %TYPE;
V_Student_name Students.student_name %TYPE;
BEGIN
OPEN comp_dept_cursor;
Loop
FETCH comp_dept_cursor INTO V_student_id,
V_student_name, V_course;
EXIT WHEN comp_dept_cursor %NOTFOUND;
DBMS_OUTPUT.PUT_LINE ('ID:' || V_student_id ||,
' Name:' || V_student_name || 'Course:' || V_course);
END Loop;
CLOSE comp_dept_cursor;
END;
/
```

→

```
DECLARE
CURSOR even_cursor IS
SELECT student_id, student_name, course
From( SELECT student_id, student_name, Course,
Row_Number() over (ORDER By student_id
```

```
AS row_num From students )
WHERE MOD (row_num , 2) = 0;
V_student_id students. student_id %.TYPE;
V_student_name students. student_name %.TYPE;
V_Course students. Course %. TYPE;
BEGIN
OPEN even_ cursor;
Loop
FETCH even_cursor INTO V_student_id, V_student
_name, V_Course ;
EXIT WHEN even_cursor %. NOTFOUND;
DBMS_OUTPUT. PUT_LINE ('ID:' || V_student_id ||
'Name:' || V_student_name || 'Course:' || V_Course);
END Loop;
Close even_cursor;
END;
/
```

3] ⟶

```
DECLARE
CURSOR high_price_cursor IS
SELECT COUNT (*) AS item_COUNT From store
WHERE Price > 10000;
V_item_Count NUMBER;
BEGIN
OPEN high_price_cursor;
FETCH high_Price_cursor INTO V_item_count;
DBMS_OUTPUT. PUT_line ('Number of items with price more
than 10000:' || V_item_count);
CLOSE high_price_cursor;
END;
```