Department of Software Engineering
Mehran University of Engineering and Technology, Jamshoro

Course: SW222 – Database Management & Administration			
Instructor	Ms Shafiya Qadeer	**Practical/Lab No.**	12
Date	2021	**CLOs**	3
Signature		**Assessment Score**	2 Marks

Topic	**To become familiar Cursors in PL/SQL**
Objectives	- To become familiar with Implicit & Explicit Cursors

Lab Discussion: Theoretical concepts and Procedural steps

Cursors

- A cursor is a temporary work area created in the system memory when a SQL statement is executed.
- This temporary work area is used to store the data retrieved from the database, and manipulate this data.
- A cursor can hold more than one row, but can process only one row at a time.
- The set of rows the cursor holds is called the active set.

2 TYPES OF CURSORS:

1. Implicit cursors
2. Explicit cursors

- These are created by default when DML statements like, INSERT, UPDATE, and DELETE statements are executed.
- They are also created when a SELECT statement that returns just one row is executed.
- In PL/SQL, you can refer to the most recent implicit cursor as the SQL cursor, which always has attributes such as %FOUND, %ISOPEN, %NOTFOUND, and %ROWCOUNT.

1. Implicit Cursors

- These are created by default when DML statements like, INSERT, UPDATE, and DELETE statements are executed.

- They are also created when a SELECT statement that returns just one row is executed.
- In PL/SQL, you can refer to the most recent implicit cursor as the SQL cursor, which always has attributes such as %FOUND, %ISOPEN, %NOTFOUND, and %ROWCOUNT.

S.No	Attribute & Description
1	**%FOUND** Returns TRUE if an INSERT, UPDATE, or DELETE statement affected one or more rows or a SELECT INTO statement returned one or more rows. Otherwise, it returns FALSE.
2	**%NOTFOUND** The logical opposite of %FOUND. It returns TRUE if an INSERT, UPDATE, or DELETE statement affected no rows, or a SELECT INTO statement returned no rows. Otherwise, it returns FALSE.
3	**%ISOPEN** Always returns FALSE for implicit cursors, because Oracle closes the SQL cursor automatically after executing its associated SQL statement.
4	**%ROWCOUNT** Returns the number of rows affected by an INSERT, UPDATE, or DELETE statement, or returned by a SELECT INTO statement.

2. Explicit Cursors

- Explicit cursors are programmer-defined cursors.
- An explicit cursor should be defined in the declaration section of the PL/SQL Block.
- It is created when you are executing a SELECT statement that returns more than one row.

SYNTAX:

CURSOR cursor_name IS select_statement;

- Working with an explicit cursor includes the following steps −

1. Declaring the cursor for initializing the memory
2. Opening the cursor for allocating the memory
3. Fetching the cursor for retrieving the data
4. Closing the cursor to release the allocated memory

- Declaring the cursor defines the cursor with a name and the associated SELECT statement.

- **For example:**

CURSOR e_employees IS

SELECT empno, ename, job FROM emp;

1. Declaring the Cursor

- Declaring the cursor defines the cursor with a name and the associated SELECT statement.

- **For example:**

CURSOR e_employees IS

SELECT empno, ename, job FROM emp;

2. Opening the Cursor

- Opening the cursor allocates the memory for the cursor and makes it ready for fetching the rows returned by the SQL statement into it.

- **For example:**

OPEN e_employees ;

3. Fetching the Cursor

- Fetching the cursor involves accessing one row at a time.

- **For example:**

FETCH e_employees INTO e_empno, e_ename, e_job;

4. Closing the Cursor

- Closing the cursor means releasing the allocated memory.

For example

CLOSE e_employees ;

```
DECLARE
   e_empno emp.empno%type;
   e_ename emp.ename%type;
   e_job emp.job%type;
   CURSOR e_employees IS
      SELECT empno, ename, job FROM emp;
BEGIN
   OPEN e_employees;
   LOOP
   FETCH e_employees into e_empno, e_ename, e_job;
      EXIT WHEN e_employees%notfound;
      dbms_output.put_line(e_empno || ' ' || e_ename || ' ' || e_job);
   END LOOP;
   CLOSE e_employees;
END;
```

Results Explain Describe Saved SQL History

```
70 LEVI MARIO TECNICO
71 CALVINO ANDREA PROFESSORE
72 FUMAROLA LUDOVICO SEGRETARIO
73 NERI ELENA DIRIGENTE
74 SANTE ANDREA DIRIGENTE
75 ESPOSITO ANDREA ISPETTORE
76 ECO MASSIMO DOTTORANDO
```

Lab Tasks

1. Run the examples given above