Group: - 1 from T4 Batch
0077 – Aakash Joshi
2039 – Akanksha Lokhande

- **Title:**
  Cursors: (All types: Implicit, Explicit, Cursor FOR Loop, Parameterized Cursor).

- **Date of completion:**

- **Problem statement:**
  Write a PL/SQL block of code using parameterized Cursor that will merge the data available in the newly created table N_RollCall with the data available in the table O_RollCall. If the data in the first table already exist in the second table, then that data should be skipped.

- **Objectives:**
  1. To develop Database programming skills.
  2. To develop basic Database administration skills.

- **Software and Hardware Requirements:**
  Software – 64-bit Open-Source Linux/Windows, Oracle.
  Hardware – Computer ,Mouse ,Keyboard , CPU

- **Theory :**

➢ Cursor in SQL-

To execute SQL statements, a work area is used by the Oracle engine for its internal processing and storing the information. This work area is private to SQL's operations. The 'Cursor' is the PL/SQL construct that allows the user to name the work area and access the stored information in it.

➢ Use of Cursor-

The major function of a cursor is to retrieve data, one row at a time, from a result set, unlike the SQL commands which operate on all the rows in the result set at one time. Cursors are used when the user needs to update records in a singleton fashion or in a row-by-row manner, in a database table.

The Data that is stored in the Cursor is called the Active Data Set. Oracle DBMS has another predefined area in the main memory Set, within which the cursors are opened. Hence the size of the cursor is limited by the size of this pre-defined area.

➢ Cursor Actions-

- Declare Cursor: A cursor is declared by defining the SQL statement that returns a result set.
- Open: A Cursor is opened and populated by executing the SQL statement defined by the cursor.
- Fetch: When the cursor is opened, rows can be fetched from the cursor one by one or in a block to perform data manipulation.
- Close: After data manipulation, close the cursor explicitly.
- Deallocate: Finally, delete the cursor definition and release all the system resources associated with the cursor.

➢ Types of Cursors-

Cursors are classified depending on the circumstances in which they are opened.

- Implicit Cursor: If the Oracle engine opened a cursor for its internal processing it is known as an Implicit Cursor. It is created "automatically" for the user by Oracle when a query is executed and is simpler to code.
- Explicit Cursor: A Cursor can also be opened for processing data through a PL/SQL block, on demand. Such a user-defined cursor is known as an Explicit Cursor.

  An explicit cursor is defined in the declaration section of the PL/SQL Block. It is created on a SELECT Statement which returns more than one row. A suitable name for the cursor.

➢ General syntax for creating a cursor-

CURSOR cursor_name IS select_statement;

cursor_name – A suitable name for the cursor.

select_statement – A select query which returns multiple rows

- How to use Explicit Cursor?-

There are four steps in using an Explicit Cursor.

1. DECLARE the cursor in the Declaration section.
2. OPEN the cursor in the Execution Section.
3. FETCH the data from the cursor into PL/SQL variables or records in the Execution Section.
4. CLOSE the cursor in the Execution Section before you end the PL/SQL Block.

Syntax:

```
DECLARE variables;
 records;
 create a cursor;
 BEGIN
OPEN cursor;
FETCH cursor;
 process the records;
 CLOSE cursor;
END;
```

Code:

```
DECLARE
  CURSOR c_rollcall IS
    SELECT * FROM n_rollcall;

  v_n_rollcall c_rollcall%ROWTYPE;

  CURSOR c_orollcall(v_username VARCHAR2) IS
    SELECT * FROM o_rollcall WHERE username = v_username;

  v_exists NUMBER := 0;

BEGIN

  OPEN c_rollcall;

  LOOP
    FETCH c_rollcall INTO v_n_rollcall;
```

```
    EXIT WHEN c_rollcall%NOTFOUND;

    OPEN c_orollcall(v_n_rollcall.username);

    FETCH c_orollcall INTO v_exists;

    IF c_orollcall%NOTFOUND THEN
      -- User does not exist in o_rollcall
      INSERT INTO o_rollcall VALUES v_n_rollcall;
    END IF;

    CLOSE c_orollcall;

  END LOOP;

  CLOSE c_rollcall;

END;
```

Output:

```
n_rollcall table:

+----------------+--------+
| USERNAME | ROLE |
+----------------+--------+
| john           | mod |
| sarah          | vip    |
| dave           | sub   |
+----------------+--------+

o_rollcall table before merge:



+----------------+--------+
| USERNAME | ROLE |
+----------------+---------+
| john           |  mod  |
+----------------+---------+
```

o_rollcall table after merge:

```
+-----------------+---------+
| USERNAME | ROLE |
+-----------------+---------+
| john            | mod  |
| sarah           |    vip  |
| dave            |   sub  |
+-----------------+---------+
```