D.Y. Patil Academic Education Excellence Federation's
Dr.D.Y. Patil Technical Campus
(Engineering and MCA)
(Approved by AICTE, DTE-Govt of Maharashtra, Affiliated to Savitribai Phule Pune University, Pune)
Sr.No.32/1/A/7, Near Talegaon Railway Station, A/P Varale, Talegaon Dabhade
Tal-Maval, Dist-Pune 410507
Ph. No.9920141406,9309228311,7666829653,9307909501
Website: www.dypatiltcs.com

Lab Manual – PL/SQL Cursors – Merging Data using Parameterized Cursor

Experiment No. 6

Title:

PL/SQL Cursors – Merging Data using Parameterized Cursor

Objectives:

- To understand the concept and types of cursors in PL/SQL.
- To implement **implicit**, **explicit**, **cursor FOR loop**, and **parameterized cursors**.
- To write a PL/SQL block that merges data between two roll call tables using parameterized cursors.
- To skip insertion of duplicate records while merging.

Problem Statement:

Write a PL/SQL block using a **parameterized cursor** that merges data from a newly created table **N_RollCall** into an existing table **O_RollCall**.

- If a record from **N_RollCall** already exists in **O_RollCall**, skip that record.
- If the record does not exist, insert it into **O_RollCall**.
- Use parameterized cursors to fetch and check records.

Software and Hardware Requirements:

- **Software:** Oracle Database 10g/11g/12c or higher, SQL*Plus / Oracle SQL Developer
- **Hardware:** Intel i3/i5 Processor, 4GB+ RAM, 500GB HDD, Windows/Linux OS

Theory :

- **Cursor:** A pointer to the context area that stores the result of a query.
- **Types of Cursors:**
    1. **Implicit Cursor** – Automatically created for single row queries (INSERT, UPDATE, DELETE).
    2. **Explicit Cursor** – Declared by the programmer for multi-row queries.
    3. **Cursor FOR Loop** – Simplifies cursor management by implicitly opening, fetching, and closing.
    4. **Parameterized Cursor** – Accepts parameters at runtime, useful for dynamic queries.
- **This experiment uses a parameterized cursor** to merge data between two roll call tables while avoiding duplicates.

Algorithm:

1. Create tables **O_RollCall** (old roll call) and **N_RollCall** (new roll call).
2. Define a **parameterized cursor** that takes Roll_no as input and checks if it exists in **O_RollCall**.
3. Loop through each record in **N_RollCall**.
4. For each record:
    - Use the cursor to check if the Roll_no already exists in **O_RollCall**.
    - If not found → insert into **O_RollCall**.
    - If found → skip.
5. Display appropriate messages using DBMS_OUTPUT.

PL/SQL Code (Parameterized Cursor Example):

```
-- Step 1: Create tables
CREATE TABLE O_RollCall (
   Roll_no NUMBER PRIMARY KEY,
   Name VARCHAR2(50)
```

```
);

CREATE TABLE N_RollCall (
   Roll_no NUMBER,
   Name VARCHAR2(50)
);

-- Step 2: Insert sample data
INSERT INTO O_RollCall VALUES (1, 'Amit');
INSERT INTO O_RollCall VALUES (2, 'Priya');

INSERT INTO N_RollCall VALUES (2, 'Priya'); -- duplicate
INSERT INTO N_RollCall VALUES (3, 'Rahul'); -- new record
INSERT INTO N_RollCall VALUES (4, 'Sneha'); -- new record
COMMIT;

-- Step 3: PL/SQL Block with Parameterized Cursor
DECLARE
   CURSOR c_check(p_roll NUMBER) IS
      SELECT Roll_no FROM O_RollCall WHERE Roll_no = p_roll;

   v_roll N_RollCall.Roll_no%TYPE;
   v_name N_RollCall.Name%TYPE;
   v_dummy O_RollCall.Roll_no%TYPE;
BEGIN
   FOR rec IN (SELECT * FROM N_RollCall) LOOP
      v_roll := rec.Roll_no;
      v_name := rec.Name;

      OPEN c_check(v_roll);
      FETCH c_check INTO v_dummy;

      IF c_check%NOTFOUND THEN
         INSERT INTO O_RollCall(Roll_no, Name)
         VALUES (v_roll, v_name);
         DBMS_OUTPUT.PUT_LINE('Inserted: ' || v_name);
      ELSE
         DBMS_OUTPUT.PUT_LINE('Skipped (Already Exists): ' || v_name);
```

```
      END IF;

      CLOSE c_check;
   END LOOP;
END;
/
```

Test Cases:

Case	Input Table (N_RollCall)	Expected Action	Result in O_RollCall
1	(2, Priya)	Already exists → Skip	No change
2	(3, Rahul)	Not exists → Insert	Added (3, Rahul)
3	(4, Sneha)	Not exists → Insert	Added (4, Sneha)

Test Data Set:

-- O_RollCall initially
(1, Amit)
(2, Priya)

-- N_RollCall
(2, Priya)
(3, Rahul)
(4, Sneha)

Conclusion / Analysis:

- Successfully demonstrated the use of a **parameterized cursor** in PL/SQL.
- The program merges new data into the old roll call while avoiding duplicates.
- This experiment also reinforces knowledge of **cursor operations (OPEN, FETCH, CLOSE)** and condition handling.
- The approach can be extended to other real-time scenarios such as merging customer data, attendance lists, or transaction records.
