



## Lab sheet – 10 (L9+L10)

### Cursor

**Name: Dishant Naik**

**Reg no.: 21MIA1127**

A cursor in PL/SQL gives a name and acts as a pointer to the area of work called a context area and then uses its information. It keeps the number of rows processed by the SQL statement. These rows are called as an active set. The size of the active set is equal to the count of the rows that meet the condition.

```
CREATE TABLE TUTOR (CODE INT NOT NULL,  
                      SUBJECT VARCHAR(15) NOT NULL,  
                      TEACHER VARCHAR(15),  
                      REVIEWS VARCHAR(10) NOT NULL,  
                      PRIMARY KEY (CODE));
```

```
INSERT INTO TUTOR (CODE,SUBJECT,TEACHER,REVIEWS)  
VALUES (1, 'Automation', 'CV RAMAN', 'five stars');
```

```
INSERT INTO TUTOR (CODE,SUBJECT,TEACHER,REVIEWS)  
VALUES (4, 'PLSQL', 'APJ', 'four stars');
```

```
INSERT INTO TUTOR (CODE,SUBJECT,TEACHER,REVIEWS)  
VALUES (2, 'Performance', 'Aryabhata', 'four stars');
```

```
SELECT * FROM TUTOR;
```

CODE	SUBJECT	TEACHER	REVIEWS
1	Automation	CV RAMAN	five stars
4	PLSQL	APJ	four stars
2	Performance	Aryabhata	four stars

### Implicit Cursors

The implicit cursors are allocated by Oracle by default while executing SQL statements. It holds the affected rows by the DML operations like UPDATE, DELETE and INSERT. Thus, implicit cursors are used when we don't have an explicit cursor in place.

While we are inserting a row, the cursor keeps that particular data. Similarly, for deletion and updating operations, the affected rows are stored by the cursors. The implicit cursors are not given any names and hence cannot be manipulated by the developers and the data contained on it cannot be used anywhere.

## Implementation Code- with the implicit cursor:

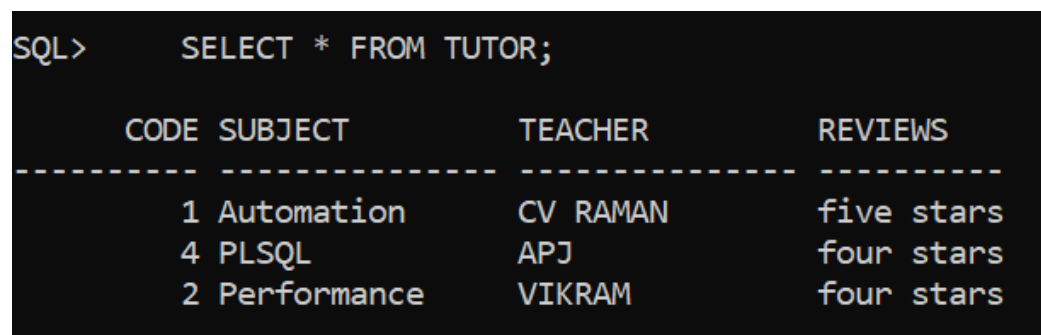
```
DECLARE
    total_count number(30);
BEGIN
    --updating a row
    UPDATE TUTOR
    SET TEACHER = 'VIKRAM' where CODE = 2;
    -- result in boolean, true returned if no rows affected
    IF sql%notfound THEN
        dbms_output.put_line('no subjects fetched');

        -- result in boolean, true returned if any rows affected
    ELSIF sql%found THEN

        -- count the number of rows affected rows affected
        total_count := sql%rowcount;
        dbms_output.put_line( total_count || ' teacher name updated ');
    END IF;
END;
/
```

Let us now verify the changes reflected in the table named TUTOR.

SELECT \* FROM TUTOR;



The screenshot shows a terminal window with a black background. At the top, the command 'SQL> SELECT \* FROM TUTOR;' is entered. Below it, the output is displayed as a table with four columns: CODE, SUBJECT, TEACHER, and REVIEWS. The table has three rows of data. The first row shows CODE 1, SUBJECT Automation, TEACHER CV RAMAN, and REVIEWS five stars. The second row shows CODE 4, SUBJECT PLSQL, TEACHER APJ, and REVIEWS four stars. The third row shows CODE 2, SUBJECT Performance, TEACHER VIKRAM, and REVIEWS four stars. The table is formatted with dashed lines for the header and data rows.

CODE	SUBJECT	TEACHER	REVIEWS
1	Automation	CV RAMAN	five stars
4	PLSQL	APJ	four stars
2	Performance	VIKRAM	four stars

## Explicit Cursors

The developers can have their own user-defined context area to run DML operations. Thus they can exercise more power over it. The declaration section of the PL/SQL block of code contains explicit cursors. It is normally built on SELECT operations that fetch multiple rows.

### Syntax of explicit cursor:

```
DECLARE
    CURSOR <<cursor name>> IS <<select statement>>
    <<Cursor variable>>
BEGIN
    OPEN <<cursor name>>;
    FETCH <<cursor name>> INTO <Cursor variable>;
    .
    .
    CLOSE <cursor name>;
END;
```

## Implementation Code: with explicit cursor:

```
SET SERVEROUTPUT ON;
DECLARE
    -- cursor declaration
    CURSOR t_tutorials IS
    SELECT code, subject, teacher FROM Tutor;
    t_code Tutor.code%type;
    t_subject Tutor.subject%type;
    t_teacher Tutor.teacher%type;

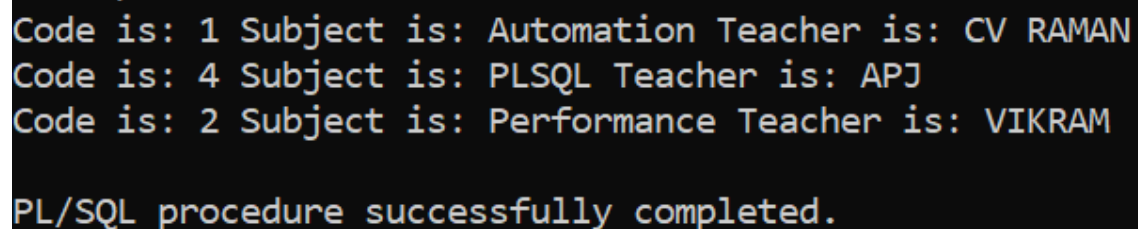
BEGIN

    -- opening a cursor
    OPEN t_tutorials;
    LOOP

        -- fetching values from cursor
        FETCH t_tutorials INTO t_code, t_subject, t_teacher;
        EXIT WHEN t_tutorials%NOTFOUND;

        -- printing in console
        dbms_output.put_line('Code is: ' || t_code || ' ' || 'Subject is: ' ||
t_subject || ' Teacher is: ' || t_teacher);
    END LOOP;
    CLOSE t_tutorials;
END;
/
```

The output of the above code should be:



```
Code is: 1 Subject is: Automation Teacher is: CV RAMAN
Code is: 4 Subject is: PLSQL Teacher is: APJ
Code is: 2 Subject is: Performance Teacher is: VIKRAM

PL/SQL procedure successfully completed.
```

## Notes:

Explicit Cursor works on the processes listed below:

**#1) Cursor declaration for memory initialization.** Here, a named context area is created which serves as a cursor name.

### Syntax:

```
CURSOR tutorial_s IS
```

```
SELECT code FROM TUTORIAL;
```

**#2) Cursor opening for memory allocation.** A cursor is now available for fetching the updated rows from the database.

**Syntax:**

OPEN tutorial\_s;

**#3) Cursor is fetched for getting the data.** After the SELECT operation is done, the rows obtained are put in the memory allocated and these are now considered as active sets. The cursor can access one row at a time.

**Syntax:**

FETCH tutorial\_s INTO c\_code;

**#4) Cursor is finally closed to free the allocated memory.** As all the records are obtained one by one, the cursor is closed to release context area memory.

**Syntax:**

CLOSE tutorial\_s;

## Exercise Questions

Table: EMP			
Column Name	Data Type	Size	Description
Empno	NUMBER	4	Employee's Identification Number
Ename	VARCHAR2	30	Employee's Name
Job	VARCHAR2	15	Employee's Designation
Sal	NUMBER	8,2	Employee's Salary
DeptNo	NUMBER	2	Employee's Department id
Commission	NUMBER	7,2	Employee's Commission

(a) Write a PL/SQL code to display the Empno, Ename and Job of employees of DeptNo 10 with CURSOR FOR LOOP Statement.

**Answer:**

```
SQL> declare
  2  cursor c1 is select empno,ename,job from employee1_MIA1127 where deptno=10;
  3  begin
  4  for rec in c1 loop
  5  dbms_output.put_line('empno: '||rec.empno);
  6  dbms_output.put_line('ename: '||rec.ename);
  7  dbms_output.put_line('job: '||rec.job);
  8  end loop;
  9  end;
 10  /

PL/SQL procedure successfully completed.
```

(b) Create a Cursor to increase the salary of employees according to the following conditions:

Salary of DeptNo 10 employees increased by 1000.

Salary of DeptNo 20 employees increased by 500.

Salary of DeptNo 30 employees increased by 800.

Also, store the EmpNo, old salary and new salary in a Table TEMP having three columns Empid, Old and New.

**Answer:**

```
SQL> declare cursor c is select * from employee1_MIA1127;
  2  begin
  3  for rec in c
  4  loop
  5  if rec.deptno=10
  6  then update employee1_MIA1127 set sal=sal+1000 where empno=rec.empno;
  7  elsif rec.deptno=20
  8  then update employee1_MIA1127 set sal=sal+500 where empno=rec.empno;
  9  elsif rec.deptno=30
 10  then update employee1_MIA1127 set sal=sal+800 where empno=rec.empno;
 11  end if;
 12  end loop;
 13  end;
 14  /

PL/SQL procedure successfully completed.
```

c) Write a program in PL/SQL to create a cursor displays the name and salary of each employee in the EMPLOYEES table whose salary is less than average salary of all employee.

**Answer:**

```
SQL> declare cursor c is select * from employee1_MIA1127;
  2  x number(8,2);
  3  begin
  4  select avg(sal) into x from employee1_MIA1127;
  5  for rec in c
  6  loop
  7  if rec.sal<x
  8  then
  9  dbms_output.put_line(rec.ename || rec.sal);
 10  end if;
 11  end loop;
 12  end;
 13  /

PL/SQL procedure successfully completed.
```

## TABLE

```
CREATE TABLE emp (  
    empno          NUMBER(4) NOT NULL CONSTRAINT emp_pk PRIMARY KEY,  
    ename          VARCHAR2(10),  
    job            VARCHAR2(9),  
    sal            NUMBER(7,2) CONSTRAINT emp_sal_ck CHECK (sal > 0),  
    deptno         NUMBER(2),  
    comm           NUMBER(7,2),  
);
```

\*\*\*\*\*

```
CREATE TABLE employee1 (  
    empno          NUMBER(4) NOT NULL CONSTRAINT emp_pk PRIMARY KEY,  
    ename          VARCHAR2(10),  
    job            VARCHAR2(9),  
    mgr            NUMBER(4),  
    hiredate       DATE,  
    sal            NUMBER(7,2) CONSTRAINT emp_sal_ck CHECK (sal > 0),  
    comm           NUMBER(7,2),  
    deptno         NUMBER(2));
```

Modify the below values as per the emp table domain requirements:

```
INSERT INTO emp VALUES (7369,'SMITH','CLERK',7902,'17-DEC-  
80',800,NULL,20);  
INSERT INTO emp VALUES (7499,'ALLEN','SALESMAN',7698,'20-FEB-  
81',1600,300,30);  
INSERT INTO emp VALUES (7521,'WARD','SALESMAN',7698,'22-FEB-  
81',1250,500,30);  
INSERT INTO emp VALUES (7566,'JONES','MANAGER',7839,'02-APR-  
81',2975,NULL,20);  
INSERT INTO emp VALUES (7654,'MARTIN','SALESMAN',7698,'28-SEP-  
81',1250,1400,30);  
INSERT INTO emp VALUES (7698,'BLAKE','MANAGER',7839,'01-MAY-  
81',2850,NULL,30);  
INSERT INTO emp VALUES (7782,'CLARK','MANAGER',7839,'09-JUN-  
81',2450,NULL,10);  
INSERT INTO emp VALUES (7788,'SCOTT','ANALYST',7566,'19-APR-  
87',3000,NULL,20);  
INSERT INTO emp VALUES (7839,'KING','PRESIDENT',NULL,'17-NOV-  
81',5000,NULL,10);  
INSERT INTO emp VALUES (7844,'TURNER','SALESMAN',7698,'08-SEP-  
81',1500,0,30);  
INSERT INTO emp VALUES (7876,'ADAMS','CLERK',7788,'23-MAY-  
87',1100,NULL,20);  
INSERT INTO emp VALUES (7900,'JAMES','CLERK',7698,'03-DEC-  
81',950,NULL,30);  
INSERT INTO emp VALUES (7902,'FORD','ANALYST',7566,'03-DEC-  
81',3000,NULL,20);  
INSERT INTO emp VALUES (7934,'MILLER','CLERK',7782,'23-JAN-  
82',1300,NULL,10);
```

\*\*\*\*\*