*** EXPERIMENT NO: 07 ***

Author: Atharva Paliwal

Roll No: 40 [5B]

Date: 06-November-2020

AIM: Write and execute SQL Programs for retrieving data using a cursor and to demonstrate various cursors.

PROBLEM STATEMENT:

Using the relation schemata established in Experiments - 02, 03, and 05, create and execute SQL programs for retrieving data using cursors.

****

QUERY 01: Write a SQL code to compile and execute an anonymous block which declares a cursor - FACULTY. The cursor buffers the records comprising - EmployeeID, Employee Name (FNAME and LNAME combined) and Designation for the Designation entered by the user. You may use either EMPLOYEE table or EMPP table for this cursor and print the buffered records. Use %NOTFOUND variable to enable cursor exit.

****

```
DECLARE
   ID EMPP.EID%TYPE;
   NAME EMPP.ENAME%TYPE;
   DESG EMPP.DESIGNATION%TYPE;
   CURSOR FACULTY IS
      SELECT EID, ENAME, DESIGNATION FROM EMPP
      WHERE UPPER(DESIGNATION) LIKE UPPER('&DESIGNATION');
BEGIN
   OPEN FACULTY;
   LOOP
      FETCH FACULTY INTO ID, NAME, DESG;
      EXIT WHEN FACULTY%NOTFOUND;
      DBMS_OUTPUT.PUT_LINE(ID||' '||RPAD(NAME,12)||' '||RPAD(DESG,12));
   END LOOP;
   CLOSE FACULTY;
END;
/
```

```
Enter value for designation: PROFESSOR
old 7: WHERE UPPER(DESIGNATION) LIKE UPPER('&DESIGNATION');
new 7: WHERE UPPER(DESIGNATION) LIKE UPPER('PROFESSOR');
7102  Samantha Jon  Professor
7101  Eugene Sabat  Professor
7103  Alexander Ll  Professor
7104  Simon Downin  Professor
```

****

QUERY 02: Modify the cursor in Query-01 as FACULTY_CFL which uses the cursor FOR loop to buffering and displaying the records (as mentioned) when employee designation is entered by the user. Use a variation of cursor FOR loop to include the ROWCOUNT variable to print serial number for the displayed records.

****

```
DECLARE
   CURSOR FACULTY_CFL IS
      SELECT EID, ENAME, DESIGNATION FROM EMPP
      WHERE UPPER(DESIGNATION) LIKE UPPER('&DESIGNATION');
BEGIN
   FOR FREC IN FACULTY_CFL LOOP
      DBMS_OUTPUT.PUT_LINE( TO_CHAR(FACULTY_CFL%ROWCOUNT)||'
      '||RPAD(FREC.EID,10)||' '||RPAD(FREC.ENAME,10)||'
      '||RPAD(FREC.DESIGNATION,10));
   END LOOP;
END;
/
```

```
Enter value for designation: PROFESSOR
old 4: WHERE UPPER(DESIGNATION) LIKE UPPER('&DESIGNATION');
new 4: WHERE UPPER(DESIGNATION) LIKE UPPER('PROFESSOR');
1 7102      Samantha J  Professor
2 7101      Eugene Sab  Professor
3 7103      Alexander   Professor
4 7104      Simon Down  Professor
```

****

QUERY 03: Modify the cursor FACULTY_CFL_A to display only those many records as desired by the user. Use %ROWCOUNT to enable the cursor to ensure this.

****

```
DECLARE
   CURSOR FACULTY_CFL_A IS
      SELECT EID, ENAME, DESIGNATION FROM EMPP
      WHERE UPPER(DESIGNATION) LIKE UPPER('&DESIGNATION');
   NUMROW NUMBER(1) := &NUMBER_OF_ROWS;
BEGIN
   FOR FREC IN FACULTY_CFL_A LOOP
      DBMS_OUTPUT.PUT_LINE( TO_CHAR(FACULTY_CFL_A%ROWCOUNT)||'
      '||RPAD(FREC.EID,10)||' '||RPAD(FREC.ENAME,10)||'
      '||RPAD(FREC.DESIGNATION,10));
      IF NUMROW=FACULTY_CFL_A%ROWCOUNT THEN
         EXIT;
      END IF;
   END LOOP;
END;
/
```

```
Enter value for designation: PROFESSOR
old 4: WHERE UPPER(DESIGNATION) LIKE UPPER('&DESIGNATION');
new 4: WHERE UPPER(DESIGNATION) LIKE UPPER('PROFESSOR');
Enter value for number_of_rows: 3
old 5: NUMROW NUMBER(1) := &NUMBER_OF_ROWS;
new 5: NUMROW NUMBER(1) := 3;
1 7102      Samantha J  Professor
2 7101      Eugene Sab  Professor
3 7103      Alexander   Professor

Enter value for designation: PROFESSOR
old 4: WHERE UPPER(DESIGNATION) LIKE UPPER('&DESIGNATION');
new 4: WHERE UPPER(DESIGNATION) LIKE UPPER('PROFESSOR');
Enter value for number_of_rows: 5
old 5: NUMROW NUMBER(1) := &NUMBER_OF_ROWS;
new 5: NUMROW NUMBER(1) := 5;
1 7102      Samantha J  Professor
2 7101      Eugene Sab  Professor
3 7103      Alexander   Professor
4 7104      Simon Down  Professor
```

****

QUERY 04: Write a SQL code to compile and execute an anonymous block which declares a cursor - EMP_SAL_INFO (Salary, Designation). Let the default values for salary and designation be 75000 and ‚Asst. Professor‛ respectively. The cursor buffers the records comprising - Employee ID, Employee Name (FNAME and LNAME combined), Designation and Salary for the Salary and Designation entered by the user. Use EMPLOYEE table for this cursor. Use this cursor to print the buffered records.

****

```
DECLARE
   CURSOR EMP_SAL_INFO(SAL EMPLOYEE.SALARY%TYPE DEFAULT 75000,
      DESG EMPLOYEE.DESIGNATION%TYPE DEFAULT 'Asst. Professor') IS
      SELECT ENO, FNAME||' '||LNAME AS NAME, DESIGNATION, SALARY FROM
   EMPLOYEE
      WHERE SALARY>SAL AND UPPER(DESIGNATION)=UPPER(DESG);
      E_SAL EMPLOYEE.SALARY%TYPE;
      E_DESG EMPLOYEE.DESIGNATION%TYPE;
BEGIN
   DBMS_OUTPUT.PUT_LINE( CHR(10));
   DBMS_OUTPUT.PUT_LINE('WITH DEFAULT VALUES: ');
   DBMS_OUTPUT.PUT_LINE( CHR(10));
   FOR EE IN EMP_SAL_INFO() LOOP
      DBMS_OUTPUT.PUT_LINE(EE.ENO||' '||RPAD(EE.NAME, 15)||' '||
      RPAD(EE.DESIGNATION, 15)||' '||LPAD(EE.SALARY,15));
   END LOOP;
   DBMS_OUTPUT.PUT_LINE(CHR(10));
   E_SAL:=&SALARY;
   DBMS_OUTPUT.PUT_LINE('WITH SOME DEFAULT VALUES: ');
   DBMS_OUTPUT.PUT_LINE( CHR(10));
   FOR EE IN EMP_SAL_INFO(E_SAL) LOOP
      DBMS_OUTPUT.PUT_LINE(EE.ENO||' '||RPAD(EE.NAME, 15)||' '||
      RPAD(EE.DESIGNATION, 15)||' '||LPAD(EE.SALARY,15));
   END LOOP;
   DBMS_OUTPUT.PUT_LINE(CHR(10));
   E_SAL:=&SALARY;
   E_DESG:='&DESIGNATION';
   DBMS_OUTPUT.PUT_LINE('WITH ALL SUPPLIED DEFAULT VALUES: ');
   DBMS_OUTPUT.PUT_LINE(CHR(10));
   FOR EE IN EMP_SAL_INFO(E_SAL, E_DESG) LOOP
      DBMS_OUTPUT.PUT_LINE(EE.ENO||' '||RPAD(EE.NAME, 15)||' '||
      RPAD(EE.DESIGNATION, 15)||' '||LPAD(EE.SALARY,15));
   END LOOP;
END;
/
```

```
Enter value for salary: 88000
old 19: E_SAL:=&SALARY;
new 19: E_SAL:=88000;
Enter value for salary: 120000
old 29: E_SAL:=&SALARY;
new 29: E_SAL:=120000;
Enter value for designation: Asso. Professor
old 30: E_DESG:='&DESIGNATION';
new 30: E_DESG:='Asso. Professor';

WITH DEFAULT VALUES:
7109 Martina Jacobso  Asst. Professor     91000
7110 William Smithfi  Asst. Professor     86400

WITH SOME DEFAULT VALUES:
7109 Martina Jacobso  Asst. Professor     91000

WITH ALL SUPPLIED DEFAULT VALUES:
7107 Christov Plutni  Asso. Professor     127400
7105 Christina Mulbo  Asso. Professor     127400
7106 Dolly Silverlin  Asso. Professor     127400
```

****

QUERY 05: Write SQL code to compile and execute a procedure – PRINT_EMPLOYEE which receives employee salary as input and prints the following particulars – employee number, employee name and salary, for employees whose salary exceeds the inputted salary. You must use a cursor - SAL_CURSOR, to buffer required result-set for bulk collect. Use TYPE statement to declare and instantiate array variables. You may also try using %ROWCOUNT. Use EMPP table as source. You may also use EMPLOYEE table.

****

```
DECLARE
   TYPE NUM_ARRAY IS VARRAY(10000) OF NUMBER;
   TYPE STR_ARRAY IS VARRAY(10000) OF VARCHAR2(50);
   TYPE NUM2_ARRAY IS VARRAY(10000) OF NUMBER;
   ENO_ARR NUM_ARRAY;
   ENAME_ARR STR_ARRAY;
   ESAL_ARR NUM2_ARRAY;
   CURSOR SAL_CURSOR IS
      SELECT ENO, FNAME||' '||LNAME AS ENAME, SALARY FROM EMPLOYEE
      WHERE SALARY>&SALARY;
BEGIN
   OPEN SAL_CURSOR;
   FETCH SAL_CURSOR
      BULK COLLECT INTO ENO_ARR, ENAME_ARR, ESAL_ARR;
   CLOSE SAL_CURSOR;
```

```
      FOR KNT IN ENO_ARR.FIRST .. ENO_ARR.LAST LOOP
      DBMS_OUTPUT.PUT_LINE(ENO_ARR(KNT)||' '||RPAD(ENAME_ARR(KNT), 15)
      ||' '||LPAD(ESAL_ARR(KNT), 15));
   END LOOP;
END;
/
```

```
Enter value for salary: 50000
old 12: WHERE SALARY>&SALARY;
new 12: WHERE SALARY>50000;
7102 Samantha Jones     146500
7101 Eugene Sabatini    150000
7103 Alexander Lloyd    148000
7104 Simon Downing            138400
7107 Christov Plutni    127400
7105 Christina Mulbo    127400
7106 Dolly Silverlin    127400
7108 Ellena Sanchez     119700
7109 Martina Jacobso     91000
7110 William Smithfi     86400

Enter value for salary: 125000
old 12: WHERE SALARY>&SALARY;
new 12: WHERE SALARY>125000;
7102 Samantha Jones     146500
7101 Eugene Sabatini    150000
7103 Alexander Lloyd    148000
7104 Simon Downing            138400
7107 Christov Plutni    127400
7105 Christina Mulbo    127400
7106 Dolly Silverlin    127400

Enter value for salary: 148000
old 12: WHERE SALARY>&SALARY;
new 12: WHERE SALARY>148000;
7101 Eugene Sabatini    150000
```

VIVA-VOICE

Q1. What is a cursor? List the steps associated with implementing a cursor.

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.

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

Q2.. What is an "active set"?

A cursor holds the rows (one or more) returned by a SQL statement. The set of rows
the cursor holds are 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.

Q3. What is a cursor FOR loop? Why it is advantageous?

The cursor FOR LOOP statement implicitly declares its loop index as a record variable of the row type that a specified cursor returns, and then opens a cursor.
With each iteration, the cursor FOR LOOP statement fetches a row from the result
set
into the record. When there are no more rows to fetch, the cursor FOR LOOP statement
closes the cursor. The cursor also closes if a statement inside the loop transfers
control outside the loop or raises an exception.
ADVANTAGES OF CURSORS USING FOR LOOP
1.No need to open the cursor.
2.Fetch the records automatically.
3.It automatically checks the end of rows.
4.It automatically closes the cursor.
5.No need to declare the variables.
6.code size will be decreased.
7.execution will be faster.
8.less fetching time.
9.It is collection of information from cursor to a variable.

Q4. Why it is a good practice to close a cursor?

The CLOSE statement closes a cursor or cursor variable, thereby allowing its resources to be reused.
After closing a cursor, you can reopen it with the OPEN statement. You must close
A cursor before reopening it.
After closing a cursor variable, you can reopen it with the OPEN-FOR statement.
You need not close a cursor variable before reopening it.
When a cursor is opened, Oracle runs the query to generate the results and
Positions the cursor before the first row of the result set. However, a
cursor can only be opened if it is not already open, attempting to open a
cursor that is already open generates a "CURSOR_ALREADY_OPEN" exception. In
other words if you declare a cursor and open it, if you try to open it again
without closing it, Oracle raises an exception.

INFERENCES

- We learnt about cursors.
- We learnt how to retrieve data using cursors and how to work using them.

**************************** END ********************************