CURSORS

Defination: A cursor is a handle, or pointer, to the context area. Through the cursor, a PL/SQL program can control the context area and what happens to it as the statement is processed.
Two important features about the cursor are
Cursors allow you to fetch and process rows returned by a SELECT statement, one row at a time.
A cursor is named so that it can be referenced.

OR

It is a temporary area for work in memory system while the execution of a statement is done. A Cursor in SQL is an arrangement of rows together with a pointer that recognizes a present row. It is a database object to recover information from a result set one row at once. It is helpful when we need to control the record of a table in a singleton technique, at the end of the day one row at any given moment. The arrangement of columns the cursor holds is known as the dynamic set.

Types of Cursors

*There are two types of cursors:

1. Implicit Cursor

These sorts of Cursors in SQL are produced and utilized by the framework amid the control of a DML inquiry (INSERT, UPDATE and DELETE). A certain cursor is likewise created by the framework when a solitary row is chosen by a SELECT charge.

2. Explicit Cursor

This kind of cursor is produced by the user utilizing a SELECT charge. This cursor contains in excess of one row. However, just a single row can be prepared at once. An express cursor moves one by one finished the records. It uses a pointer that holds the record of a column. Subsequent to bringing a row, the cursor pointer moves to the following column.

Attributes Used in Cursor.

CURSOR ATTRIBUTE	SYNTAX	DESCRIPTION
%NOTFOUND	cursor_name%NOTFOUND	%NOTFOUND returns **TRUE** if last fetch did not return a row, Else **FALSE** if last fetch returns row.
%FOUND	cursor_name%FOUND	%FOUND returns **TRUE** if the cursor is open, fetches the row till the last fetch. **FALSE** if last fetch did not return any row.
%ROWCOUNT	cursor_name%ROWCOUNT	%ROWCOUNT keeps track of fetched rows from cursor until it is closed.
%ISOPEN	cursor_name%ISOPEN	%ISOPEN returns **TRUE** if its cursor or cursor variable is open, otherwise, %ISOPEN returns **FALSE**.

EMPLICIT Cursor(Programs):

1. Pl/SQL Program to Show the uses of implicit cursor without using any attribute:

```
SQL>   DECLARE
  2    EMPLOYEE_NAME VARCHAR2(35);
  3    EMPLOYEE_JOB   VARCHAR2(35);
  4     NEWemp_id NUMBER:=&employee_id;
  5      BEGIN
  6           SELECT EMP_NAME,JOB
  7     into     EMPLOYEE_NAME,EMPLOYEE_JOB
  8     from employee
  9     where EMP_ID= NEWemp_id;
 10      dbms_output.Put_line ('Employee name:- '||EMPLOYEE_NAME||' '||'EMPLOYEE_JOB :-'||EMPLOYEE_JOB);
 11     EXCEPTION
 12          WHEN no_data_found THEN
 13            dbms_output.Put_line ('There is no employee with the ID '||to_char(NEWemp_id));
 14       END;
 15       /
Enter value for employee_id: 6
old     4:     NEWemp_id NUMBER:=&employee_id;
new     4:     NEWemp_id NUMBER:=6;
Employee name:- KARAN SHARDUL EMPLOYEE_JOB :-MANAGER

PL/SQL procedure successfully completed.
```

2. PL/SQL Program to perform Implicit Cursor Using <u>%NOTFOUND Attribute.</u>

```
SQL> DECLARE
  2     CURSOR c1 IS SELECT EMP_NAME,salary    FROM employee where EMP_ID<6;
  3              my_ename employee.EMP_NAME%TYPE;
  4        my_salary employee.salary%TYPE;
  5     BEGIN
  6        OPEN c1;
  7        LOOP
  8          FETCH c1 INTO my_ename, my_salary;
  9          IF c1%NOTFOUND THEN -- fetch failed, so exit loop
 10     -- "EXIT WHEN c1%NOTFOUND OR c1%NOTFOUND IS NULL;"
 11              EXIT;
 12          ELSE     -- fetch succeeded
 13            DBMS_OUTPUT.PUT_LINE
 14              ('Name = ' || my_ename || ', salary = ' || my_salary);
 15          END IF;
 16        END LOOP;
 17     END;
 18     /
Name = PRATIK SULTANE, salary = 125000
Name = SPANDAN MARATHE, salary = 70000
Name = MANGESH SHIMPI, salary = 98000
Name = ROHAN JADHAV, salary = 60000


PL/SQL procedure successfully completed.
```

Explicit Cursor

Every Cursor in SQL contains the followings 4 sections:

- ✧ Declaring the cursor for initializing the memory
- ✧ Opening the cursor for allocating the memory
- ✧ Fetching the cursor for retrieving the data
- ✧ Closing the cursor to release the allocated memory

Declaring a Cursor

Cursors are declared much like a variable. A name is given, there are statements to open the cursor, retrieve the query result, and finally close the cursor. Note that, different SQL implementations support the use of cursors in a different way. But there is a general agreement on how the cursor should be written.

We must use SQL statements to fully implement cursor functionality because simply declaring a cursor is not enough to extract data from a SQL database. There are four basic steps to declare a cursor:

- **DECLARE CURSOR:** The declaration begins by giving cursor a name and assigning the query expression to be invoked when the cursor is opened.

  ***Syntax:**Cursor Cur-name is Select Statement;

- **OPEN:** The open statement executes the query expression assigned and make ready query result for subsequent FETCH.

  ***Syntax:**OPEN cursor_name;

- **FETCH:** Retrieves data values into variables which then can be passed to host programming language or to other embedded SQL statements.

  ***Syntax:**FETCH cursor_name INTO record_name;

- **CLOSE:** The cursor is closed from fetching any more query result.

  ***Syntax:**CLOSE cursor_name;

Explicit Cursors are classified into

1) **Normal cursor**

2) **Parameterized cursor**

3) **Cursor For Loops and**

4) **REF cursors**

1. **Normal cursor(Explicit Cursor):**

DISPLAYING DATA OF A TABLE (Explicit Cursor).

```
SQL> DECLARE
  2       CURSOR cur_emp_detail IS
  3          SELECT emp_id,
  4                 EMP_NAME,
  5                 salary
  6          FROM     employee;
  7       TYPE type_record_type IS RECORD (
  8          emp_id employee.emp_id%TYPE,
  9          EMP_NAME employee.emp_name%TYPE,
 10          Employee_salary employee.salary%TYPE );
 11
 12       emp_rec_type    type_record_type;
 13  BEGIN
 14       OPEN cur_emp_detail;
 15       LOOP
 16            FETCH cur_emp_detail INTO emp_rec_type;
 17            EXIT WHEN cur_emp_detail%NOTFOUND;
 18            dbms_output.Put_line('Employees Information::    '
 19                                 ||'   ID: '
 20                                 ||emp_rec_type.emp_id
 21                                 ||'|     Name: '
 22                                 ||emp_rec_type.emp_name
 23                                 ||'|     Salary: '
 24                                 ||emp_rec_type.employee_salary);
 25       END LOOP;
 26       dbms_output.Put_line('Total number of Employees : '
 27                            ||cur_emp_detail%rowcount);
 28       CLOSE cur_emp_detail;
 29  END;
 30  /
Employees Information::      ID: 1|     Name: PRATIK SULTANE|      Salary: 125000
Employees Information::      ID: 2|     Name: SPANDAN MARATHE|      Salary: 70000
Employees Information::      ID: 3|     Name: MANGESH SHIMPI|      Salary: 98000
Employees Information::      ID: 4|     Name: ROHAN JADHAV|      Salary: 60000
Employees Information::      ID: 6|     Name: KARAN SHARDUL|      Salary: 100000
Employees Information::      ID: 7|     Name: PIYUSH PAWAR|      Salary: 120000
Employees Information::      ID: 8|     Name: KUNAL RAJPUT|      Salary: 69000
Employees Information::      ID: 9|     Name: PRATIK YEOLE|      Salary: 49000
Employees Information::      ID: 10|     Name: PRAKASH BHABAD|      Salary: 97000
Employees Information::      ID: 11|     Name: RAJ KANADE|      Salary: 80000
Employees Information::      ID: 12|     Name: KUNAL AHER|      Salary: 65000
Employees Information::      ID: 13|     Name: LALIT PAWAR|      Salary: 70000
Employees Information::      ID: 14|     Name: RITESH DESHMUKH|      Salary: 58000
Employees Information::      ID: 15|     Name: KUNAL KEDARE|      Salary: 45000
Total number of Employees : 14

PL/SQL procedure successfully completed.
```

2. **Parameterized cursor**

3. **Cursor For Loops**

Program No 1 Fetch Emp Data Into Temp Table By Cursor Using For Loop.

Temp Table SQL Query:

```
 create table temp (
 col_no1    NUMBER(4),
 col_no2 varchar2(20),
 col_no3 varchar2(25),
 col_no4    NUMBER(10)
);
```

```
SQL>    DECLARE
  2          CURSOR c1 is
  3    select EMP_ID,EMP_NAME,JOB,SALARY from employee
  4     order by EMP_ID asc;
  5     my_eno number(4);
  6     my_ename VARCHAR2(20);
  7     my_JOB varchar2(25);
  8     my_sal      NUMBER(10);
  9       begin
 10       open c1;
 11            for i in 1..8 loop
 12         fetch c1 into
 13           my_eno,my_ename, my_job,my_sal;
 14     exit when c1%notfound;/*in case number is Requested*/
 15     INSERT INTO temp VALUES ( my_eno,my_ename, my_job,my_sal);
 16     commit;
 17     end loop;
 18     close c1;
 19     end;
 20     /
```

PL/SQL procedure successfully completed.

```
SQL> select * from temp;
```

COL_NO1	COL_NO2	COL_NO3	COL_NO4
1	PRATIK SULTANE	Project Leader	125000
2	SPANDAN MARATHE	SALESMAN	70000
3	MANGESH SHIMPI	Receptionist	98000
4	ROHAN JADHAV	MANAGER	60000
6	KARAN SHARDUL	MANAGER	100000
7	PIYUSH PAWAR	Executive Assistant	120000
8	KUNAL RAJPUT	Accountant	69000
9	PRATIK YEOLE	PRESIDENT	49000

8 rows selected.

4. REF Cursor

```
SQL> CREATE OR REPLACE FUNCTION get_all_data(
  2            in_EMP_ID IN employee.EMP_ID%TYPE)
  3       RETURN SYS_REFCURSOR
  4  AS
  5     EMP_DATA SYS_REFCURSOR;
  6  BEGIN
  7
  8       OPEN EMP_DATA FOR
  9       SELECT      emp_id,
 10     Emp_name,
 11  HIREDATE,JOB,Salary
 12         FROM    employee
 13  WHERE
 14       EMP_ID = in_EMP_ID
 15       ORDER BY
 16               Emp_id;
 17  RETURN EMP_DATA;
 18  END;
 19  /

Function created.

SQL> DECLARE
  2     EMP_DATA SYS_REFCURSOR;
  3     E_emp_id    employee.emp_id%TYPE;
  4     E_EMP_NAME employee.emp_name%TYPE;
  5     E_SALARY employee.salary%TYPE;
  6
  7
  8  BEGIN
  9       -- get the ref cursor from function
 10     EMP_DATA := get_all_data(10);
 11
 12       -- process each employee
 13       LOOP
 14            FETCH
 15               EMP_DATA
 16            INTO
 17                E_emp_id ,
 18              E_EMP_NAME,
 19              E_SALARY;
 20
 21            EXIT
 22       WHEN    EMP_DATA%notfound;
 23            dbms_output.put_line( E_emp_id || ' ' ||    E_EMP_NAME|| ' - ' ||      E_SALARY );
 24       END LOOP;
 25       -- close the cursor
 26       CLOSE EMP_DATA;
 27  END;
 28  /
10   PRAKASH BHABAD - 97000

PL/SQL procedure successfully completed.
```

```
SQL> select * from employee where EMP_ID='10';

    EMP_ID EMP_NAME            JOB                    HIREDATE     SALARY
---------- -------------------- ---------------------- -------- ----------
        10 PRAKASH BHABAD       Office Manager         08-09-81     102000
```

Types of Cursor in SQL Server

- ✓ **STATIC CURSOR:** A static cursor populates the outcome set amid cursor creation and the object result is reserved for the lifetime of the cursor. A static cursor can push ahead and in reverse.
- ✓ **FAST_FORWARD:** This is the default sort of cursor. It is indistinguishable from the static with the exception of that you can just look forward.
- ✓ **DYNAMIC:** In a dynamic cursor, increases and deletes are noticeable for others in the data source while the cursor is open.
- ✓ **KEYSET:** This is like a dynamic cursor aside from we can't see records others include. On the off chance that another client deletes a table, it is distant from our table set.
- ✓

❖ Disadvantages/Limitation Of The Cursor.

Cursor requires a network roundtrip each time it fetches a record, thus consume network resources. While data processing, it issues locks on part of the table, or on the whole table.

What are the disadvantages of cursors?

Disadvantages of cursors
- Uses more resources because Each time you fetch a row from the cursor, it results in a network roundtrip
- There are restrictions on the SELECT statements that can be used.
- Because of the round trips, performance and speed is slow

Importance of Cursor in PL/SQL

Pointing to the memory location and performing actions accordingly is one of the important tasks in any programming language. In PL/SQL, it is done by Cursors. Cursors play a crucial role when it comes to performing the different task by giving a name to the memory area (context area) where the result of SQL queries are saved. We can access the records one by one and perform any manipulations in it if required or display it on the console accordingly. Explicit Cursors are more efficient, give more programmatic control and less vulnerable to data errors so they are very useful in PL/SQL programming than Implicit ones.

Conclusion – SQL Cursor

Hence, in this SQL Cursor Micro-Project, we discussed Cursor in SQL. Moreover, we learned parts, terms, and use of SQL Cursor. Also, we discussed types of Cursors in SQL. Along with this, we saw SQL Cursors example With It's Type Implicit and Explicit..