**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.

1. **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.

|  |  |  |
| --- | --- | --- |
| **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**. |

### *****Attributes Used in Cursor.*****

### ****EMPLICIT Cursor(Programs):****

# **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.

1. **PL/SQL Program to perform Implicit Cursor Using %NOTFOUND Attribute.**

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.

1. **Parameterized cursor**
2. **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.

1. **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

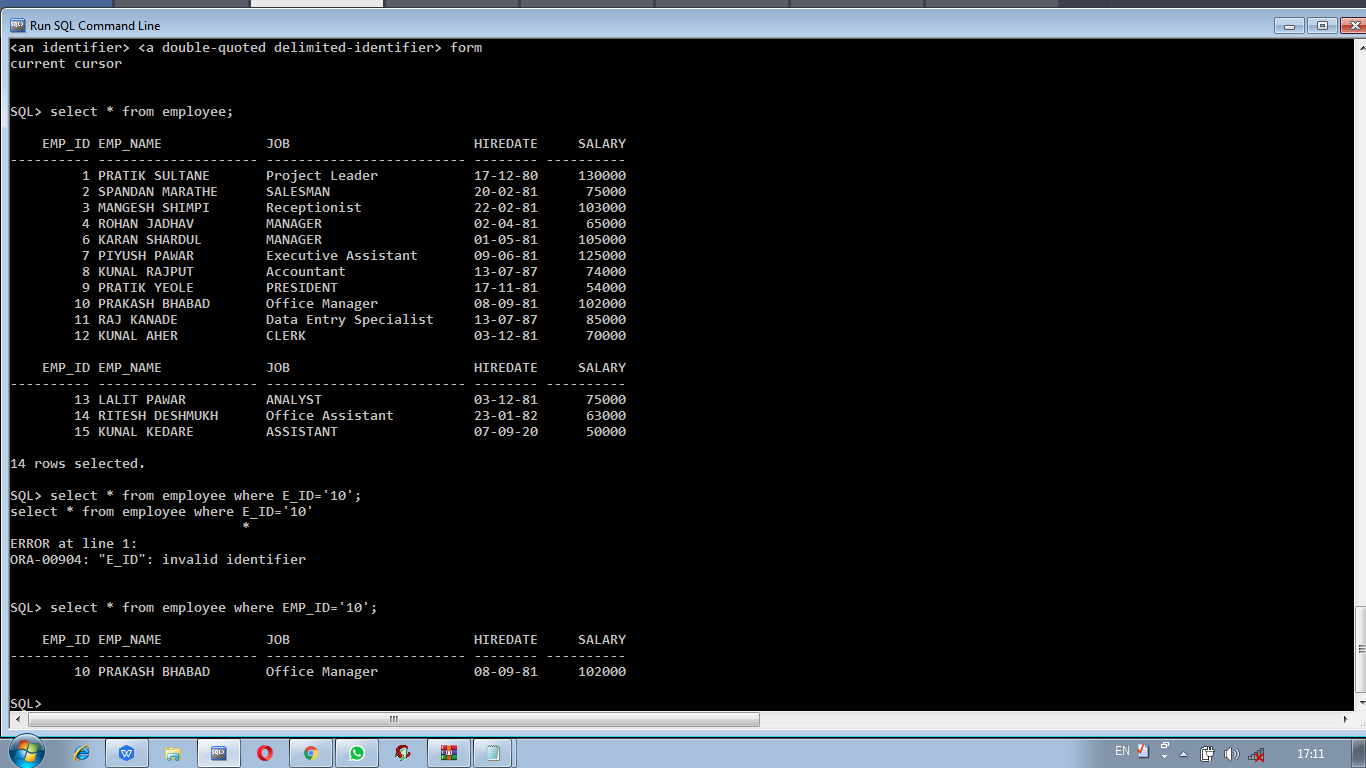
26 CLOSE EMP\_DATA;

27 END;

28 /

10 PRAKASH BHABAD - 97000

PL/SQL procedure successfully completed.



**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..