# Oracle PL/SQL Cursor: Implicit, Explicit, Cursor FOR Loop [Example]

## What is CURSOR in PL/SQL?

A Cursor is a pointer to this context area. Oracle creates context area for processing an SQL statement which contains all information about the statement.

PL/SQL allows the programmer to control the context area through the cursor. A cursor holds the rows returned by the SQL statement. The set of rows the cursor holds is referred as active set. These cursors can also be named so that they can be referred from another place of the code.

In this tutorial you will learn-

* [Implicit Cursor](https://www.guru99.com/pl-sql-cursor.html" \l "1)
* [Explicit Cursor](https://www.guru99.com/pl-sql-cursor.html" \l "2)
* [Cursor Attributes](https://www.guru99.com/pl-sql-cursor.html" \l "3)
* [FOR Loop Cursor statement](https://www.guru99.com/pl-sql-cursor.html" \l "4)

The cursor is of two types.

* Implicit Cursor
* Explicit Cursor

## Implicit Cursor

Whenever any DML operations occur in the database, an implicit cursor is created that holds the rows affected, in that particular operation. These cursors cannot be named and, hence they cannot be controlled or referred from another place of the code. We can refer only to the most recent cursor through the cursor attributes.

## Explicit Cursor

Programmers are allowed to create named context area to execute their DML operations to get more control over it. The explicit cursor should be defined in the declaration section of the PL/SQL block, and it is created for the 'SELECT' statement that needs to be used in the code.

Below are steps that involved in working with explicit cursors.

* Declaring the cursor

Declaring the cursor simply means to create one named context area for the 'SELECT' statement that is defined in the declaration part. The name of this context area is same as the cursor name.

* Opening Cursor

Opening the cursor will instruct the PL/SQL to allocate the memory for this cursor. It will make the cursor ready to fetch the records.

* Fetching Data from the Cursor

In this process, the 'SELECT' statement is executed and the rows fetched is stored in the allocated memory. These are now called as active sets. Fetching data from the cursor is a record-level activity that means we can access the data in a record-by-record way.

Each fetch statement will fetch one active set and holds the information of that particular record. This statement is same as 'SELECT' statement that fetches the record and assigns to the variable in the 'INTO' clause, but it will not throw any exceptions.

* Closing the Cursor

Once all the record is fetched now, we need to close the cursor so that the memory allocated to this context area will be released.

Syntax:

DECLARE

CURSOR <cursor\_name> IS <SELECT statement^>

<cursor\_variable declaration>

BEGIN

OPEN <cursor\_name>;

FETCH <cursor\_name> INTO <cursor\_variable>;

.

.

CLOSE <cursor\_name>;

END;

* In the above syntax, the declaration part contains the declaration of the cursor and the cursor variable in which the fetched data will be assigned.
* The cursor is created for the 'SELECT' statement that is given in the cursor declaration.
* In execution part, the declared cursor is opened, fetched and closed.

## Cursor Attributes

Both Implicit cursor and the explicit cursor has certain attributes that can be accessed. These attributes give more information about the cursor operations. Below are the different cursor attributes and their usage.

|  |  |
| --- | --- |
| Cursor Attribute | Description |
| %FOUND | It returns the Boolean result 'TRUE' if the most recent fetch operation fetched a record successfully, else it will return FALSE. |
| %NOTFOUND | This works oppositely to %FOUND it will return 'TRUE' if the most recent fetch operation could not able to fetch any record. |
| %ISOPEN | It returns Boolean result 'TRUE' if the given cursor is already opened, else it returns 'FALSE' |
| %ROWCOUNT | It returns the numerical value. It gives the actual count of records that got affected by the DML activity. |

Example 1: In this example, we are going to see how to declare, open, fetch and close the explicit cursor.

We will project all the employee's name from emp table using a cursor. We will also use cursor attribute to set the loop to fetch all the record from the cursor.

DECLARE

CURSOR guru99\_det IS SELECT emp\_name FROM emp;

lv\_emp\_name emp.emp\_name%type;

BEGIN

OPEN guru99\_det;

LOOP

FETCH guru99\_det INTO lv\_emp\_name;

IF guru99\_det%NOTFOUND

THEN

EXIT;

END IF;

Dbms\_output.put\_line(‘Employee Fetched:‘||lv\_emp\_name);

END LOOP;

Dbms\_output.put\_line(‘Total rows fetched is‘||guru99\_det%R0WCOUNT);

CLOSE guru99\_det;

END:

/

Output

Employee Fetched:BBB

Employee Fetched:XXX

Employee Fetched:YYY

Total rows fetched is 3

Code Explanation:

* Code line 2: Declaring the cursor guru99\_det for statement 'SELECT emp\_name FROM emp'.
* Code line 3: Declaring variable lv\_emp\_name.
* Code line 5: Opening the cursor guru99\_det.
* Code line 6: Setting the Basic loop statement to fetch all the records in the 'emp' table.
* Code line 7: Fetches the guru99\_det data and assign the value to lv\_emp\_name.
* Code line 9: Using the cursor attribute '%NOTFOUND' to find whether all the record in the cursor is fetched. If fetched then it will return 'TRUE' and control will exit from the loop, else the control will keep on fetching the data from the cursor and print the data.
* Code line 11: EXIT condition for the loop statement.
* Code line 12: Print the fetched employee name.
* Code line 14: Using the cursor attribute '%ROWCOUNT' to find the total number of records that got affected/fetched in the cursor.
* Code line 15: After exiting from the loop the cursor is closed and the memory allocated is set free.

## FOR Loop Cursor statement

"FOR LOOP" statement can be used for working with cursors. We can give the cursor name instead of range limit in the FOR loop statement so that the loop will work from the first record of the cursor to the last record of the cursor. The cursor variable, opening of cursor, fetching and closing of the cursor will be done implicitly by the FOR loop.

Syntax:

DECLARE

CURSOR <cursor\_name> IS <SELECT statement>;

BEGIN

FOR I IN <cursor\_name>

LOOP

.

.

END LOOP;

END;

* In the above syntax, the declaration part contains the declaration of the cursor.
* The cursor is created for the 'SELECT' statement that is given in the cursor declaration.
* In execution part, the declared cursor is setup in the FOR loop and the loop variable 'I' will behave as cursor variable in this case.

Example 1: In this example, we will project all the employee name from emp table using a cursor-FOR loop.

DECLARE

CURSOR guru99\_det IS SELECT emp\_name FROM emp;

BEGIN

FOR lv\_emp\_name IN guru99\_det

LOOP

Dbms\_output.put\_line(‘Employee Fetched:‘||lv\_emp\_name.emp\_name);

END LOOP;

END;

/

Output

Employee Fetched:BBB

Employee Fetched:XXX

Employee Fetched:YYY

Code Explanation:

* Code line 2: Declaring the cursor guru99\_det for statement 'SELECT emp\_name FROM emp'.
* Code line 4: Constructing the 'FOR' loop for the cursor with the loop variable lv\_emp\_name.
* Code line 5: Printing the employee name in each iteration of the loop.
* Code line 8: Exit the loop

Note: In Cursor-FOR loop, cursor attributes cannot be used since opening, fetching and closing of the cursor is done implicitly by FOR loop.

## What WHERE CURRENT OF clause does in a cursor?

PL/SQL provides the WHERE CURRENT OF clause for both UPDATE and DELETE statements inside a cursor.This allows you to easily make changes to the most recently fetched row of data.   
  
**Syntax:**

UPDATE table\_name  
     SET set\_clause  
WHERE CURRENT OF cursor\_name;

Notice that the WHERE CURRENT OF clause references the cursor and not the record into which the next fetched row is deposited.

## Can you pass a parameter to a cursor?

- PL/SQL also allows you to pass parameters into cursors. It eases your work because:  
1. A parameter makes the cursor more reusable.   
2. A parameter avoids scoping problems.  
  
- However, you should pass parameters when you are going to use it at more then one place and when there are going to be different values for the same WHERE statement.   
  
**Example:**

Parameterized cursor:  
  
/\*Create a table\*/  
create table Employee  
(  
ID VARCHAR2(4 BYTE)NOT NULL,  
First\_Name VARCHAR2(10 BYTE)  
);  
  
/\*Insert some data\*/  
Insert into Employee (ID, First\_Name) values ('01','Harry');  
  
/\*create cursor\*/  
declare  
cursor c\_emp(cin\_No NUMBER)is select count(\*) from employee where id=cin\_No;  
v\_deptNo employee.id%type:=10;  
v\_countEmp NUMBER;  
begin  
open c\_emp (v\_deptNo);  
fetch c\_emp into v\_countEmp;  
close c\_emp;  
end;  
  
/\*Using cursor\*/  
Open c\_emp (10);