

## Cursors

Oracle creates a memory area, known as the context area, for processing an SQL statement, which contains all the information needed for processing the statement.

PL/SQL controls the context area through a cursor. A cursor holds the rows (one or more) returned by a SQL

There are two types of cursors

- Implicit cursors
- Explicit cursors

### Implicit Cursors

Implicit cursors are automatically created by Oracle whenever an SQL statement is executed, when there is no explicit cursor for the statement.

Whenever a DML statement (INSERT, UPDATE and DELETE) is issued, an implicit cursor is associated with this statement. For INSERT operations, the cursor holds the data that needs to be inserted. For UPDATE and DELETE operations, the cursor identifies the rows that would be affected.

In PL/SQL, you can refer to the most recent implicit cursor as the **SQL cursor**, which always has attributes such as **%FOUND**, **%ISOPEN**, **%NOTFOUND**, and **%ROWCOUNT**. The SQL cursor has additional attributes

Select \* from customers;

ID	NAME	AGE	ADDRESS	SALARY
1	Raju	32	Karimnagar	20000
2	Ravi	35	Hyderabad	35000
3	Manoj	25	Delhi	12000
4	Manish	36	Warangal	22000
5	Rahul	21	MP	33500
6	Ramu	20	Bhopal	22500

```

DECLARE
    total_rows number(2);
BEGIN
    UPDATE customers
    SET salary = salary + 500;
    IF sql%notfound THEN
        dbms_output.put_line('no customers selected');
    ELSIF sql%found THEN
        total_rows := sql%rowcount;
        dbms_output.put_line( total_rows || ' customers selected ');
    END IF;
END;

```

### Explicit Cursors

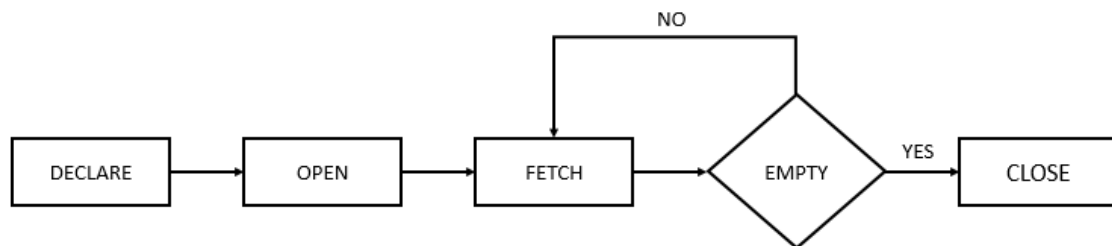
Explicit cursors are programmer-defined cursors for gaining more control over the context area. An explicit cursor should be defined in the declaration section of the PL/SQL Block. It is created on a SELECT Statement which returns more than one row.

The syntax for creating an explicit cursor is –

```
CURSOR cursor_name IS select_statement;
```

Working with an explicit cursor includes the following steps –

- 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



```
DECLARE
CURSOR <cursor_name> IS <SELECT statement^>
<cursor_variable declaration>
BEGIN
OPEN <cursor_name>;
FETCH <cursor_name> INTO <cursor_variable>;
.
.
CLOSE <cursor_name>;
END;
```

### **Declaring the Cursor**

Declaring the cursor defines the cursor with a name and the associated SELECT statement. For example –

```
CURSOR c_customers IS
    SELECT id, name, address FROM customers;
```

### **Opening the Cursor**

Opening the cursor allocates the memory for the cursor and makes it ready for fetching the rows returned by the SQL statement into it. For example, we will open the above defined cursor as follows –

```
OPEN c_customers;
```

### **Fetching the Cursor**

Fetching the cursor involves accessing one row at a time. For example, we will fetch rows from the above-opened cursor as follows –

```
FETCH c_customers INTO c_id, c_name, c_addr;
```

### **Closing the Cursor**

Closing the cursor means releasing the allocated memory. For example, we will close the above-opened cursor as follows –

```
CLOSE c_customers;
```

## Example

Following is a complete example to illustrate the concepts of explicit cursors &minua;

```
DECLARE
  c_id customers.id%type;
  c_name customerS.No.ame%type;
  c_addr customers.address%type;
  CURSOR c_customers is
    SELECT id, name, address FROM customers;
BEGIN
  OPEN c_customers;
  LOOP
  FETCH c_customers into c_id, c_name, c_addr;
    EXIT WHEN c_customers%notfound;
    dbms_output.put_line(c_id || ' ' || c_name || ' ' || c_addr);
  END LOOP;
  CLOSE c_customers;
END;
/
```

ID	NAME	ADDRESS
1	Raju	Karimnagar
2	Ravi	Hyderabad
3	Manoj	Delhi
4	Manish	Warangal
5	Rahul	MP
6	Ramu	Bhopal