**WEEK-12**

**Definition:-**

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. A **cursor**is a pointer to this context area.

There are two types of cursors −

1. Implicit cursors
2. 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. Programmers cannot control the implicit cursors and the information in it. If you use an implicit cursor, Oracle will perform the open, fetches, and close for you automatically. Implicit cursors are used in statements that return only one row.

In PL/SQL, you can refer to the most recent implicit cursor as the SQL cursor, which always having some attributes. The following table provides the description of the most used attributes.

|  |  |
| --- | --- |
| **Attribute** | **Description** |
|  |  |
| %ROWCOUNT | The number of rows processed by a SQL statement. |
| %FOUND | TRUE if at least one row was processed. |
| %NOTFOUND | TRUE if no rows were processed. |
| %ISOPEN | TRUE if cursor is open or FALSE if cursor has not been opened or has been closed. Only used with explicit cursors. Always returns FALSE for implicit cursors, because Oracle closes the SQL cursor automatically after executing its associated SQL statement. |

**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:-

1. Declaring the cursor for initializing the memory
2. Opening the cursor for allocating the memory
3. Fetching the cursor for retrieving the data
4. Closing the cursor to release the allocated memory

***Declaring the Cursor***

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

**Syntax:**

CURSOR cursor\_name IS 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.

**Syntax:**

OPEN cursor\_name;

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.

**Syntax:**

FETCH cursor\_name INTO cursor\_variable(s);

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.

**Syntax:**

CLOSE cursor\_name;

For example, we will close the above-opened cursor as follows:

CLOSE c\_customers;

**1)** **To write an Implicit Cursor for updating the salary of each customer by 500 and determine the number of rows affected.**

**STEP 1: Creation of customers table**

SQL> create table customers(id number,name varchar2(10),age number,location varchar2(15),salary number);

Table created.

**STEP 2: Inserting values in customers table**

SQL> insert into customers values (1,'Ramesh',32,'Hyderabad',15000);

1 row created.

SQL> insert into customers values (2,'Rahul',29,'Mumbai',13000);

1 row created.

SQL> insert into customers values (3,'Ranjith',23,'Chennai',9000);

1 row created.

SQL> insert into customers values (4,'Ram cahran',25,'Banglore',17000);

1 row created.

SQL> insert into customers values (5,'Rajesh',28,'Mysore',16000);

1 row created.

**STEP 3: Display the records of customer details**



**STEP 4:**

**The following program will update the table and increase the salary of each customer by 500 and use the SQL%ROWCOUNT attribute to determine the number of rows affected.**

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;

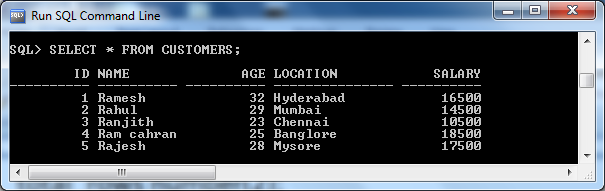
END;

/

5 customers selected

PL/SQL procedure successfully completed.

**STEP 5: Display the records of customer details**

****

**2)** **To write an Implicit Cursor for displaying the required customer details.**

DECLARE

var\_id number;

var\_name varchar2(10);

var\_age number;

var\_location varchar2(10);

var\_salary number;

req\_id number:=&req\_id;

BEGIN

SELECT id,name,age,location,salary

INTO var\_id, var\_name, var\_age, var\_location, var\_salary

FROM customers

WHERE id=req\_id;

dbms\_output.put\_line ('Customer details regarding to required id '|| chr(10)||var\_id||chr(9)||var\_name|| chr(9)||var\_age|| chr(9)|| var\_location|| chr(9)|| var\_salary);

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

dbms\_output.put\_line ('No Customer details regarding to required id ');

END;

/

Enter value for req\_id: 5

old 7: req\_id number:=&req\_id;

new 7: req\_id number:=5;

Customer details regarding to required id

5 Rajesh 28 Mysore 16000

PL/SQL procedure successfully completed.

SQL> /

Enter value for req\_id: 6

old 7: req\_id number:=&req\_id;

new 7: req\_id number:=6;

No Customer details regarding to required id

PL/SQL procedure successfully completed.

**3)** **To write an Explicit Cursor for displaying the customer details whose id is less than 5.**

DECLARE

c\_id customers.id%type;

c\_name customers.name%type;

c\_location customers.location%type;

--declaring a cursor

CURSOR c\_customers is

SELECT id, name, location FROM customers WHERE id<5;

BEGIN

--opening a cursor

OPEN c\_customers;

LOOP

--fetching records from a cursor

FETCH c\_customers into c\_id, c\_name, c\_location;

--testing exit conditions

EXIT WHEN c\_customers%notfound;

dbms\_output.put\_line(c\_id || chr(9) || c\_name || chr(9) || c\_location);

END LOOP;

--closing the cursor

CLOSE c\_customers;

END;

/

1 Ramesh Hyderabad

2 Rahul Mumbai

3 Ranjith Chennai

4 Ram cahran Banglore

PL/SQL procedure successfully completed.PL/SQL procedure successfully completed.