NAME: Atharva Chavan
TE-A IT
ROLL NO: T1851010
PRN: 71901316L

Group B: SQL & PL/SQL

Assignment No. 10

Aim: Write a PL/SQL block to implement all types of cursor.

Objective:

- To study and implement PL/SQLcursors.

Theory :

A **cursor** is a pointer to this context area. PL/SQL controls the context area through a cursor. A cursor holds the rows (one or more) returned by a SQL statement. The set of rows the cursor holds is referred to as the **active set**.

You can name a cursor so that it could be referred to in a program to fetch and process the rows returned by the SQL statement, one at a time. There are two types of cursors −

- Implicitcursors
- Explicitcursors

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 init.

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 beaffected.

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, **%BULK_ROWCOUNT** and **%BULK_EXCEPTIONS**, designed for use with the **FORALL** statement. The following table provides the description of the most used attributes–

S.No	Attribute & Description
1	**%FOUND** Returns TRUE if an INSERT, UPDATE, or DELETE statement affected one or more rows or a SELECT INTO statement returned one or more rows. Otherwise, it returns FALSE.
2	**%NOTFOUND** The logical opposite of %FOUND. It returns TRUE if an INSERT, UPDATE, or DELETE statement affected no rows, or a SELECT INTO statement returned no rows. Otherwise, it returns FALSE.
3	**%ISOPEN** Always returns FALSE for implicit cursors, because Oracle closes the SQL cursor automatically after executing its associated SQL statement.
4	**%ROWCOUNT** Returns the number of rows affected by an INSERT, UPDATE, or DELETE statement, or returned by a SELECT INTO 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 −

- Declaring the cursor for initializing thememory
- Opening the cursor for allocating thememory
- Fetching the cursor for retrieving thedata
- Closing the cursor to release the allocatedmemory

- DeclaringtheCursor

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 theCursor

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 theCursor

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;
```

Conclusion:-

We have studied and implemented cursors in PL/SQL.

Code & Output: -

1) **FirstExample**

```
mysql>createprocedureCONNAMSAL()
    ->begin
    ->declareenamevarchar(100);
    ->declareesalaryinteger(10);
    ->declarev_finishedintegerdefaultO;
    ->declarec1cursorforselectNAME,SALARYfromEMPLOYEE;
    ->declarecontinuehandlerforNOTFOUNDsetv_finished=1;
    ->openc1;
    ->get_emp:LOOP
    ->fetchc1intoename,esalary;ifv_finished=1then
    ->leaveget_emp;endif;
    ->selectconcat(ename,esalary);endloopget_emp;
    ->closec1;
    -> end $
QueryOK,0rowsaffected(0.05sec)

mysql>callCONNAMSAL()$
+--------------------+
|concat(ename,esalary)|
+--------------------+
|SHREYAS10000        |
+--------------------+
1rowinset(0.00sec)

+--------------------+
|concat(ename,esalary)|
+--------------------+
|PIYUSH20000         |
+--------------------+
1rowinset(0.00sec)
       -----
QueryOK,0rowsaffected(0.00sec)
```

2) **SecondExample**

```
       -----
```

```
mysql>SELECT*FROMEMPLOYEE$
+------+--------+--------+--------+------+
|ID    |MARKS1|MARKS2|MARKS3|GRADE|
+------+--------+--------+--------+------+
|    1|      50|      50|      50|NULL|
|    2|      60|      62|      63|NULL|
|    1|      65|      70|      75|NULL|
|    3|      90|      90|      95|NULL|
|    4|      80|      80|      85|NULL|
+------+--------+--------+--------+------+
5rowsinset(0.00sec)

mysql>createprocedureregrade2()
    ->          begin
    ->          declare
    ->          eid,m1,m2,m3int;
    ->          declareavgfloat;
    ->          declarefinishedintdefault0;
    ->          declarec1cursorforselectID,MARKS1,MARKS2,MARKS3fromEMPLOYEE;
    ->          declarecontinuehandlerfornotfoundsetfinished=1;
    ->          openc1;
    ->          l4:loopfetchc1intoeid,m1,m2,m3;
    ->          if(finished=1)
    ->          then
    ->leavel4;
```

```
    ->end if;
    ->setavgl=(m1+m2+m3)/3;
    ->if(avgl>=80)
    ->then
    ->updateEMPLOYEEsetgrade="a"whereeid=id;
    ->elseif(avgl>=60andavgl<80)
    ->then
    ->updateEMPLOYEEsetgrade="b"whereeid=id;
    ->elseif(avgl>=40andavgl<60)
    ->then
    ->updateEMPLOYEEsetgrade="c"whereeid=id;
    ->else
    ->          updateEMPLOYEEsetgrade="f"whereeid=id;
    ->endif;
    ->endif;
    ->endif;
    ->endloop;
    ->closec1;
    ->end;
    -> $
QueryOK,0rowsaffected(0.00sec)

mysql>callgrade2()$
QueryOK,0rowsaffected(0.20sec)

mysql>select*fromEMPLOYEE$
+-----+------+-------+-------+------+
|ID   |MARKS1|MARKS2|MARKS3|GRADE|
+-----+------+-------+-------+------+
|    1|      50|      50|      50|b      |
|    2|      60|      62|      63|b      |
|    1|      65|      70|      75|b      |
|    3|      90|      90|      95|a      |
|    4|      80|      80|      85|a      |
+-----+------+-------+-------+------+
5rowsinset(0.00sec)
```

3) ThirdExample

```
mysql>createprocedurestat()
    ->begin
    ->declareid,salint;
    ->declarefinishedintdefault0;
    ->declarec2cursorforselectcid,salaryfromcustomer;
    ->declarecontinuehandlerfornotfoundsetfinished=1;
    ->openc2;
    ->l1:loopfetchc2intoid,sal;
    ->if(finished=1)
    ->thenleavel1;
    -> end if;
    ->if(sal>=60000)
    ->thenupdatecustomersetstatus="platinum"wherecid=id;
    ->elseif(sal>=40000andsal<60000)
    ->thenupdatecustomersetstatus="gold"wherecid=id;
    ->elseif(sal<40000andsal>=10000)
    ->thenupdatecustomersetstatus="silver"wherecid=id;
    -> endif;
    -> endif;
    -> endif;
    ->endloop;
    ->closec2;
    ->end;
    -> $$
```

```
Query OK, 0 rows affected (0.17 sec)

mysql> select * from customer;
    -> $$
+-----+-------------+--------+--------+
| cid | cname       | salary | status |
+-----+-------------+--------+--------+
|   1 | Atharva     |  98000 | NULL   |
|   2 | mahi        |  10000 | NULL   |
|   3 | bhagyashree |  75000 | NULL   |
|   4 | mitesh      |  67000 | NULL   |
|   5 | sarika      |  15000 | NULL   |
|   6 | sushila     |  45000 | NULL   |
|   7 | reshma      |  65000 | NULL   |
|   8 | pratibha    |  99000 | NULL   |
+-----+-------------+--------+--------+
8 rows inset (0.00 sec)

mysql> call stat$$
Query OK, 0 rows affected, 1 warning (0.24 sec)

mysql> select * from customer;
    -> $$
+-----+-------------+--------+----------+
| cid |  cname      | salary | status   |
+-----+-------------+--------+----------+
|   1 | Atharva     |  98000 | platinum |
|   2 | mahi        |  10000 | silver   |
|   3 | bhagyashree |  75000 | platinum |
|   4 | mitesh      |  67000 | platinum |
|   5 | sarika      |  15000 | silver   |
|   6 | sushila     |  45000 | gold     |
|   7 | reshma      |  65000 | platinum |
|   8 | pratibha    |  99000 | platinum |
+-----+-------------+--------+----------+
8 rows inset (0.00 sec
```