Roll No. :- COMPTEB1449

Name :- Paarth Kothari

PRN :- 72018337F

Assignment No. 6

Title :- Write PL/SQL block to implement all types of cursors (All types: Implicit, Explicit, Cursor FOR Loop, Parameterized Cursor).

Problem Statement :- Write a PL/SQL block of code using parameterized cursor that will merge the data available in newly created table N_RollCall with the data available in the O_RollCall. If the data in the first table already exists in the second table then that data should be skipped.

Objectives :-

1. To learn and understand PL/SQL in Oracle.
2. To learn and understand cursors.

Theory :-

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; for example, the number of rows processed, etc.

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 −

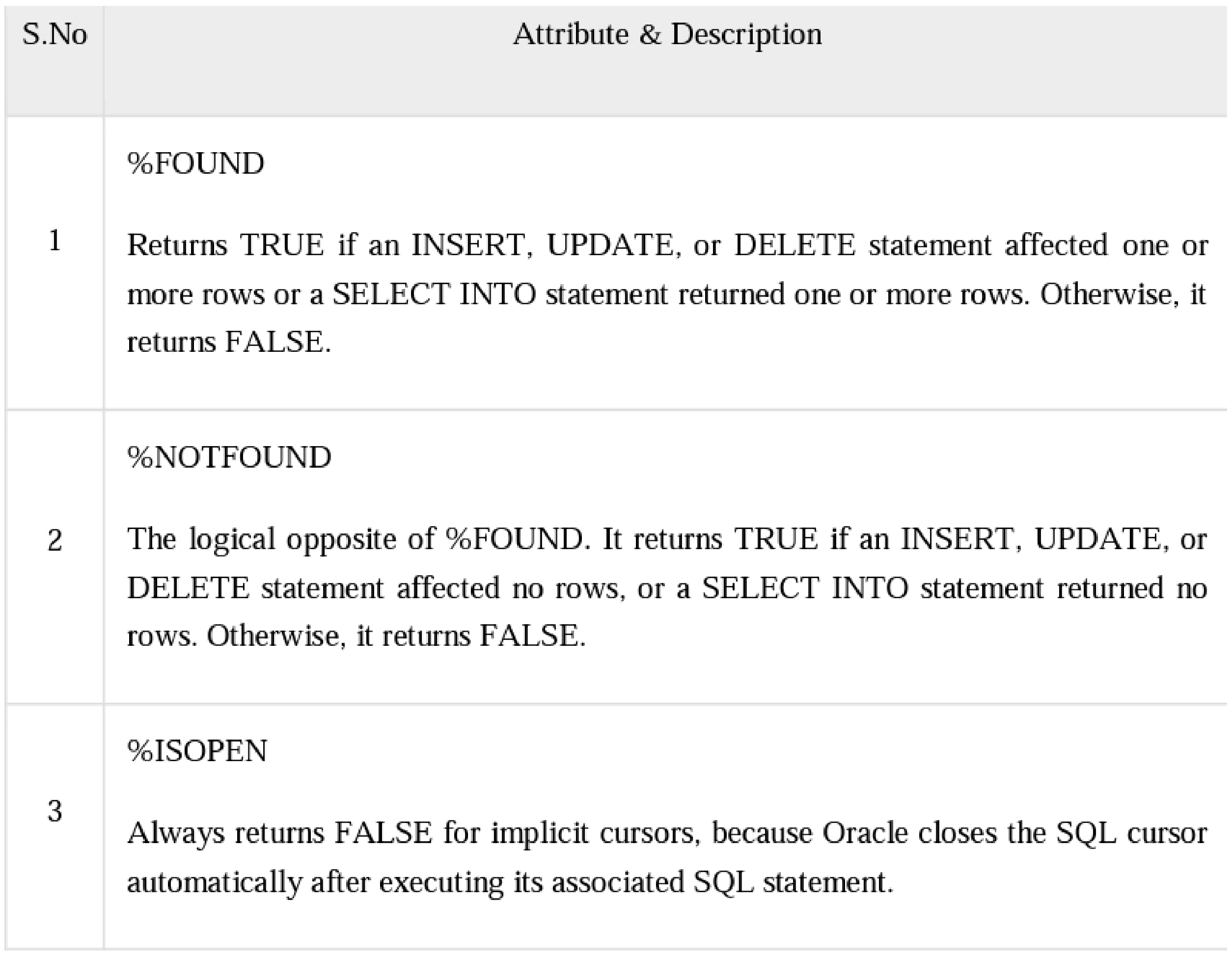
1. Implicit cursors
2. Explicit cursors

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.

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, %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.

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.

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;

Outcome:- After completing this assignment, students will be able to learn about different types of cursors (All types: Implicit, Explicit, Cursor FOR Loop, Parameterized Cursor).

Code and Output :-

```
mysql > use Abhi;
Reading table information for completion of table
and column names You can turn off this feature to get a quicker startup
with - A Database changed mysql > create table o_rollcall(
  roll_no int,
  name varchar(20),
  address varchar(20)
);
Query OK,
0 rows affected (0.28 sec) mysql > create varchar(20)
);
table n_rollcall(
  roll_no Query OK,
  0 rows affected (0.27 sec) int,
  namevarchar(20),
  address mysql > insert into o_rollcall
  values
    ('1', 'Hitesh', 'Nandura');
Query OK,
1 row affected (0.05 sec) mysql > insert into o_rollcall
values
  ('2', 'Piyush', 'MP');
Query OK,
1 row affected (0.06 sec) mysql > insert into o_rollcall
values
  ('3', 'Ashley', 'Nsk');
Query OK,
1 row affected (0.05 sec) mysql > insert into o_rollcall
values
```

```
  ('4', 'Kalpesh', 'Dhule');
Query OK,
1 row affected (0.05 sec) mysql > insert into o_rollcall
values
  ('5', 'Abhi', 'Satara');
Query OK,
1 row affected (0.04 sec) mysql > delimiter // mysql > create procedure
p3(in r1 int)-> begin -> declare r2 int;
-> declare exit_loop boolean;
-> declare c1 cursor for
select
  roll_no
from
  o_rollcall
where
  roll_no > r1;
-> declare continue handler for not found
set
  exit_loop = true;
-> open c1;
-> e_loop : loop -> fetch c1 into r2;
-> if not exists(
  select
    *
  from
    n_rollcall
  where
    roll_no = r2
)-> then -> insert into n_rollcall
select
  *
from
  o_rollcall
where
  roll_no = r2;
-> end if;
-> if exit_loop -> then -> close c1;
-> leave e_loop;
-> end if;
-> end loop e_loop;
-> end -> // Query OK,
0 rows affected (0.00 sec) mysql > call p3(3);
-> // Query OK,
0 rows affected (0.10 sec) mysql >

select * from n_rollcall;
```

```
mysql> select * from n_rollcall;
-> //



+---------+---------+---------+
| roll_no | name
| address |
+---------+---------+---------+
| 4 | Kalpesh | Dhule |
| 5 | Abhi | Satara |
| 1 | Hitesh | Nandura |
| 2 | Piyush | MP |
| 3 | Ashley | Nsk |+---------+---------+---------+
5 rows in set (0.00 sec)

mysql> insert into o_rollcall values('6','Patil','Kolhapur');
-> //

Query OK, 1 row affected (0.04 sec)
```

```
mysql> call p3(4);
-> //

Query OK, 0 rows affected (0.05 sec)

mysql> select * from n_rollcall;
-> //
+---------+---------+----------+
| roll_no | name
| address
|
```

```
+---------+---------+----------+
| 4 | Kalpesh | Dhule |
| 5 | Abhi | Satara |
| 1 | Hitesh | Nandura |
| 2 | Piyush | MP |
| 3 | Ashley | Nsk |
| 6 | Patil | Kolhapur |
+---------+---------+----------+
6 rows in set (0.00 sec)
mysql>
```

Conclusion :- Thus we have successfully implemented PL/SQL block to retrieve fine for issued library book by reading borrower information from the database.