

Code

```
mysql> drop database if exists A6;  
Query OK, 2 rows affected (0.03 sec)
```

```
mysql> create database A6;  
Query OK, 1 row affected (0.00 sec)
```

```
mysql> use A6;  
Database changed
```

```
mysql>  
mysql> CREATE TABLE O_rollCall(  
-> roll_no INT NOT NULL,  
-> name VARCHAR(20)  
-> );  
;Query OK, 0 rows affected (0.03 sec)
```

```
mysql>  
mysql> CREATE TABLE N_rollCall(  
-> roll_no INT NOT NULL,  
-> name VARCHAR(20)  
-> );  
Query OK, 0 rows affected (0.05 sec)
```

```
mysql> INSERT INTO O_rollCall VALUES(1, 'Prathamesh');  
Query OK, 1 row affected (0.00 sec)
```

```
mysql> INSERT INTO O_rollCall VALUES(2, 'Aditya');  
Query OK, 1 row affected (0.00 sec)
```

```
mysql> INSERT INTO O_rollCall VALUES(3, 'Ram');  
Query OK, 1 row affected (0.00 sec)
```

```
mysql> INSERT INTO O_rollCall VALUES(4, 'Sooraj');  
Query OK, 1 row affected (0.01 sec)
```

```
mysql> INSERT INTO O_rollCall VALUES(5, 'Shreya');  
Query OK, 1 row affected (0.00 sec)
```

```
mysql>  
mysql> INSERT INTO N_rollCall VALUES(1, 'Prathamesh');  
Query OK, 1 row affected (0.01 sec)
```

```
mysql> INSERT INTO N_rollCall VALUES(3, 'Ram');
```

Query OK, 1 row affected (0.00 sec)

```
mysql> INSERT INTO N_rollCall VALUES(5, 'Shreya');
```

Query OK, 1 row affected (0.00 sec)

```
mysql> INSERT INTO N_rollCall VALUES(6, 'Swatej');
```

```
INSERT INTO N_rollCall VALUES(7, 'Yuvraj');Query OK, 1 row affected (0.01 sec)
```

```
mysql> INSERT INTO N_rollCall VALUES(7, 'Yuvraj');
```

Query OK, 1 row affected (0.01 sec)

```
mysql> SELECT * FROM O_rollCall;
```

roll_no	name
1	Prathamesh
2	Aditya
3	Ram
4	Sooraj
5	Shreya

5 rows in set (0.00 sec)

```
mysql> SELECT * FROM N_rollCall;
```

roll_no	name
1	Prathamesh
3	Ram
5	Shreya
6	Swatej
7	Yuvraj

5 rows in set (0.00 sec)

```
mysql> delimiter $$
```

```
mysql> CREATE PROCEDURE MergeTables()
```

```
-> BEGIN
```

```
-> DECLARE roll INT;
```

```
-> DECLARE finished INT DEFAULT 0;
```

```
-> DECLARE cur1 cursor for select roll_no from O_rollCall;
```

```
-> DECLARE CONTINUE handler for NOT found set finished = 1;
```

```
-> OPEN cur1;
```

```
->
```

```
-> loop1: loop
```

```
-> IF finished = 1 THEN
```

```
-> close cur1;
```

```
-> leave loop1;
```

```
-> END IF;
```

```

-> FETCH cur1 INTO roll;
-> IF NOT EXISTS(select roll_no from N_rollCall where roll_no = roll) THEN
-> INSERT INTO N_rollCall select * from O_rollCall where roll_no = roll;
-> END IF;
-> END loop loop1;
-> END
-> $$

```

Query OK, 0 rows affected (0.02 sec)

```

mysql> delimiter ;
mysql> call MergeTables();
Query OK, 0 rows affected (0.01 sec)

```

```

mysql>
mysql>
mysql> SELECT * FROM O_rollCall;

```

```

+-----+-----+
| roll_no | name    |
+-----+-----+
|      1 | Prathamesh |
|      2 | Aditya    |
|      3 | Ram       |
|      4 | Sooraj    |
|      5 | Shreya    |
+-----+-----+

```

5 rows in set (0.00 sec)

```

mysql> SELECT * FROM N_rollCall;

```

```

+-----+-----+
| roll_no | name    |
+-----+-----+
|      1 | Prathamesh |
|      3 | Ram       |
|      5 | Shreya    |
|      6 | Swatej    |
|      7 | Yuvraj    |
|      2 | Aditya    |
|      4 | Sooraj    |
+-----+-----+

```

7 rows in set (0.00 sec)

```

mysql>

```

Assignment - 6

Title - Cursors

Date of Completion: 14/09/2020

Date of Submission: 30/09/2020

Problem Statement:

Write a PL/SQL block of code using parameterized cursor, that will merge the data available in newly created table N-EmpID with data available in the O-EmpID. Duplicate data must be skipped.

Objective:

To understand and implement types of cursors with PL/SQL block of code.

Outcomes:

1) Implement types of cursors.

Theory:

ORACLE:

PL/SQL:

* Cursors:

A cursor is a temporary work area created in system memory when SQL statement is executed.

A cursor can hold more than one row but can process only one row at a time.

Types of Cursors:

1) Implicit Cursors:

These are created by default when DML statements like insert, update and delete statements are executed. They are also created when select statements that returns just one row is executed.

2) Explicit cursors:

They must be created when you are executing a select statement that returns more than one rows.

Only one row can be processed at a time.

When you fetch a row the current row position moves to next row.

Implicit Cursors:

1) % FOUND

returns 1 if DML statement affected more than 1 row.

2) $\%NOTFOUND$ -

returns true if DML statements affected no rows.

3) $\%ISOPEN$ -

always returns false for implicit cursor.

4) $\%ROWCOUNT$ -

returns number of rows affected by an insert, update, delete statements.

Syntax:

declare

declaration statements

begin

execution statements

if sql $\%NOTFOUND$ then

- . -

else if sql $\%FOUND$ then

- . -

end if;

end;

Explicit Cursors:

Syntax:

declare

declaration

cursor _name_ is select statement;
begin

~~open _name_ is select statement;~~

open _name_;

loop

fetch _name_ into variables;

exit when _name_ % attribute

end loop;

close _name_;

end;

• CURSOR FOR LOOP

Syntax:

for recordindex in cursor_name

loop

{ statements }

end loop;

Date _____
Page _____

- Parameterized cursor:

Syntax:

```
declare
cursor cur_name (parameters)
is select-statements;
begin
open cur_name (params);
loop
fetch curname into variables
exit when curname % attribute;
} statements}
end loop;
close curname;
end;
```

- MSQL Cursors.

Syntax:

```
create procedure name (params)
begin
declarations
a declare cur_name cursor for
select statements;
open cur_name;
loop
fetch cur_name into variable;
if condn then
} statements}
end if;
```


end loop loop1;
close cur_name;
end;

• TEST CASES:

input	expected output	Status
call procs1()	1. old 2. new 3. old 4. new 5. new	Success

• Conclusion:

- Thus we implemented cursors in MySQL stored procedure with parameter.
- We learnt about types of cursors cursor for loop & parameterized cursor in PL/SQL.