
✅ AIM

To write a PL/SQL program using a parameterized cursor that merges data from the table O_Roll_Call into N_Roll_Call. If a roll number already exists in N_Roll_Call, that row should be skipped. Only new/unique data should be inserted.

✅ THEORY

✓ 1. Cursor

A cursor is a database pointer used to fetch and process rows one-by-one.
In PL/SQL/MySQL stored procedures, cursors allow row-level processing.

✓ 2. Parameterized Cursor

A parameterized cursor accepts parameters, which allows filtering rows dynamically based on user input.
Example:
cursor c1 is select * from table where roll_no > r1;

✓ 3. NOT FOUND Handler

When the cursor reaches the end of the result set, the NOT FOUND condition occurs.
We use a continue handler to set a flag and exit the loop safely.

✓ 4. NOT EXISTS

Before inserting a row into N_Roll_Call, we check whether that roll_no already exists.
This prevents duplicate entries.

✓ 5. Delimiter

MySQL treats semicolon (;) as end of a statement.
Stored procedures contain many semicolons, so we temporarily change the delimiter:

delimiter //

... procedure code ...

//

delimiter ;

This allows the entire procedure to execute correctly.

✅ **FULL CODE (SAME AS YOURS – NOT CHANGED)**

```
use newdb;

create table o_rollcall(roll_no int, name varchar(20),adress varchar(20));
create table n_rollcall(roll_no int, name varchar(20),adress varchar (20));

insert into o_rollcall values (1,"madhu","nashik");
insert into o_rollcall values (2,"renuka","pune");
insert into o_rollcall values (3,"saloni","mumbai");
insert into o_rollcall values (4,"vidhi", "nashik");

select * from o_rollcall;

delimiter //
create procedure p3(in r1 int)
begin
declare r2 int;
declare exit_loop boolean default false;

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;
//
delimiter ;

select * from n_rollcall;

call p3(3);
call p3(0);
```

💡 **CODE EXPLANATION (FULL, SIMPLE, CLEAR)**

1. Table Creation

Two tables created:

- **o_rollcall → source table**
- **n_rollcall → target table**

2. Sample Insert

Adds 4 sample rows into o_rollcall.

⭐ INSIDE THE PROCEDURE (MAIN LOGIC)

declare r2 int;

Stores one roll_no fetched from cursor.

declare exit_loop boolean default false;

Flag to stop the loop when cursor ends.

declare c1 cursor for select roll_no ...

Cursor that fetches roll numbers greater than the input (r1).
This makes it a parameterized cursor.

Example:
If r1 = 2 → cursor fetches roll_no = 3,4

---

declare continue handler for not found ...

When cursor has no more rows:

- **NOT FOUND occurs**
- **This handler sets exit_loop = true**
- **Loop will stop safely**

---

open c1;

Opens the cursor for fetching rows.

LOOP BEGINS

fetch c1 into r2;

Fetches the next roll_no into r2.

Duplicate Check

if not exists (select * from n_rollcall where roll_no = r2)

If roll number is not already present in n_rollcall → insert it.

Insert Row

insert into n_rollcall

select * from o_rollcall where roll_no = r2;

Copies the entire row with roll_no = r2.

Exit Condition

if exit_loop then

close c1;

leave e_loop;

end if;

When cursor ends:

- **exit_loop becomes true**
- **cursor closed**
- **loop exited**

🎯 OUTPUT FLOW (UNDERSTANDING THE RESULT)

✓ First Call:

call p3(3)

Cursor selects roll_no > 3 → only roll_no 4
→ inserts row 4 into n_rollcall

✓ Second Call:

call p3(0)

Cursor selects roll_no > 0 → (1,2,3,4)

Now insert:

- **1: new → inserted**
- **2: new → inserted**

- **3: new → inserted**
- **4: already exists → skipped**

Final n_rollcall contains roll_no: 1, 2, 3, 4

📝 **VIVA QUESTIONS WITH ANSWERS (Perfect for External Exam)**

Q1. What is the aim of this program?

To merge new roll numbers from O_Roll_Call into N_Roll_Call without inserting duplicates, using a parameterized cursor.

Q2. Why do we use delimiter in MySQL stored procedures?

Because stored procedures contain many semicolons.
Changing delimiter prevents MySQL from misinterpreting semicolons inside the procedure.

Q3. What is a parameterized cursor?

A cursor that accepts input parameters to filter the result set dynamically.

Q4. What is the role of NOT FOUND handler?

It detects the end of cursor records and sets a flag to exit the loop safely.

Q5. Why do we use NOT EXISTS?

To ensure that only new roll numbers are inserted.
Prevents duplicate records in n_rollcall.

Q6. Why is cursor used here?

Because we need to process records one-by-one and check each before insertion.

Q7. What does 'leave e_loop' do?

It exits the loop labelled e_loop.

Q8. What happens if a roll number is already present in n_rollcall?

The program skips insertion for that roll number.
