Cursors in MySQL

Row-by-row processing using cursors inside stored procedures

Instructor: Sriya Ivaturi

What is a Cursor?

- A cursor is a database object that allows you to process query results row by row.
- It is used when you need to perform actions on each row individually inside a stored procedure or trigger.
- Useful for complex, conditional, or sequential row processing that a normal SQL query cannot handle.

Bank Account Interest Calculation

- Each customer has a different account type or interest rate.
- Steps you perform using a cursor:
  - Loop through every account in the database.
  - Check account type (Savings, Current, Fixed).
  - Apply a different interest formula depending on the type.
  - Update the balance of that account individually.

This is a practical case where row-by-row processing is needed, which makes cursors ideal.

When to Use Cursors

- When a single SQL query is not sufficient
- Complex row-wise calculations
- Conditional logic on each row
- Dynamic updates or inserts

Steps to Use a Cursor

1. DECLARE the cursor
2. OPEN the cursor
3. FETCH rows into variables
4. LOOP through rows
5. CLOSE the cursor

Syntax of Cursor in MySQL

1. DECLARE cursor_name CURSOR FOR SELECT_statement;
2. OPEN cursor_name;
3. FETCH cursor_name INTO var1, var2, ...;
4. CLOSE cursor_name;

Cursor with Loop Structure

```
DECLARE done INT DEFAULT 0;
DECLARE name_var VARCHAR(50);
DECLARE emp_cursor CURSOR FOR SELECT
name FROM employees;
DECLARE CONTINUE HANDLER FOR NOT FOUND
SET done = 1;

OPEN emp_cursor;
read_loop: LOOP
 FETCH emp_cursor INTO name_var;
 IF done THEN LEAVE read_loop; END IF;
 SELECT CONCAT('Employee: ', name_var);
END LOOP;
CLOSE emp_cursor;
```

Explanation of Important Parts

- **DECLARE CURSOR:**
  - defines result set
- **CONTINUE HANDLER:**
  - handles end of data
- **LOOP:**
  - processes each row
- **LEAVE:**
  - exits loop after last row

Simple Cursor Example

```
DELIMITER $$
CREATE PROCEDURE ShowEmployees()
BEGIN
 DECLARE done INT DEFAULT 0;
 DECLARE emp_name VARCHAR(50);
 DECLARE emp_cursor CURSOR FOR SELECT name
FROM employees;
 DECLARE CONTINUE HANDLER FOR NOT FOUND SET
done = 1;
 OPEN emp_cursor;
 read_loop: LOOP
   FETCH emp_cursor INTO emp_name;
   IF done THEN LEAVE read_loop; END IF;
   SELECT CONCAT('Employee: ', emp_name);
 END LOOP;
 CLOSE emp_cursor;
END$$
DELIMITER ;
```

Questions

Q1: What happens if you forget to declare a CONTINUE HANDLER?

Q2: What is the difference between LEAVE and ITERATE?

Answers

- A1: Cursor throws error on FETCH after last row
- A2:
  - LEAVE: exits the loop
  - ITERATE: skips to next iteration

Summary

- Cursors allow row-wise logic in stored procedures
- CONTINUE HANDLER prevents runtime errors
- LOOP + FETCH for row processing
- Use LEAVE to exit loop safely