Using ITERATE in MySQL

- A Simple Guide to Flow Control in Loops
- MySQL Stored Procedures Tutorial

Instructor: Sriya Ivaturi

What is ITERATE?

- - ITERATE is used in loops like LOOP, WHILE, or REPEAT.
- - It skips the rest of the loop body and goes to the next iteration.
- - Works like 'continue' in many programming languages.

- Syntax:
- ITERATE label_name;

Syntax Structure

- [loop_label]: LOOP
- IF condition THEN
- ITERATE loop_label;
- END IF;

- -- More statements
- END LOOP;

- - The label_name is required to tell MySQL which loop to continue.

Basic Example with LOOP

```
DELIMITER $$


CREATE PROCEDURE simple_iterate()
BEGIN
  DECLARE i INT DEFAULT 0;
  simple_loop: LOOP
    SET i = i + 1;
    IF i = 3 THEN
      ITERATE simple_loop;
    END IF;
    SELECT i;
    IF i >= 5 THEN
      LEAVE simple_loop;
    END IF;
  END LOOP;
END$$
DELIMITER ;
```

- Skips i = 3 and prints 1, 2, 4, 5

Example with WHILE Loop

```
DELIMITER $$

CREATE PROCEDURE iterate_while()
BEGIN
  DECLARE i INT DEFAULT 0;

  WHILE i < 5 DO
    SET i = i + 1;

    IF i = 2 THEN
      ITERATE my_loop;
    END IF;

    SELECT i;
  END WHILE;
END$$

DELIMITER ;
```

Note: ITERATE needs a label. WHILE loop must be labeled to use it.

Key Points

- - ITERATE is only valid inside loops.
- - Requires a loop label.
- - Helps skip processing for certain values.
- - Often used with IF to filter data or control flow.