Database Fundamentals

Cybersecurity Department

Course Code: ***CBS 213***

Practical Lecture 8: Decision-Making in MySQL

Halal Abdulrahman Ahmed

Outlines

- What is decision-making in MySQL
- Difference between IF statement vs IF() function
- IF–THEN statement
- IF–THEN–ELSE statement
- IF–THEN–ELSEIF–ELSE statement
- How DELIMITER works

Learning Outcomes

By the end of this lecture, students will be able to:

- Explain how conditional logic works in MySQL
- Write IF statements inside stored procedures
- Choose the correct IF structure based on the problem
- Apply IF logic to real tables in the University database
- Debug errors related to IF blocks and DELIMITER

Introduction: Decision-Making in MySQL

Decision-making in **MySQL** refers to controlling the flow of SQL statements based on specific conditions. MySQL offers conditional control flow with IF statements, allowing us to execute blocks of SQL code depending on whether a condition is **true** or **false**. This is known as **decision-making**, or **conditional control flow**. MySQL provides different decision-making constructs:

- **IF–THEN**
- **IF–THEN–ELSE**
- **IF–THEN–ELSEIF–ELSE**

Introduction: Decision-Making in MySQL

The **IF statement** used inside stored programs is *not* the same as the **IF() function** used inside SELECT queries.

Example:

- IF(condition, value_if_true, value_if_false) → this is a *function*
- IF condition THEN ... END IF; → this is a *control flow statement*

Types of IF Statements in MySQL

MySQL supports **three** main forms of IF statements when writing procedures:

Type	Purpose
IF–THEN	Execute a block of code only if a condition is TRUE
IF–THEN–ELSE	Execute one block if TRUE, another block if FALSE
IF–THEN–ELSEIF–ELSE	Check multiple conditions sequentially

IF Statement vs IF() Function

Type	Purpose	Example
IF() function	Used inside SELECT	SELECT IF(Salary > 1200, 'High','Low');
IF statement	Used inside procedures	IF Salary > 1200 THEN SELECT 'High'; END IF;

University Database Tables

We will use the same structure from previous lectures:

- **Department** (DepartmentID, DepartmentName)
- **Lecturer** (LecID, LecName, DepartmentID, Salary)
- **Course** (CourseID, CourseName, DepartmentID)

Why DELIMITER Is Required

- Stored procedures contain multiple semicolons ;. MySQL needs to know:
- **When the procedure ends**, not every time it sees ;.
- So we temporarily change the delimiter:

```
DELIMITER $$

CREATE PROCEDURE ...
BEGIN
    ...
END $$

DELIMITER ;
```

Creating a Database (University Example)

```sql
CREATE DATABASE IF NOT EXISTS UniversityDB;
USE UniversityDB;

-- Department Table
CREATE TABLE IF NOT EXISTS Department (
    DepartmentID INT NOT NULL AUTO_INCREMENT,
    DepartmentName VARCHAR(50),
    PRIMARY KEY (DepartmentID)
);
```

Create database

Create Department table

Creating a Database (University Example)

```sql
-- Lecturer Table
CREATE TABLE IF NOT EXISTS Lecturer (
    LecID INT NOT NULL AUTO_INCREMENT,
    LecName VARCHAR(50),
    DepartmentID INT,
    Salary INT,
    PRIMARY KEY (LecID)
);

-- Course Table
CREATE TABLE IF NOT EXISTS Course (
    CourseID INT NOT NULL AUTO_INCREMENT,
    CourseName VARCHAR(50),
    DepartmentID INT,
    PRIMARY KEY (CourseID)
);
```

Create Lecturer table

Create Course table

```sql
-- Insert Sample Data
INSERT INTO Department (DepartmentName) VALUES
('IT'), ('Computer Science'), ('Business');

INSERT INTO Lecturer (LecName, DepartmentID, Salary) VALUES
('Dr. Ahmed', 1, 1500),
('Ms. Sara', 2, 1200),
('Mr. Dara', 1, 1000),
('Dr. Roj', 3, NULL);

INSERT INTO Course (CourseName, DepartmentID) VALUES
('Database Systems', 1),
('Java Programming', 1),
('Machine Learning', 2),
('Business Ethics', 3);
```

IF–THEN Statement

Execute SQL statements **only when a condition is TRUE**.

```
IF condition THEN
    statements;
END IF;
```

Check if a lecturer earns below the minimum salary (1200) (University Example)

```
DELIMITER $$

CREATE PROCEDURE CheckSalary (inputLecID INT)
BEGIN
    DECLARE lecSalary INT;

    SELECT Salary INTO lecSalary
    FROM Lecturer
    WHERE LecID = inputLecID;

    IF lecSalary < 1200 THEN
        SELECT 'This lecturer earns below the university minimum salary.' AS Result;
    END IF;

END $$

DELIMITER ;
```

IF–THEN–ELSE Statement

Run one block if TRUE, another if FALSE.

```
IF condition THEN
    statements;
ELSE
    else_statements;
END IF;
```

Check if a lecturer has a salary assigned (University Example)

```
DELIMITER $$

CREATE PROCEDURE CheckSalaryStatus (inputLecID INT)
BEGIN
    DECLARE lecSalary INT;

    SELECT Salary INTO lecSalary
    FROM Lecturer
    WHERE LecID = inputLecID;

    IF lecSalary IS NOT NULL THEN
        SELECT CONCAT('Lecturer has salary: ', lecSalary) AS Result;
    ELSE
        SELECT 'Salary not assigned for this lecturer.' AS Result;
    END IF;

END $$

DELIMITER ;
```

IF–THEN–ELSEIF–ELSE Statement

Check **multiple conditions**, similar to grading, salary categories, etc.

```
IF condition THEN
    statements;
ELSEIF condition2 THEN
    statements2;
ELSE
    else_statements;
END IF;
```

Categorize lecturer salary into LOW, MEDIUM, HIGH (University Example)

```
DELIMITER $$

CREATE PROCEDURE SalaryCategory (inputLecID INT)
BEGIN
    DECLARE lecSalary INT;

    SELECT Salary INTO lecSalary
    FROM Lecturer
    WHERE LecID = inputLecID;

    IF lecSalary IS NULL THEN
        SELECT 'No salary assigned.' AS Category;

    ELSEIF lecSalary < 1100 THEN
        SELECT 'LOW salary category' AS Category;
```

Categorize lecturer salary into LOW, MEDIUM, HIGH (University Example)

```
    ELSEIF lecSalary BETWEEN 1100 AND 1400 THEN
        SELECT 'MEDIUM salary category' AS Category;

    ELSE
        SELECT 'HIGH salary category' AS Category;
    END IF;

END $$

DELIMITER ;
```

References

- Budy, S., Reese, G., & Tahaghoghi, S. M. M. (Year unknown). *Learning MySQL*. O'Reilly Media.

- Beaulieu, A. (Year unknown). *MySQL Cookbook* (4th ed.). O'Reilly Media.


Any
Question