Lecture Notes – ControlFlowStoredProcedures

Section 1: Lecture Summary

This lecture covers control flow statements within stored procedures, specifically focusing on conditional logic using IF-ELSE statements. The content demonstrates how to declare variables, implement decision-making logic, and execute different SQL statements based on conditions. Two practical examples are presented: checking product stock levels and determining employee bonus eligibility.

Section 2: Key Concepts and Explanations

Stored Procedure Fundamentals with Control Flow

A **stored procedure** is a precompiled SQL statement stored in the database that can be executed repeatedly. Control flow allows procedures to make decisions and perform different operations based on conditions.

Variable Declaration

Variables must be declared before use within a procedure. The syntax is:

DECLARE variable_name data_type;

Variables store values temporarily and can be assigned results from queries using the INTO clause.

Conditional Statements (IF-ELSE)

The IF-ELSE structure evaluates a condition and executes different code blocks based on the result. The syntax is:

```
IF condition THEN
   statements_if_true;
ELSE
   statements_if_false;
END IF;
```

The condition can use comparison operators: greater than (>), less than (<), equal to (=), greater than or equal to (>=), less than or equal to (<=), or not equal to (<>).

Storing Query Results in Variables

Single-value query results can be captured using the INTO keyword:

```
SELECT column INTO variable_name FROM table WHERE condition;
```

This allows the result to be used in subsequent conditional logic.

Procedure Invocation

Stored procedures are executed using the CALL statement:

```
CALL procedure_name(parameters);
```

Section 3: Example Code and Use Cases

Example 1: Product Stock Status Check

Scenario: Check if a product has healthy stock or low stock based on a threshold of 50 units.

```
CREATE PROCEDURE CheckStock(IN pProductID INT)
BEGIN
    DECLARE stock_quantity INT;

    SELECT Stock INTO stock_quantity
    FROM Products
    WHERE ProductID = pProductID;

    IF stock_quantity > 50 THEN
        SELECT 'Product in stock' AS StockStatus;
    ELSE
        SELECT 'Low stock' AS StockStatus;
    END IF;
END
```

How it works:

1. The procedure accepts a product ID as input parameter

2. The SELECT INTO statement retrieves the stock value and stores it in the stock_quantity variable

3. The IF-ELSE logic compares the stock quantity against the threshold of 50

4. Different messages are returned based on the condition

Execution:

```
CALL CheckStock(106);
```

If ProductID 106 has a stock of 20, the result will be "Low stock".

Example 2: Employee Bonus Eligibility

Scenario: Determine if an employee qualifies for a bonus based on salary threshold of 80,000.

```
CREATE PROCEDURE CheckBonusEligibility(IN pEmpID INT)
BEGIN
    DECLARE emp_salary DECIMAL(10, 2);

    SELECT Salary INTO emp_salary
    FROM Employees
    WHERE EmpID = pEmpID;

    IF emp_salary >= 80000 THEN
        SELECT CONCAT(FirstName, ' ', LastName, ' is eligible for bonus')
AS BonusStatus
        FROM Employees
        WHERE EmpID = pEmpID;
    ELSE
        SELECT CONCAT(FirstName, ' ', LastName, ' is not eligible for
bonus') AS BonusStatus
        FROM Employees
        WHERE EmpID = pEmpID;
    END IF;
END
```

How it works:

1. The procedure accepts an employee ID as input

2. The employee's salary is retrieved and stored in the emp_salary variable

3. The condition checks if salary is greater than or equal to 80,000

4. Eligible employees receive a different message than ineligible ones

Execution:

```
CALL CheckBonusEligibility(5);
```

Section 4: Key Takeaways

Variable Declaration: Always declare variables before using them in procedures; use meaningful names with prefixes (like p for parameters) for clarity.

Conditional Logic: IF-ELSE statements enable decision-making within procedures; conditions use standard comparison operators to evaluate variable values.

Query Result Storage: Use the INTO clause to capture single values from SELECT queries into variables for use in subsequent logic.

Procedure Calls: Execute stored procedures with the CALL statement followed by the procedure name and required parameters in parentheses.

Practical Applications: These patterns are useful for business logic such as inventory management, eligibility determination, and status evaluations based on data thresholds.