Lecture Notes – MultipleParameters

Section 1: Lecture Summary

This lecture covers the creation and execution of stored procedures with multiple parameters in MySQL. The focus is on building a practical procedure that modifies product prices in the eCommerceDB database. The example demonstrates how to define input parameters, write UPDATE statements within a procedure, and call the procedure with appropriate arguments to perform database modifications efficiently.

Section 2: Key Concepts and Explanations

Stored Procedures with Multiple Parameters

A stored procedure is a reusable SQL statement saved in the database. When a procedure requires multiple inputs, you define multiple **IN parameters** in the procedure signature. Each parameter requires a name and data type specification.

Parameter Declaration

Parameters are declared between the procedure name and the BEGIN keyword. The syntax follows the pattern: IN parameter_name data_type. For price-related operations, the DECIMAL(10,2) data type is appropriate, allowing 10 total digits with 2 decimal places.

UPDATE Statements in Procedures

The procedure body contains standard SQL statements. The UPDATE statement modifies existing records by setting column values based on WHERE clause conditions. When referencing procedure parameters within the UPDATE statement, you use the parameter name directly (prefixed with the parameter indicator if needed in your MySQL version).

Procedure Execution

To execute a stored procedure, use the CALL keyword followed by the procedure name and the required arguments in parentheses. The arguments must match the parameter order and data types defined in the procedure.

Section 3: Example Code and Use Cases

Creating a Stored Procedure to Update Product Price

```
DELIMITER //

CREATE PROCEDURE UpdateProductPrice(
    IN p_product_id INT,
    IN p_new_price DECIMAL(10,2)
)
BEGIN
    UPDATE Products
    SET Price = p_new_price
    WHERE ProductID = p_product_id;
END//

DELIMITER ;
```

This procedure accepts two input parameters: the product ID to identify which product to modify, and the new price value to apply. The UPDATE statement targets the Products table, sets the Price column to the new value, and filters by ProductID using the WHERE clause.

Calling the Procedure

```
CALL UpdateProductPrice(101, 399.99);
```

This call updates the product with ProductID 101 to a new price of 399.99. After execution, you can verify the change by querying the Products table.

Verification Query

```
SELECT ProductID, ProductName, Price
FROM Products
WHERE ProductID = 101;
```

Section 4: Key Takeaways

Stored procedures with multiple parameters enable you to create reusable database operations without writing the same SQL repeatedly. By defining clear parameter names and data types, you make procedures self-documenting and maintainable. The UPDATE statement within a procedure follows standard SQL syntax, making it straightforward to implement data modifications. This approach reduces code duplication and allows for controlled, parameterized database changes through simple procedure calls.