Stored procedures

Stored procedures in databases like MySQL are precompiled SQL statements that are stored in the database itself and can be executed multiple times. They are beneficial for encapsulating business logic and complex queries, promoting code reuse, and enhancing performance by reducing network traffic.

Difference between Procedure and Function

1. Purpose:

- **Procedure:** Used for performing an operation or a series of operations. Procedures can contain DML (Data Manipulation Language) and DDL (Data Definition Language) statements.
- **Function:** Returns a value and is primarily used for computations.

2. Return Type:

- **Procedure:** Does not return a value. It may have OUT or INOUT parameters to pass values back to the caller.
- **Function:** Returns a single value using the RETURN statement.

3. Usage in SQL Statements:

- **Procedure:** Can be called using CALL statement or integrated into SQL queries.
- **Function:** Can be used directly in SQL statements or assigned to variables.

4. Transactions:

- **Procedure:** Can contain transaction control statements (BEGIN, COMMIT, ROLLBACK) to manage transactions.
- **Function:** Cannot contain transaction control statements.

5. Exception Handling:

- **Procedure:** Can handle exceptions using DECLARE HANDLER.
- **Function:** Does not support exception handling.

6. Use of SELECT:

- **Procedure:** Can execute SELECT statements but does not return a result set directly.
- **Function:** Can execute SELECT statements and return a result set.

7. Calling From SQL:

- **Procedure:** Called using CALL procedure_name();.
- **Function:** Used in SQL queries like SELECT function_name();.

Creating a Procedure

To create a procedure in MySQL, you use the CREATE PROCEDURE statement followed by the procedure name, parameters (if any), and the SQL statements that define the procedure's functionality.

Syntax for Creating a Procedure

```
CREATE PROCEDURE procedure_name (parameter_list)
    [characteristics]
    BEGIN
        -- SQL statements
    END //

CREATE TABLE Employees10 (
    EmployeeID INT PRIMARY KEY AUTO_INCREMENT,
    FirstName VARCHAR(50),
    LastName VARCHAR(50),
    Position VARCHAR(50),
    Salary DECIMAL(10, 2)

);
```

Example of Creating a Procedure

Let's create a simple procedure that inserts a new employee record into an Employees table.

```
CREATE PROCEDURE InsertEmployee (
    IN p_first_name VARCHAR(50),
    IN p_last_name VARCHAR(50),
    IN p_position VARCHAR(50),
    IN p_salary DECIMAL(10, 2)
)
BEGIN
    INSERT INTO Employees (FirstName, LastName, Position, Salary,
HireDate)

VALUES (p_first_name, p_last_name, p_position, p_salary);

END
```

Calling a Procedure

After creating a procedure, you can call it using the CALL statement:

```
CALL InsertEmployee('Rahul', 'Sharma', 'Manager', 60000, '2020-01-
15');
```

Benefits of Stored Procedures

- **Code Reusability:** Procedures can be called multiple times, promoting code reuse.
- **Improved Performance:** Reduces network traffic by executing a batch of SQL statements at once.
- **Enhanced Security:** Users can execute procedures without needing direct access to underlying tables.
- **Encapsulation of Logic:** Business logic is encapsulated within the database, improving maintainability.

Stored procedures are powerful tools in database management systems for managing complex SQL logic and improving application performance and security. They provide a structured way to encapsulate operations that need to be executed frequently or as part of larger transactions.

Creating a stored procedure in MySQL that includes both IN and OUT parameters involves defining the procedure, specifying the parameters, and writing the SQL statements that the procedure will execute. Here is an example to illustrate this:

Example Scenario

Suppose you have a products table with the following columns: product_id, product_name, and price. You want to create a stored procedure that takes a product ID as an input (IN parameter) and returns the product name and price as output (OUT parameters).

Step-by-Step Instructions

1. **Create the Table**

   First, ensure you have a table to work with. You can create the products table with the following SQL:

```
CREATE TABLE products (
    product_id INT PRIMARY KEY,
    product_name VARCHAR(100),
    price DECIMAL(10, 2)
);

INSERT INTO products (product_id, product_name, price)
VALUES
(1, 'Product A', 10.00),
(2, 'Product B', 20.00),
(3, 'Product C', 30.00);
```

2. **Create the Stored Procedure**

   Next, create a stored procedure that includes IN and OUT parameters:

```
CREATE PROCEDURE GetProductDetails (
    IN in_product_id INT,
```

```
    OUT out_product_name VARCHAR(100),
    OUT out_price DECIMAL(10, 2)
)
BEGIN
    SELECT product_name, price
    INTO out_product_name, out_price
    FROM products
    WHERE product_id = in_product_id;
END
```

3. **Call the Stored Procedure**

   To call the stored procedure and retrieve the output parameters, use the following commands:

```
-- Declare variables to hold the output values
SET @product_name = '';
SET @price = 0;

-- Call the stored procedure
CALL GetProductDetails(1, @product_name, @price);

-- Retrieve the output values
SELECT @product_name AS product_name, @price AS price;
```

Explanation

1. **Table Creation**: The products table is created with columns for product_id, product_name, and price. Sample data is inserted into the table.
2. **Stored Procedure Creation**:
   - The CREATE PROCEDURE statement defines the procedure GetProductDetails with one IN parameter (in_product_id) and two OUT parameters (out_product_name and out_price).

- The SELECT statement within the procedure retrieves the product_name and price for the given product_id and stores them in the OUT parameters.

3. **Calling the Procedure**:

   - Variables @product_name and @price are declared to hold the output values.
   - The CALL statement invokes the stored procedure with the input product ID and the output variables.
   - The SELECT statement retrieves and displays the output values stored in the variables.

This example demonstrates how to create and use a stored procedure with IN and OUT parameters in MySQL.