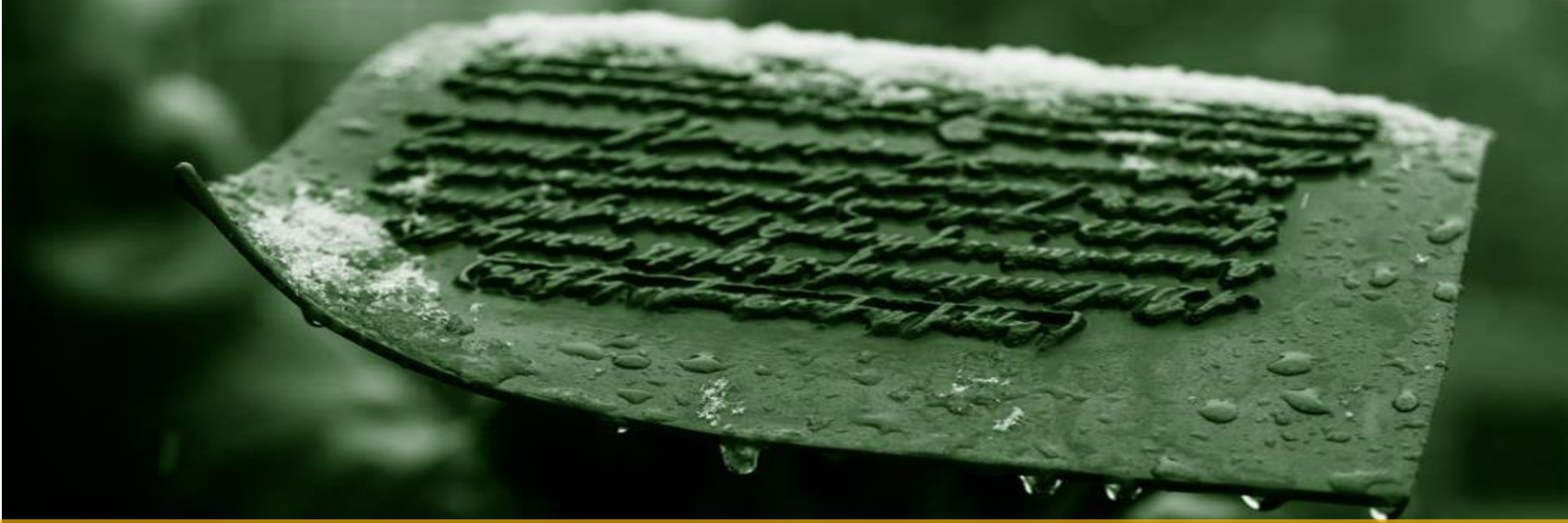

MODULE 10 – STORED PROCEDURES IN MYSQL SERVER

IT 207 – IT Programming

GEORGE MASON UNIVERSITY

Lecture Outline

- Introduction to MySQL Stored Procedures
- Creating/Deleting/Altering Stored Procedures
- Using Variables and IN/OUT/INOUT Parameters
- Using Conditional Statements

What is a Stored Procedure in MySQL?

- A stored procedure is a series of SQL instructions associated with a name that is stored and executed within the database server.
- SQL stored procedures can accept multiple inputs and give multiple outputs
- Stored procedures can be called from:
  - Standard languages
  - Scripting languages
  - MySQL command prompt

Advantages of Stored Procedures

- Security:
  - Stored procedures are always stored on the database server.
  - Access privileges for application are given to only access the stored procedure rather than having access to the underlying table
- Optimizations:
  - SQL can be optimized by the DBMS compiler
  - Reduce network traffic between applications and MySQL server
- Code Reuse
  - Reduces efforts of duplicating the same code in an application
  - Standardized processing
  - Specialization among developers

STORED PROCEDURE SQL STATEMENTS

CREATING STORED PROCEDURES – 1

- Syntax for creating a Stored Procedure

```
CREATE PROCEDURE <proc-name> (parameter list )
  BEGIN
        -- execution code
  END;
```

- CREATE PROCEDURE creates a procedure with proc-name
- The parameter list has the following specification:

```
[IN | OUT | INOUT] <param_name> <param_type>
```

  - IN mode allows values to be passed into the procedure,
  - OUT mode allows values to be passed back from procedure to the calling program

CREATING STORED PROCEDURES – 2

MySQL is NOT case sensitive

- Stored Procedure Syntax

```
CREATE PROCEDURE <proc-name> (parameter list )
  BEGIN
        -- execution code
  END;
```

  - CREATE PROCEDURE creates a procedure with proc-name
  - The parameter list has the following specification:

```
[IN | OUT | INOUT] <param_name> <param_type>
```

    - IN mode allows you to pass values into the procedure,
    - OUT mode allows you to pass value back from procedure to the calling program

- Stored procedures end with a semicolon

Changing the Delimiter Character

- MySQL uses the semicolon as a *delimiter* to separate SQL statements and execute each separately.
- In stored procedures
  - SQL are separated by semicolons
  - A semicolon is used to terminate the storing procedure

MySQL will not treat the whole stored procedure as a single statement, but as many statements.

Redefine the delimiter to be a something other than a semicolon temporarily.

Steps to Create a Stored procedure

1. Change the delimiter character.
2. Define the procedure header using the CREATE PROCEDURE statement.
3. Mark the beginning of the procedure using the BEGIN keyword.
4. Write the procedure statements.
5. Mark the end of the procedure using the END keyword.
6. Change the delimiter back to semicolon.

Example 1 - Create a Stored Procedure

- Create a procedure that retrieves all the product codes and product names from the Classicmodels DB
  - The SQL statement

```
SELECT productCode, productName FROM products;
```

  - The equivalent Stored Procedure

```
DELIMITER $$
CREATE PROCEDURE getProductCodeName ()
    BEGIN
            SELECT productCode, productName FROM products;
   END;
$$
DELIMITER ;
```

EXAMPLE 1 – ON MYSQL COMMAND LINE

```
mysql> DELIMITER $$
mysql> CREATE PROCEDURE getProductCodeName ()  BEGIN SELECT productCode, productName FROM products; END;$$
Query OK, 0 rows affected (0.00 sec)

mysql> DELIMITER ;
mysql> CALL getProductCodeName();
+-------------+----------------------------------------------+
| productCode | productName                                  |
+-------------+----------------------------------------------+
| S10_1678    | 1969 Harley Davidson Ultimate Chopper        |
| S10_1949    | 1952 Alpine Renault 1300                     |
| S10_2016    | 1996 Moto Guzzi 1100i                        |
| S10_4698    | 2003 Harley-Davidson Eagle Drag Bike         |
| S10_4757    | 1972 Alfa Romeo GTA                          |
| S10_4962    | 1962 LanciaA Delta 16V                       |
| S12_1099    | 1968 Ford Mustang                            |
| S12_1108    | 2001 Ferrari Enzo                            |
| S12_1666    | 1958 Setra Bus                               |
| S12_2823    | 2002 Suzuki XREO                             |
| S12_3148    | 1969 Corvair Monza                           |
| S12_3380    | 1968 Dodge Charger                           |
| S12_3891    | 1969 Ford Falcon                             |
| S12_3990    | 1970 Plymouth Hemi Cuda                      |
| S12_4473    | 1957 Chevy Pickup                            |
| S12_4675    | 1969 Dodge Charger                           |
| S18_1097    | 1940 Ford Pickup Truck                       |
| S18_1129    | 1993 Mazda RX-7                              |
| S18_1342    | 1937 Lincoln Berline                         |
| S18_1367    | 1936 Mercedes-Benz 500K Special Roadster     |
| S18_1589    | 1965 Aston Martin DB5                        |
| S18_1662    | 1980s Black Hawk Helicopter                  |
| S18_1749    | 1917 Grand Touring Sedan                     |
| S18_1889    | 1948 Porsche 356-A Roadster                  |
| S18_1984    | 1995 Honda Civic                             |
| S18_2238    | 1998 Chrysler Plymouth Prowler               |
| S18_2248    | 1911 Ford Town Car                           |
```

Removing/Altering Stored Procedures

- Syntax for removing a Stored Procedure

```
DROP PROCEDURE <proc-name> ;
```

  - The DROP statement will delete the procedure from the MySQL server.
- Altering a Stored Procedure is NOT supported on the MySQL command line.
- To change the body of a Stored Procedure on MySQL command line
  1. Drop the procedure.
  2. Create a new procedure with the updated statements.

IF EXISTS – IF NOT EXISTS KEYWORDS

- The IF EXISTS and IF NOT EXISTS keywords can be used when creating, updating and/or deleting stored procedures.

```
CREATE PROCEDURE [IF NOT EXISTS] <proc-name> (parameter list)
        procedure_body;
```

```
DROP PROCEDURE [IF EXISTS] <proc-name> ;
```

STORED PROCEDURE VARIABLES & PARAMETERS

Declaring Variables

- The DECLARE keyword is used to declare variables in stored procedures
- The DECLARE statement follows the BEGIN keyword
- The declaration statement has the following syntax:

```
DECLARE variable_name datatype(size) [DEFAULT default_value];
```

  - The variable name must follow the naming rules of MySQL table column names.
  - The datatype refers to MySQL data types such as INT, VARCHAR , and DATETIME.
  - Optionally, a default value can be specified for the variable.
    - A variable without a DEFAULT value will be assigned the value NULL.

Example 2 – Declaring Variables in Stored Procedures

- Declaring a Decimal variable

```
DECLARE totalSale DEC(10,2) DEFAULT 0.0;
```

- Declaring multiple variables using the same DECLARE keyword

```
DECLARE x, y INT DEFAULT 0;
```

VARIABLE ASSIGNMENT STATEMENT

- The SET statement is used to assign a value to a variable.

```
SET variable_name = value;
```

- Example

```
DECLARE total INT DEFAULT 0;
SET total = 10;
```

- A value can be assigned to a variable using the SELECT INTO statement to assign the result of a query to a variable

```
DECLARE productCount INT DEFAULT 0;

SELECT COUNT(*)
INTO productCount
FROM products;
```

Example 3 – Stored Procedures with Variables

- Write a stored procedure that counts the number of rows in the Orders table

```
DELIMITER $$

CREATE PROCEDURE GetTotalOrder()
BEGIN
        DECLARE totalOrder INT DEFAULT 0;

SELECT COUNT(*)
   INTO totalOrder
   FROM orders;

   SELECT totalOrder;
END$$

DELIMITER ;
```

Stored Procedure Parameters

- A parameter in a stored procedure has one of three modes: IN, OUT, or INOUT.
- IN parameter
  - It is the default mode
  - An argument is passed to the stored procedure.
  - Parameters are passed as a copy.
- Out Parameter
  - Its the initial value cannot be accessed by the stored procedure
  - It can be changed in the stored procedure and its new value is passed back to the calling program.
- INOUT parameter
  - It is a combination of IN and OUT parameters.
  - The calling program can pass an argument and its value will be accessible to the stored procedure
  - The stored procedure can change the value of the parameter

Example 4 – IN Parameter

- Finds all offices that are in a country specified by the input parameter countryName:

```
DELIMITER //

CREATE PROCEDURE GetOfficeByCountry( IN countryName VARCHAR(255) )
BEGIN
        SELECT *
        FROM offices
        WHERE country = countryName;
END //

DELIMITER ;
```

EXAMPLE 4 – MYSQL RESULTS

- Finds all offices that are located in a country specified by the input parameter countryName:

```
CALL GetOfficeByCountry('USA');
```

```
mysql> DELIMITER //
mysql>
mysql> CREATE PROCEDURE GetOfficeByCountry(
    -> IN countryName VARCHAR(255)
    -> )
    -> BEGIN
    -> SELECT *
    ->  FROM offices
    -> WHERE country = countryName;
    -> END //
Query OK, 0 rows affected (0.08 sec)

mysql>
mysql> DELIMITER ;
mysql> CALL GetOfficeByCountry('USA');
+------------+---------------+-----------------+----------------------+--------------+-------+---------+------------+-----------+
| officeCode | city          | phone           | addressLine1         | addressLine2 | state | country | postalCode | territory |
+------------+---------------+-----------------+----------------------+--------------+-------+---------+------------+-----------+
| 1          | San Francisco | +1 650 219 4782 | 100 Market Street    | Suite 300    | CA    | USA     | 94080      | NA        |
| 2          | Boston        | +1 215 837 0825 | 1550 Court Place     | Suite 102    | MA    | USA     | 02107      | NA        |
| 3          | NYC           | +1 212 555 3000 | 523 East 53rd Street | apt. 5A      | NY    | USA     | 10022      | NA        |
+------------+---------------+-----------------+----------------------+--------------+-------+---------+------------+-----------+
3 rows in set (0.02 sec)

Query OK, 0 rows affected (0.02 sec)
```

TY

Example 5 – OUT Parameter

- Print the number of orders listed by order status

```
DELIMITER $$

CREATE PROCEDURE GetOrderCountByStatus (
        IN  orderStatus VARCHAR(25),
        OUT total INT
)
BEGIN
        SELECT COUNT(orderNumber)
        INTO total
        FROM orders
        WHERE status = orderStatus;
END$$

DELIMITER ;
```

EXAMPLE 5 – MYSQL RESULTS

- Print the number of orders listed by order status

```
CALL GetOrderCountByStatus('Shipped',@total);
SELECT @total;
```

  - @total is a session variable that holds the value returned from the stored procedure

```
mysql> DELIMITER $$
mysql>
mysql> CREATE PROCEDURE GetOrderCountByStatus (
    -> IN  orderStatus VARCHAR(25),
    -> OUT total INT
    -> )
    -> BEGIN
    -> SELECT COUNT(orderNumber)
    -> INTO total
    -> FROM orders
    -> WHERE status = orderStatus;
    -> END$$
Query OK, 0 rows affected (0.06 sec)

mysql>
mysql> DELIMITER ;
mysql> CALL GetOrderCountByStatus('Shipped',@total);
Query OK, 1 row affected (0.03 sec)

mysql> SELECT @total;
+--------+
| @total |
+--------+
|    303 |
+--------+
1 row in set (0.00 sec)
```

Example 6 – INOUT Parameter

- Write a stored procedure that increments a counter

```
DELIMITER $$

CREATE PROCEDURE SetCounter(
        INOUT counter INT,
        IN inc INT
)
BEGIN
        SET counter = counter + inc;
END$$

DELIMITER ;
```

Example 6 – INOUT Parameter MySQL Run

- Write a stored procedure that increments a counter

```
SET @counter = 1;
CALL SetCounter(@counter,1); -- 2
CALL SetCounter(@counter,1); -- 3
CALL SetCounter(@counter,5); -- 8
SELECT @counter; -- 8
```

```
mysql> DELIMITER $$
mysql>
mysql> CREATE PROCEDURE SetCounter(
    -> INOUT counter INT,
    ->     IN inc INT
    -> )
    -> BEGIN
    -> SET counter = counter + inc;
    -> END$$
Query OK, 0 rows affected (0.08 sec)

mysql>
mysql> DELIMITER ;
mysql> SET @counter = 1;
Query OK, 0 rows affected (0.00 sec)

mysql> CALL SetCounter(@counter,1); -- 2
Query OK, 0 rows affected (0.01 sec)

mysql> CALL SetCounter(@counter,1); -- 3
Query OK, 0 rows affected (0.00 sec)

mysql> CALL SetCounter(@counter,5); -- 8
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT @counter; -- 8
+----------+
| @counter |
+----------+
|        8 |
+----------+
```

STORED PROCEDURE CONDITIONAL STATEMENT

Conditional Statements

- In SQL, the conditional statement has three forms:
  - Simple IF-THEN statement
    - Executes a set of SQL statements if a specified condition is true.
  - IF-THEN-ELSE statement
    - Executes a set of SQL statements if a specified condition is true and executes another set of SQL statement if a specified condition is false.
  - IF-THEN-ELSEIF-ELSE statement.
    - Chains several IF-ELSE statements together

EXAMPLE 7 – CONDITIONAL STATEMENTS

- Write a stored procedure to set the customer level based on the customer's credit limit

```
DELIMITER $$
CREATE PROCEDURE GetCustomerLevel(
   IN  pCustomerNumber INT,
   OUT pCustomerLevel  VARCHAR(20))
BEGIN
   DECLARE credit DECIMAL DEFAULT 0;
   SELECT creditLimit
   INTO credit
   FROM customers
   WHERE customerNumber = pCustomerNumber;
   IF credit > 50000 THEN
      SET pCustomerLevel = 'PLATINUM';
   ELSEIF credit <= 50000 AND credit > 10000 THEN
      SET pCustomerLevel = 'GOLD';
   ELSE
      SET pCustomerLevel = 'SILVER';
   END IF;
END $$
DELIMITER ;
```

Stored Procedure Statements & Other Advanced Topics

- SQL supports other stored procedure statements
  - CASE Statement
  - Looping statements
  - Error Handling
  - Etc…

https://www.mysqltutorial.org/mysql-stored-procedure-tutorial.aspx

SUMMARY

- In MySQL a stored procedure is a collection of pre-compiled SQL statements stored inside the database.
- Stored Procedure increases the performance of the applications.
- Stored procedures can have private variables, and can take in values and return values through parameters
- SQL support different control structures as Conditional statements and Looping statements