IT Department

User-Defined Functions in MySQL

Soma Soleiman Zadeh

Database Systems II (IT 226)

Spring 2024 - 2025

Week 9

April 08, 2025

Outline

- **Stored Routines** (Subprograms) in MySQL
  - User-defined **Functions**
  - **Procedures**
- **Variables** in MySQL
- **Delimiter**
- Syntax of **Creating Function** in MySQL

Stored Routines in MySQL

- A **stored routine** is a set of SQL statements to perform a particular task, which can be stored in the server.
- MySQL **Stored Routines** (Subprograms):
    1. **Functions:** A function **must** return a single value; mainly used to **compute and return a value**.
    2. **Procedures:** A procedure may or may not return a value; mainly used to **manipulate the data**.

Functions in MySQL

- A **MySQL function** is a **named, reusable sequence of SQL statements**.
- It takes **zero** or **more arguments** as **input**, performs specific operations, and **returns a value**.
- **Functions** can be utilized within **SQL statements**, making them a powerful tool for data manipulation and calculation.

Variable in MySQL

- **Variables** are containers used to store data.
- Three types of variables in MySQL:
  - **User-Defined** Variable:
    - ✓ The **user-defined variable** enables us to store a value in one statement and later refer it to another statement.
  - **Local** Variable
    - ✓ The scope of the **local variable** is in a function or procedure block in which it is declared.
  - **System** Variable
    - ✓ MySQL contains various **system** variables that configure its operation.

User-Defined Variable

- The user-defined variable name starts with **@ symbol**.
- **Declare and Initialize a user-defined variable** in MySQL:

```
SET @variable_name = value;
```

- **Example:**

```
SET @MyName= 'Kate';

SET @gpa= 3.2;
```

User-Defined Variable

- After initializing a variable, you can change its value using a **SELECT INTO** statement.
- Syntax of **SELECT INTO** Statement:

```
SELECT  new_value  INTO  @variable_name;
```

- **Example:**

```
SET  @salary= 2000;

SELECT  3000  INTO @salary;
```

User-Defined Variable

- The **SELECT** statement is used to display the variable's data.
- **Syntax:**

```
SELECT  @variable_name;
```

- **Example:**

```
SELECT  @MyName;

SELECT  @salary;
```

Local Variable

- The **local variable** is declared inside a function or procedure block.
- Local variable name is not prefixed by **@** symbol.
- **Declare a local variable** in MySQL:

```
DECLARE variable_name datatype;
```

- **Example:**

```
DECLARE MyName varchar(100);
DECLARE gpa decimal(5,2);
```

Local Variable

- After declaring a local variable, you can initialize or change its value using **SELECT INTO** statement.
- Syntax of **SELECT INTO** Statement:

```
SELECT new_value INTO variable_name;
```

- **Example:**

```
DECLARE gpa decimal(5,2);
SELECT 3.2 INTO gpa;
```

What is DELIMITER?

- In MySQL, a **delimiter** is a special character(s) used to signal the end of an SQL statement.
- The **semicolon (;)** is the default delimiter in MySQL, which is used to separate statements from each other.
- When working with complex statements in procedures and functions, semicolons within the statement can cause issues with execution.
- **Solution:** Temporarily redefine the delimiter to a different character (such as **$$** or **//**) using the **DELIMITER** command.

DELIMITER Command in MySQL

- Syntax of **DELIMITER** Command:

```
DELIMITER   delimiter_character
Statement   delimiter_character
DELIMITER   ;
```

- Example of redefining the delimiter to $$:

```
DELIMITER   $$
Select * from products$$
DELIMITER   ;
```

Syntax of Creating Function in MySQL

```
DELIMITER //
CREATE FUNCTION function_name (input_parameter1 datatype1,
                               input_parameter2 datatype2,
                               ... )
RETURNS return_datatype
BEGIN
        <Declaration Part of Function>
        <Execution Part of Function>
        RETURN return_value;
END//
DELIMITER ;
```

Invoking (Calling) the Function

- When a function is created, it is not executed until the function is invoked (called).
  - **Step 1** – Creating a function
  - **Step 2** – Invoking (Calling) the function

- How to invoke (call) a function?
  - A function can be called by specifying its **name** and **parameter list**.
  - To display the return value of a function when it is called, use the **SELECT** statement.

```
SELECT function_name (parameter_values);
```

Creating Function – Example #1

- Create a function named **SumTwoNumbers** that takes two integer numbers and returns the sum of them.
- Then invoke (call) the function by setting some user-defined variables.

Creating Function

```
DELIMITER   $$
CREATE FUNCTION SumTwoNumbers (a int, b int)
RETURNS int DETERMINISTIC
BEGIN
        DECLARE result int;
        SELECT a+b into result;
        RETURN result;
END $$
DELIMITER   ;
```

Calling Function

```
SET @a = 3;
SET @b = 6;
SELECT SumTwoNumbers (@a , @b);
```

Calling Function and storing the return value in a variable

```
SET @a = 3;
SET @b = 6;
SET @c = 0;
SELECT SumTwoNumbers (@a , @b) INTO @c;
SELECT @c;
```

Creating Function – Example #2

- Consider the following table (**item**) containing data related to items stocked in a store stock.
- Create a function named **averagePrice** to calculate and return the average price of all items in the **item** table.

```
DELIMITER %%
CREATE FUNCTION averagePrice ( )
RETURNS DECIMAL(5,2) DETERMINISTIC
BEGIN
        DECLARE avgprice DECIMAL(5,2);
        SELECT avg(price) INTO avgprice FROM Item;
        RETURN avgprice;
END%%
DELIMITER   ;
```

Item

ItemID	ItemName	Price	StockQuantity
1	Calculator	20	31
2	Punching Machine	15	25
3	Scissors	5	100

- Invoke (call) the function.

```
SET @result = 0;
SELECT averagePrice( ) INTO @result;
SELECT @result;
```

Class Work

- Consider the following table (**Employee**) containing employees' data.
- Create a function named **UpdatedSalary** to **take** the **ID** of an employee and **return** his/her updated salary amount considering a 10% increase.

```
DELIMITER //
CREATE FUNCTION UpdatedSalary (ID int)
RETURNS decimal(6 , 2) DETERMINISTIC
BEGIN
        DECLARE oldsalary decimal(6 , 2);
        DECLARE newsalary decimal(6 , 2);
        SELECT Salary INTO oldsalary FROM Employee WHERE EID = ID;
        SELECT (oldsalary+(10/100)*oldsalary) INTO newsalary;
        RETURN newsalary;
END//
DELIMITER ;
```

Employee

EID	EName	Salary
1	Sina	2000.5
2	Mahmood	1000
3	Sonia	3000

- Invoke (call) the function by taking employee ID=2.

```
SET @result = 0;
SELECT UpdatedSalary (2) INTO @result;
SELECT @result;
```