IT Department

Procedures in MySQL

Soma Soleiman Zadeh

Database Systems II (IT 216)

Spring 2025 - 2026

Week 10

April 6, 2026

Outline

- **Procedures** in MySQL
- Syntax of Creating **Procedure** in MySQL
- **IN** Parameter
- **OUT** Parameter
- **INOUT** Parameter
- **INVOKING** a Procedure

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**.

Function or Procedure?

- Finding name of a book with minimum price. → *Function*
- Updating the price of a book named "Machine Learning" to 40. → *Procedure*
- Adding 5 dollars to all books' prices. → *Procedure*
- Finding the average price of all books. → *Function*

Book

ISBN	bookName	Price
113	Intro to Python	45
114	Machine Learning	30
115	Computer Networks	75

What is Procedure?

- **Procedure** is a block of code bundled under a name that can be reused for two types of tasks:

Highly-frequent tasks

- Authentication to Facebook

Highly Sensitive Activity

- Amazon Checkout

- Usually **INSERT**, **UPDATE** or **DELETE**.

Procedures in MySQL

- **Procedures** can be reused over and over again.
- **Procedures** will be invoked using **CALL** statement.
- A **Procedure** has
  - Name
  - A Parameters List (Optional)
  - SQL Statements

Variables in Procedure

- The concept of User-defined variables and Local variables in procedures are the same as these concepts in functions.
  - **User-defined** Variables
  - **Local** Variables

- The next slides on user-defined and local variables are a review on how to declare and initialize these variables in MySQL.

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 in 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;
```

Syntax of Creating Procedure in MySQL

```
DELIMITER //
CREATE PROCEDURE procedure_name (IN / OUT parameter1 datatype1,
                                 IN / OUT parameter2 datatype2,
                                 ... )
BEGIN
        <Declaration Part of Procedure>
        <Execution Part of Procedure>
END//
DELIMITER ;
```

Procedure

```
DELIMITER //

CREATE PROCEDURE procedure_name

(IN / OUT parameter1 datatype1,
IN / OUT parameter2 datatype2,
... )

BEGIN
        <Declaration Part of Procedure>
        <Execution Part of Procedure>
END//

DELIMITER ;
```

Function

```
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 ;
```

Procedure with NO Parameters

- A procedure without parameters has no input and output parameters.
- It is called by the procedure name, followed by ( ), without any parameters.
- It is used for simple queries.

Procedure with NO Parameters

- Consider the following table.
- Create a **procedure** to display all the books' information.
- Call (Invoke) the procedure.

Book

ISBN	bookName	Price
113	Intro to Python	45
114	Machine Learning	30
115	Computer Networks	75

```
DELIMITER //
CREATE PROCEDURE displaybooks( )
BEGIN
        SELECT * FROM Book;
END//
DELIMITER ;
```

Step 1 – Creating procedure

```
CALL displaybooks( );
```

Step 2 – Calling procedure

Procedure with IN Parameter

- An **IN** parameter is used to take a parameter as input.
- An **IN** parameter passes a value into a procedure when it is invoked.
- The procedure might modify the value,
  - but the modification is not visible to the caller when the procedure returns.

Procedure with IN Parameter

- Consider the following table.
- Create a procedure that **takes ISBN of a book** and updates the price of only that book by adding 10 dollars.
- Call (Invoke) the procedure by passing ISBN=114 to the procedure.

```
DELIMITER //
CREATE PROCEDURE updatePrice(IN ISBN int)
BEGIN
        UPDATE Book SET Price = Price + 10 WHERE Book.ISBN = ISBN;
END//
DELIMITER ;
```

```
CALL updatePrice(114);
```

```
SELECT * FROM Book;
```

Book

ISBN	bookName	Price
113	Intro to Python	45
114	Machine Learning	30
115	Computer Networks	75

Step 1 – Creating procedure

Step 2 – Calling procedure

Procedure with OUT Parameter

- An **OUT** parameter is used to pass a parameter as output.
- An **OUT** parameter passes a value from the procedure back to the caller.
- For each **OUT**, pass a user-defined variable in the **CALL** statement that invokes the procedure so that you can obtain its value when the procedure returns.
- Its initial value is ***NULL*** within the procedure, and
  - its value is visible to the caller when the procedure returns.

Procedure with OUT Parameter

- Consider the following tables.
- Create a procedure to display the average price of all books.
- Call (Invoke) the procedure.

Book

ISBN	bookName	Price
113	Intro to Python	45
114	Machine Learning	30
115	Computer Networks	75

```
DELIMITER //
CREATE PROCEDURE displayAVGPrice(OUT averagePrice decimal(5,2) )
BEGIN
        SELECT avg(price) INTO averagePrice FROM Book;
END//
DELIMITER ;
```

Step 1 – Creating procedure

```
SET @result = 0;
CALL displayAVGPrice(@result);
SELECT @result;
```

Step 2 – Calling procedure

Class Work – Procedure with IN and OUT Parameters

- Create a procedure that **takes ISBN of a book**, and
  - displays "Expensive" if book price is more than $60,
  - displays "Reasonable" if price is between 40 and 60,
  - otherwise displays "Cheap".
- Then call the procedure by passing ISBN=115 to the procedure.

Book

ISBN	bookName	Price
113	Intro to Python	45
114	Machine Learning	30
115	Computer Networks	75

Class Work – Procedure with IN and OUT Parameters

```
DELIMITER //
CREATE PROCEDURE PriceLevel(IN ISBN int , OUT priceCategory varchar(15))
BEGIN
        DECLARE bookPrice int;
        SELECT Price INTO bookPrice FROM Book WHERE Book.ISBN = ISBN;
        SELECT
            CASE
                WHEN bookPrice>60 THEN 'Expensive'
                WHEN bookPrice BETWEEN 40 AND 60 THEN 'Reasonable'
                ELSE 'Cheap'
            END INTO priceCategory;
END//
DELIMITER ;
```

Book

ISBN	bookName	Price	AuthorID
113	Intro to Python	45	1
114	Machine Learning	30	1
115	Computer Networks	75	2

Step 1 – Creating procedure

```
SET @result = 0;
CALL PriceLevel(115 , @result);
SELECT @result;
```

Step 2 – Calling procedure

Procedure with INOUT Parameter

- An **INOUT** parameter is a combination of IN and OUT parameters.
- It means that the calling program may pass the argument,
  - and the procedure can modify the **INOUT** parameter and
  - pass the new value back to the calling program.
- For each **INOUT** parameter, pass a user-defined variable in the **CALL** statement that invokes the procedure so that you can obtain its value when the procedure returns.