MySQL Stored Procedures (Create, List, Alter, & Drop)

April 13, 2021

MYSQL

Home » SysAdmin » MySQL Stored Procedures (Create, List, Alter, & Drop)

Introduction

MySQL stored procedures **group multiple tasks** into one and save the task on the server for future use.

Stored procedures simplify database management and reduce network traffic. For example, issuing a query to the MySQL server processes the query and returns the results. Using stored procedures saves the queries on the server so they can be executed later.

In this tutorial, you will learn to create, list, alter, and drop stored procedures.

phoenixNAP GLOBAL IT SERVICES
MySQL Stored Procedures {Create, List, Alter, & Drop}
SQL
stored procedures

Prerequisites

- MySQL Server and MySQL Workbench installed
- A MySQL user account with root privileges

What Are Stored Procedures in MySQL?

MySQL stored procedures are **pre-compiled SQL statements** stored in a database. They are subroutines containing a name, a parameter list, and SQL statements.

All relational database systems support stored procedures and do not require any additional runtime-environment packages.

How to Use Stored Procedures?

To invoke stored procedures, you can use the **`CALL`** statement or other stored procedures. The first time a stored procedure is invoked, MySQL looks it up in the database catalog, compiles the code, places it in the **cache memory**, and executes it.

Subsequent runs in the same session execute stored procedures from the cache memory, making them extremely useful for repetitive tasks.

Stored procedures make use of **parameters** to pass values and customize results. Parameters are used to specify the columns in a table in which the query operates and returns results.

Stored procedures can also include the **`IF`**, **`CASE`**, and **`LOOP` control flow statements** that procedurally implement the code.

Create Stored Procedure

Create a stored procedure in two ways:

1. **Use MySQL Shell**

Use the following syntax to create a stored procedure in MySQL:

```
DELIMITER //
CREATE PROCEDURE procedure_name ( IN | OUT | INOUT parameter_na
me parameter_datatype (length), … )
BEGIN
    SQL statements
END //
DELIMITER ;
```

By default, the syntax is associated with the database in use, but you can also use the syntax for another database by specifying the database name in the following way: **`database_name.procedure_name`**.

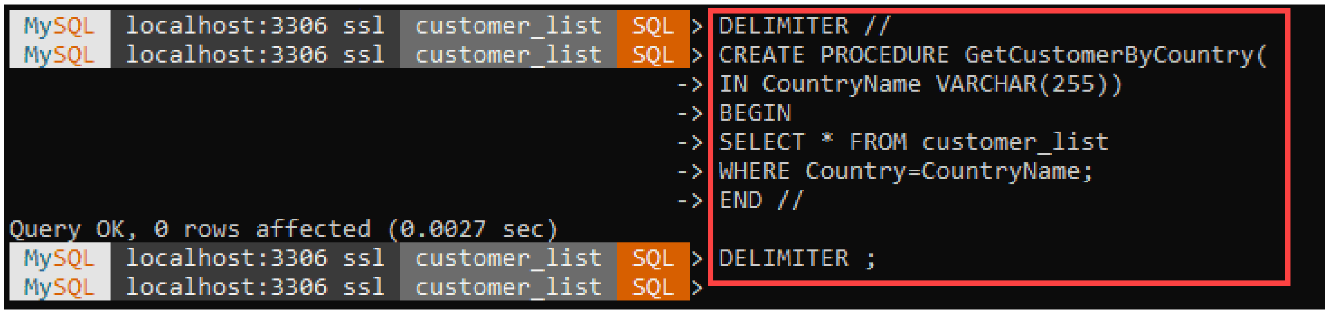
Here, the first **`DELIMITER`** argument sets the default delimiter to **`//`**, while the last **`DELIMITER`** argument sets it back to the semicolon **`;`**. To use multiple statements, specify different delimiters like **`$$`**.

The procedure name comes after the **`CREATE PROCEDURE`** argument. After the procedure name, use parenthesis to specify the parameters to use in the procedure, the name of the parameter, the data type, and data length. Separate each parameter with a comma.

The **parameter modes** are:

- **`IN`** – Use to pass a parameter as input. When it is defined, the query passes an argument to the stored procedure. The value of the parameter is always protected.
- **`OUT`** – Use to pass a parameter as output. You can change the value within the stored procedure, and the new value is passed back to the calling program.
- **`INOUT`** – A combination of **`IN`** and **`OUT`** parameters. The calling program passes the argument, and the procedure can modify the **`INOUT`** parameter, passing the new value back to the program.

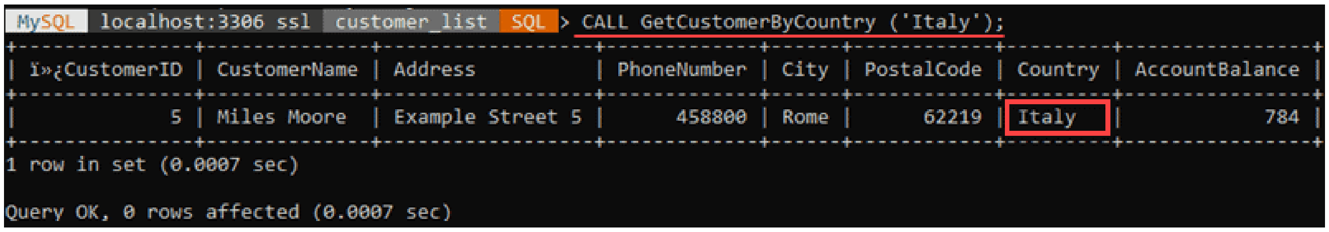
For example:



```
MySQL  localhost:3306 ssl  customer_list  SQL > DELIMITER //
MySQL  localhost:3306 ssl  customer_list  SQL > CREATE PROCEDURE GetCustomerByCountry(
                                              -> IN CountryName VARCHAR(255))
                                              -> BEGIN
                                              -> SELECT * FROM customer_list
                                              -> WHERE Country=CountryName;
                                              -> END //
Query OK, 0 rows affected (0.0027 sec)
MySQL  localhost:3306 ssl  customer_list  SQL > DELIMITER ;
MySQL  localhost:3306 ssl  customer_list  SQL >
```

Execute the stored procedure by calling it:

```
CALL procedure_name;
```



```
MySQL  localhost:3306 ssl  customer_list  SQL > CALL GetCustomerByCountry ('Italy');
+---------------+--------------+------------------+-------------+------+------------+---------+----------------+
| ï»¿CustomerID | CustomerName | Address          | PhoneNumber | City | PostalCode | Country | AccountBalance |
+---------------+--------------+------------------+-------------+------+------------+---------+----------------+
|             5 | Miles Moore  | Example Street 5 |      458800 | Rome |      62219 | Italy   |            784 |
+---------------+--------------+------------------+-------------+------+------------+---------+----------------+
1 row in set (0.0007 sec)

Query OK, 0 rows affected (0.0007 sec)
```

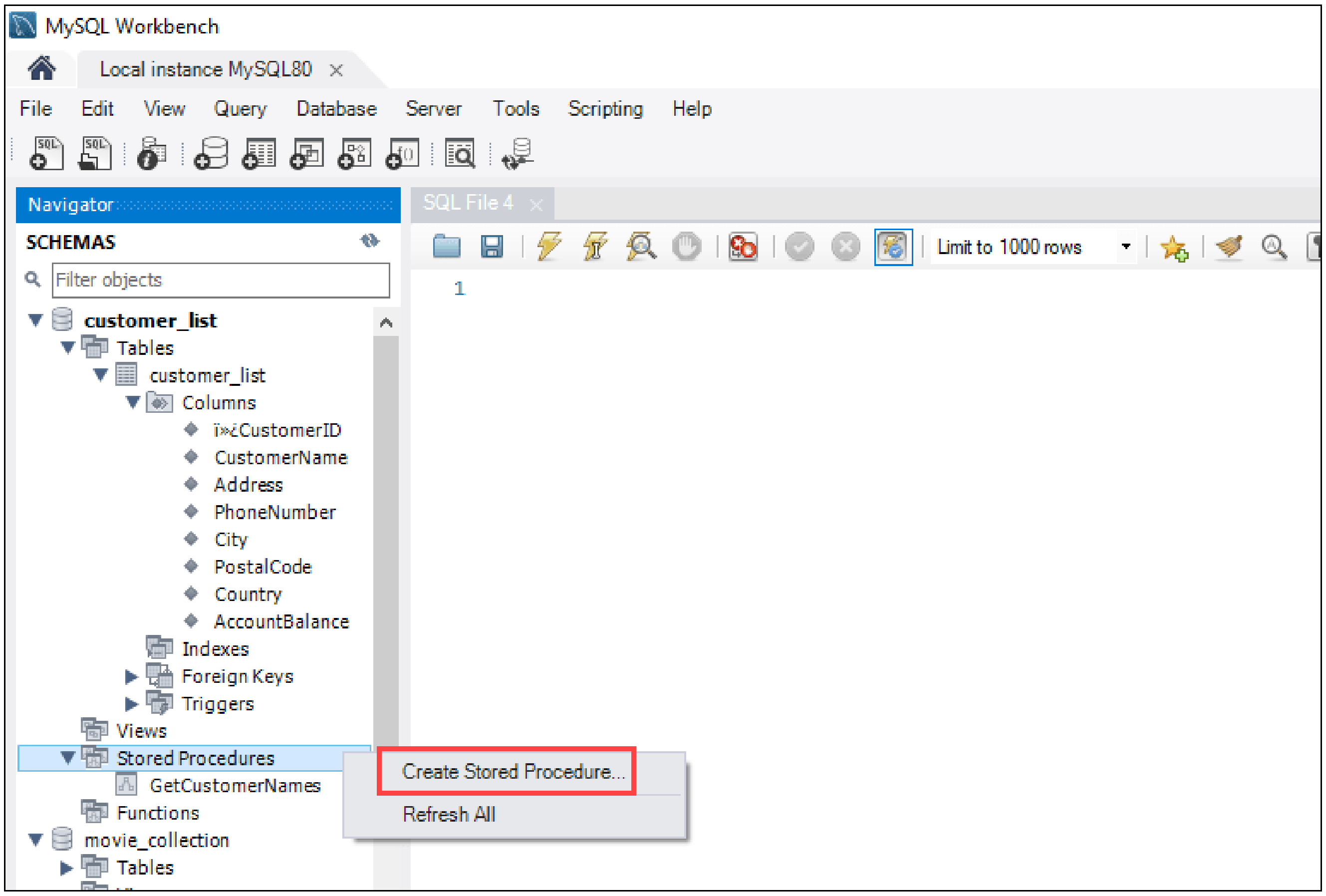
The query returns results for the stored procedure.

2. **Use MySQL Workbench**

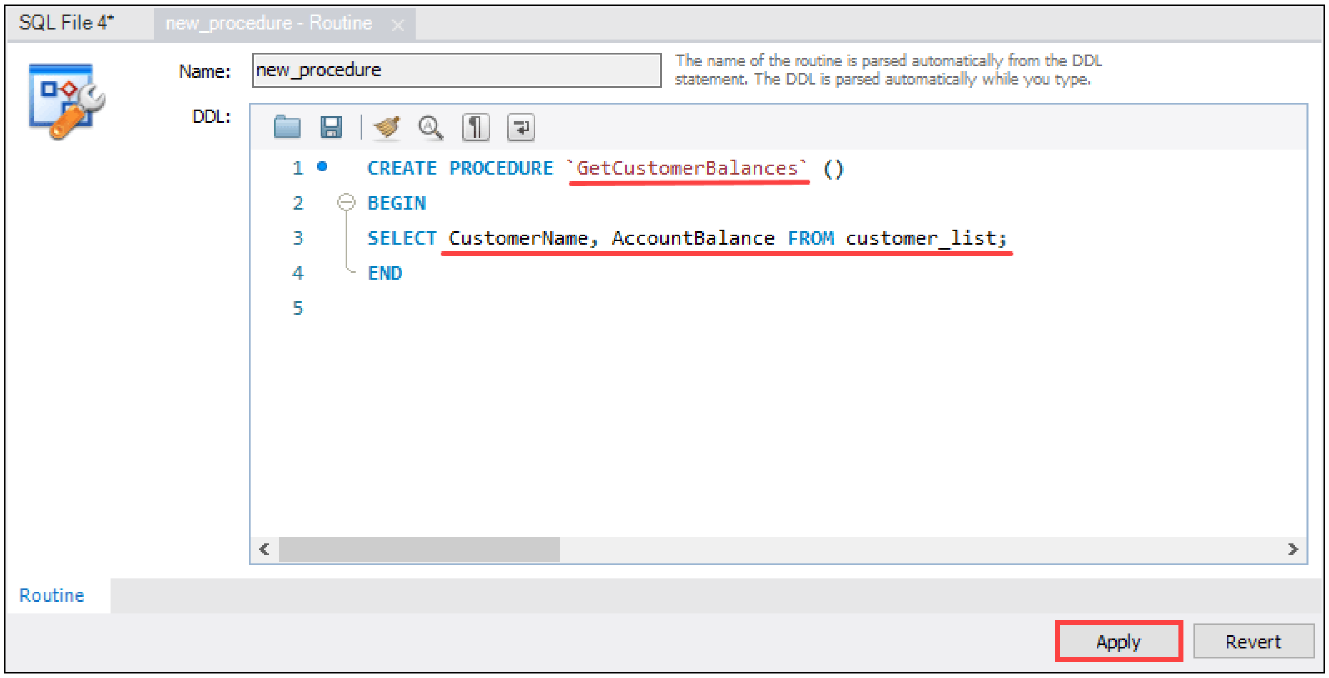
Another way to create a stored procedure is to use the MySQL Workbench Wizard. The wizard is intuitive and simplifies the process since you do not have to place delimiters or worry about the format.

Follow these steps:

Step 1: Right-click **Stored Procedures** in the Navigator window of MySQL Workbench and choose **Create Stored Procedure…** to start the wizard.



Step 2: Specify the procedure name and enter the code within the **BEGIN** … **END** block.



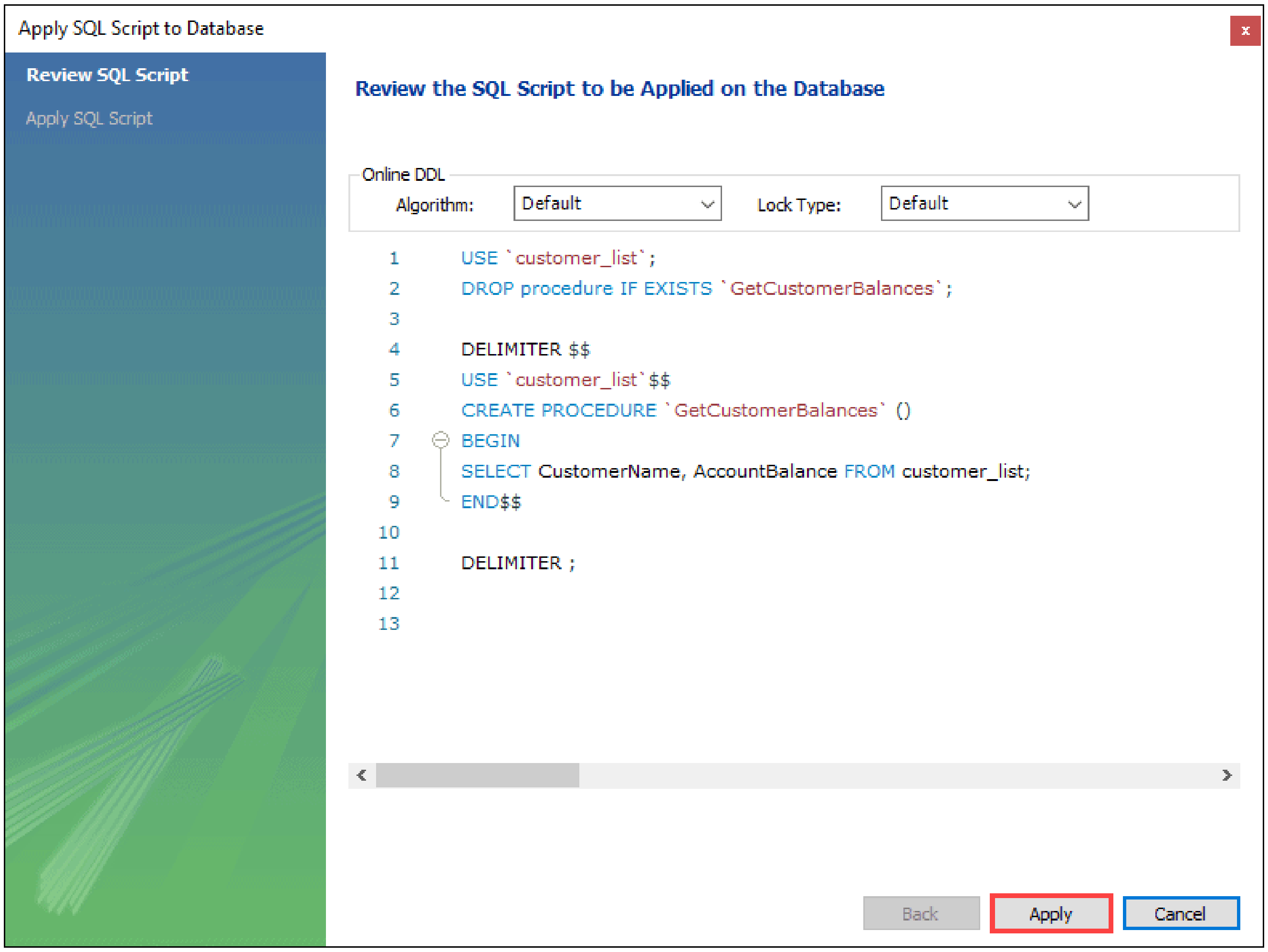
Name: new_procedure

The name of the routine is parsed automatically from the DDL statement. The DDL is parsed automatically while you type.

DDL:

```
CREATE PROCEDURE `GetCustomerBalances` ()
BEGIN
SELECT CustomerName, AccountBalance FROM customer_list;
END
```

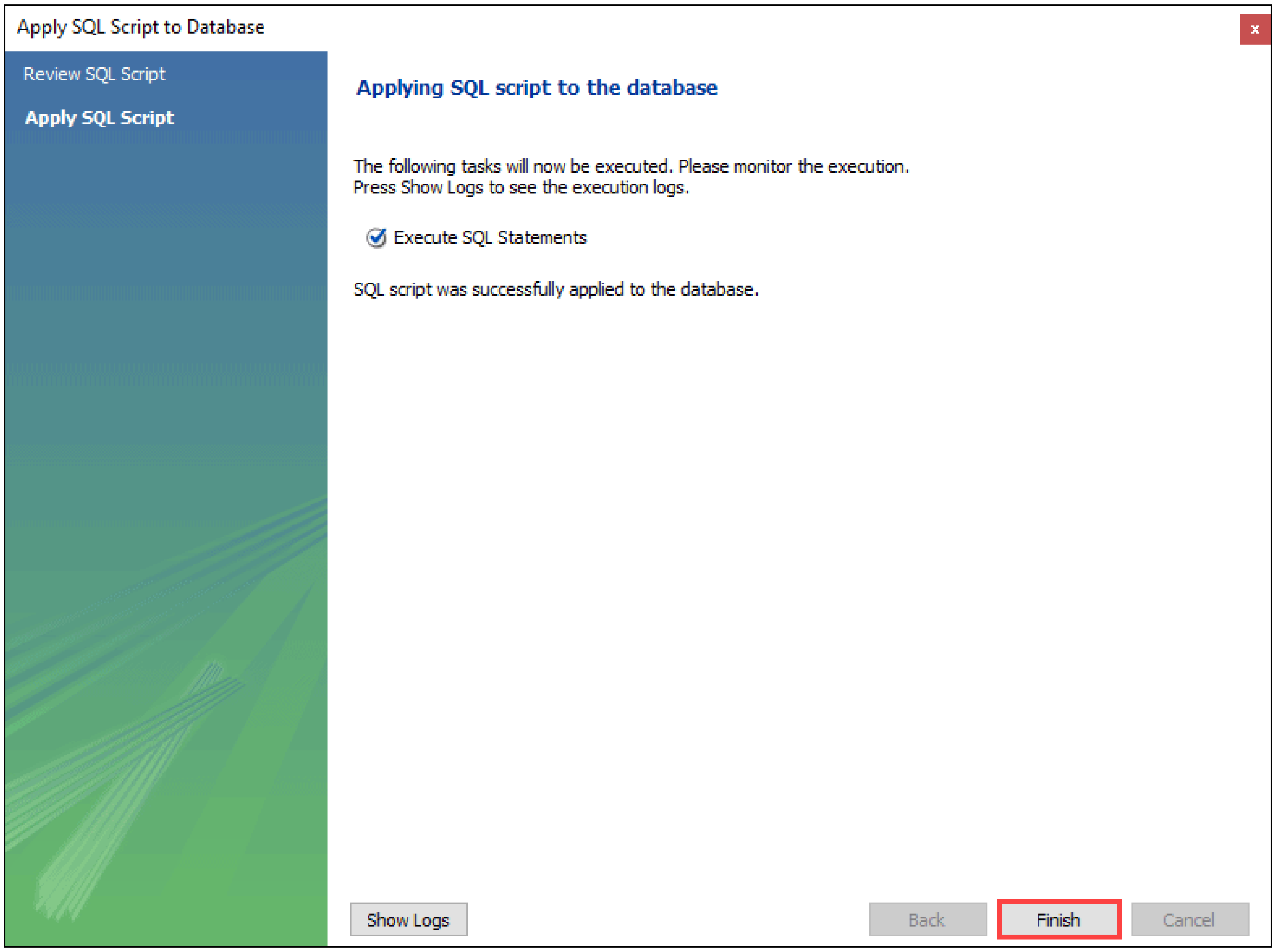
Step 3: Review the code and click **Apply**.


Apply SQL Script to Database
Review SQL Script
Apply SQL Script
Review the SQL Script to be Applied on the Database
Back
Apply
Cancel

Online DDL
Algorithm: Default
Lock Type: Default

```
USE `customer_list`;
DROP procedure IF EXISTS `GetCustomerBalances`;

DELIMITER $$
USE `customer_list`$$
CREATE PROCEDURE `GetCustomerBalances` ()
BEGIN
SELECT CustomerName, AccountBalance FROM customer_list;
END$$

DELIMITER ;
```

Step 4: Confirm execution by clicking **Apply** and create the procedure by clicking **Finish**.



Applying SQL script to the database

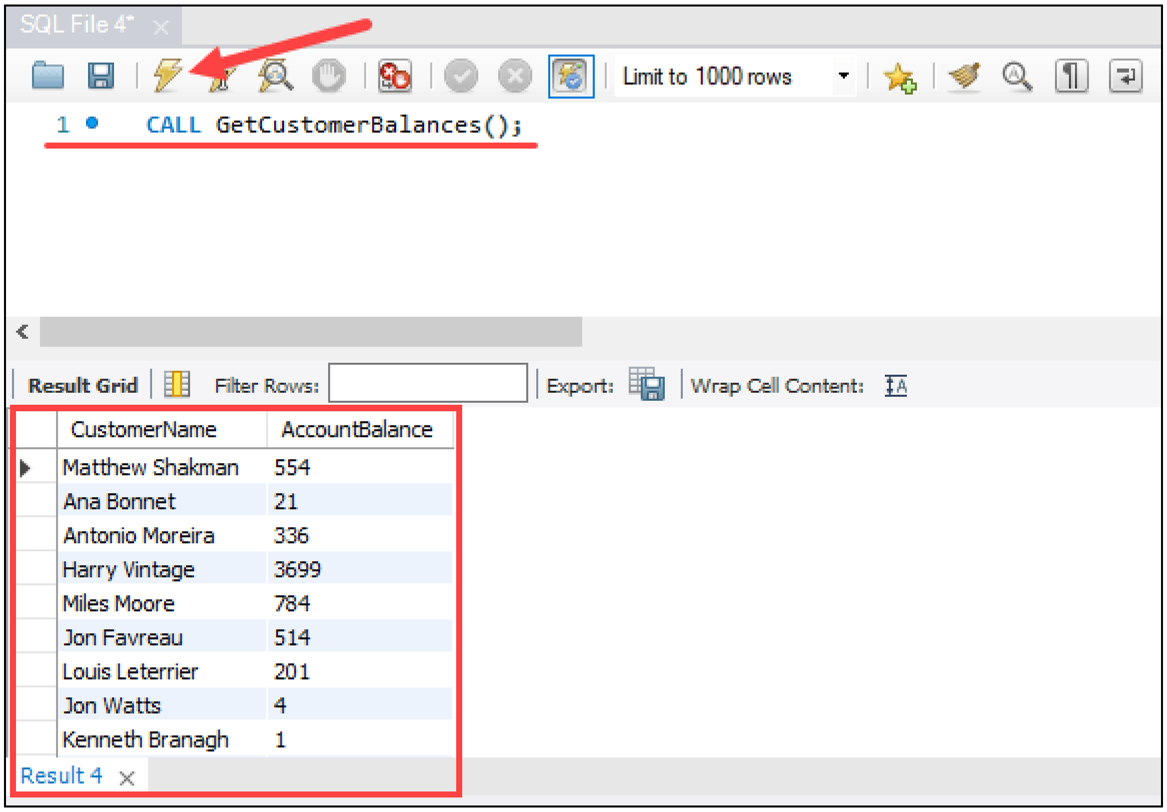
The following tasks will now be executed. Please monitor the execution.
Press Show Logs to see the execution logs.

✓ Execute SQL Statements

SQL script was successfully applied to the database.

Step 5: Execute the procedure to see if it works. Create a new SQL tab for executing queries.

Step 6: `CALL` the procedure in the SQL tab and click **Execute**.



```
CALL GetCustomerBalances();
```

CustomerName	AccountBalance
Matthew Shakman	554
Ana Bonnet	21
Antonio Moreira	336
Harry Vintage	3699
Miles Moore	784
Jon Favreau	514
Louis Leterrier	201
Jon Watts	4
Kenneth Branagh	1

If no errors return, MySQL executes the stored procedure and displays the results.

Note: Refer to our MySQL guide on tables to learn how to create a table in MySQL.

List Stored Procedures

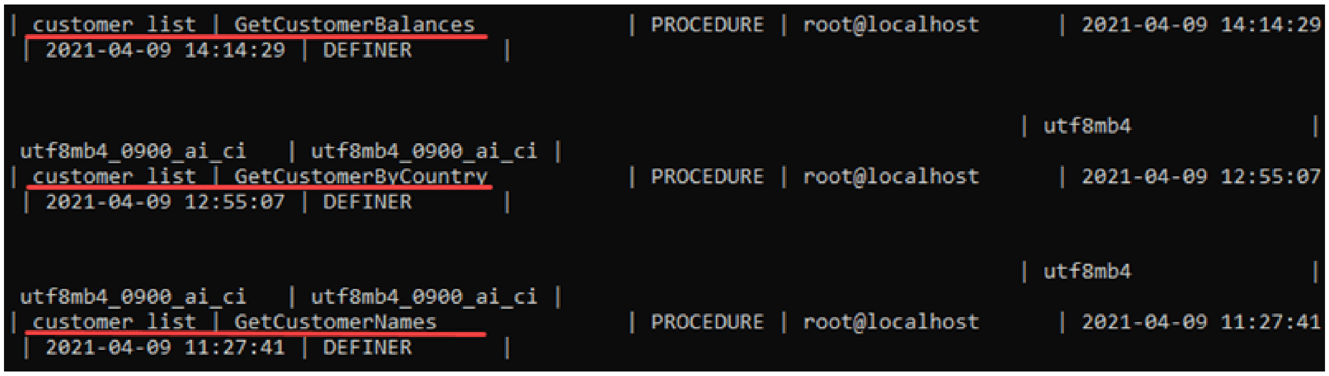
There are three ways to see a list of all stored procedures:

1. **Use MySQL Shell**

To get a list of all stored procedures you have access to, including their characteristics, use the following syntax:

```
SHOW PROCEDURE STATUS [LIKE 'pattern' | WHERE search_condition]
```

The **`SHOW PROCEDURE STATUS`** statement returns a lengthy output. The statement displays the names and characteristics of stored procedures that you have access to on the server.

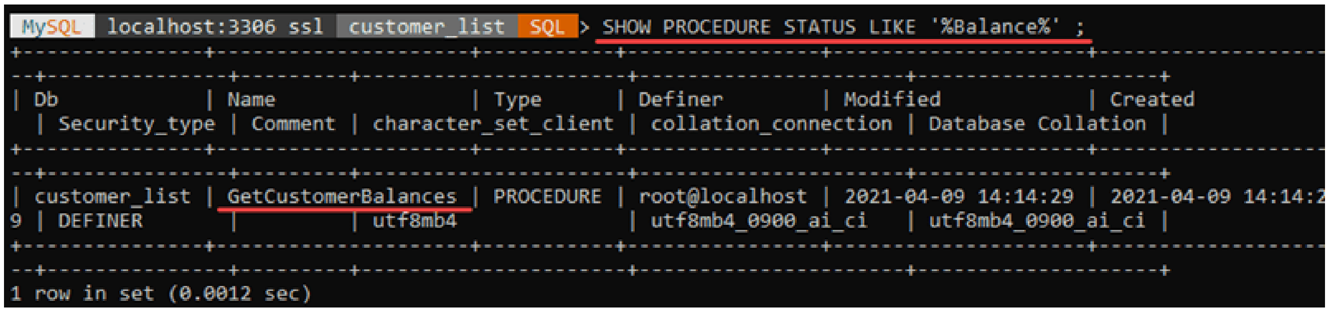


```
| customer_list | GetCustomerBalances          | PROCEDURE | root@localhost      | 2021-04-09 14:14:29
 | 2021-04-09 14:14:29 | DEFINER         |


                                                                                | utf8mb4                |
utf8mb4_0900_ai_ci    | utf8mb4_0900_ai_ci |
| customer_list | GetCustomerByCountry         | PROCEDURE | root@localhost      | 2021-04-09 12:55:07
 | 2021-04-09 12:55:07 | DEFINER         |


                                                                                | utf8mb4                |
utf8mb4_0900_ai_ci    | utf8mb4_0900_ai_ci |
| customer_list | GetCustomerNames             | PROCEDURE | root@localhost      | 2021-04-09 11:27:41
 | 2021-04-09 11:27:41 | DEFINER         |
```

Scroll through the output to find the procedures currently on the server.

The **`LIKE`** argument finds stored procedures containing a specific word in their name. Use **`%`** to replace any number of characters, including zero.

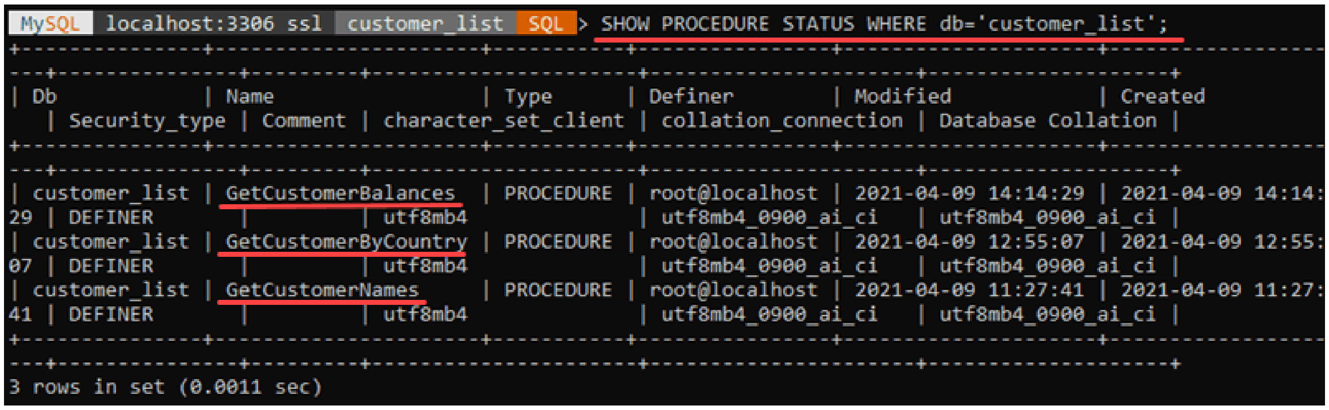
For example:



```
MySQL  localhost:3306 ssl  customer_list  SQL > SHOW PROCEDURE STATUS LIKE '%Balance%' ;
+---------------+---------------------+-----------+----------------+---------------------+---------------------
--+---------------+---------+----------------------+----------------------+--------------------+
| Db            | Name                | Type      | Definer        | Modified            | Created
  | Security_type | Comment | character_set_client | collation_connection | Database Collation |
+---------------+---------------------+-----------+----------------+---------------------+---------------------
--+---------------+---------+----------------------+----------------------+--------------------+
| customer_list | GetCustomerBalances | PROCEDURE | root@localhost | 2021-04-09 14:14:29 | 2021-04-09 14:14:2
9 | DEFINER       |         | utf8mb4              | utf8mb4_0900_ai_ci   | utf8mb4_0900_ai_ci |
+---------------+---------------------+-----------+----------------+---------------------+---------------------
--+---------------+---------+----------------------+----------------------+--------------------+
1 row in set (0.0012 sec)
```

The **`WHERE`** argument allows you to list stored procedures only in a particular database.

For example:



```
MySQL  localhost:3306 ssl  customer_list  SQL > SHOW PROCEDURE STATUS WHERE db='customer_list';
+---------------+----------------------+-----------+----------------+---------------------+--------------------
---+---------------+---------+----------------------+----------------------+--------------------+
| Db            | Name                 | Type      | Definer        | Modified            | Created
   | Security_type | Comment | character_set_client | collation_connection | Database Collation |
+---------------+----------------------+-----------+----------------+---------------------+--------------------
---+---------------+---------+----------------------+----------------------+--------------------+
| customer_list | GetCustomerBalances  | PROCEDURE | root@localhost | 2021-04-09 14:14:29 | 2021-04-09 14:14:
29 | DEFINER       |         | utf8mb4              | utf8mb4_0900_ai_ci   | utf8mb4_0900_ai_ci |
| customer_list | GetCustomerByCountry | PROCEDURE | root@localhost | 2021-04-09 12:55:07 | 2021-04-09 12:55:
07 | DEFINER       |         | utf8mb4              | utf8mb4_0900_ai_ci   | utf8mb4_0900_ai_ci |
| customer_list | GetCustomerNames     | PROCEDURE | root@localhost | 2021-04-09 11:27:41 | 2021-04-09 11:27:
41 | DEFINER       |         | utf8mb4              | utf8mb4_0900_ai_ci   | utf8mb4_0900_ai_ci |
+---------------+----------------------+-----------+----------------+---------------------+--------------------
---+---------------+---------+----------------------+----------------------+--------------------+
3 rows in set (0.0011 sec)
```

In this example, the statement returns only the stored procedures for the '*customer_list*' database.

2. **Use Data Dictionary**

The *information_schema* database contains a table called **routines**, which has information on stored procedures and functions related to all databases on the current MySQL server.

Use the following syntax to see all stored procedures for a database:

```
SELECT
    routine_name
FROM
    information_schema.routines
WHERE
    routine_type = 'PROCEDURE'
        AND routine_schema = 'database_name';
```

3. **Use MySQL Workbench**

For a GUI approach to viewing stored procedures, use MySQL Workbench. Follow these steps to see stored procedures:

Step 1: Double-click the database you want to use in the **Navigator** section.

Step 2: Expand the **Stored Procedures** drop-down item.

This item shows all stored procedures for the current database.

Alter Stored Procedure

To alter a stored procedure means to **change the characteristics** of a procedure. There is **no statement** in MySQL for **modifying the parameters or the body** of a stored procedure. To change parameters or the body, drop the stored procedure and create a new one.

Alter a stored procedure in two ways:

1. **Use MySQL Shell**

Change a procedure characteristic by using the **`ALTER PROCEDURE`** statement. For example, we can add a comment to a procedure we created previously. The syntax is:

```
ALTER PROCEDURE procedure_name
COMMENT 'Insert comment here';
```

2. **Use MySQL Workbench**

MySQL Workbench GUI allows users to alter a stored procedure where users can **add parameters or change the code.** MySQL Workbench **drops** the existing stored procedure and **creates** a new one after the changes have been made.

Follow these steps:

Step 1: In the Navigator section, right-click the stored procedure you want to modify. Select the **Alter Stored procedure…** item.

Step 2: When the tab opens, make the desired changes to the existing stored procedure and click **Apply**.

Step 3: An **SQL Script review window** appears showing the process – dropping the existing stored procedure and creating a new one containing the changes.

Click **Apply** and then **Finish** in the next window to execute the script.

Drop Stored Procedure

To drop (delete) a procedure:

1. **Use MySQL Shell**

Delete a stored procedure from the server by using the **`DROP PROCEDURE`** statement.

The basic syntax is:

```
DROP PROCEDURE [IF EXISTS] stored_procedure_name;
```

The **`IF EXISTS`** parameter drops the stored procedure only if it exists on the server. Enter the name of the stored procedure in place of the **`stored_procedure_name`** syntax.

For example:

Since there is no procedure named '*test*' on the server, the output states that 0 rows were affected and that the specified procedure does not exist.

Dropping a non-existing procedure without the **`IF EXISTS`** parameter returns an error.

2. **Use MySQL Workbench**

To drop a stored procedure with MySQL Workbench, follow these steps:

Step 1: Expand the Stored Procedures item in the Navigator section. Right-click the stored procedure you want to delete and choose **Drop Stored Procedure...** in the context menu.

Step 2: In the confirmation window, click **Drop Now** to delete the stored procedure.

This action **permanently deletes** the procedure.

Note: Check out our tutorial to learn how to drop a table in MySQL.

MySQL Stored Procedures Advantages and Disadvantages

Stored procedures have several advantages and disadvantages as they tailor to specific needs. Below are some of the advantages and disadvantages.

Advantages of Using Stored Procedures

The advantages of stored procedures are:

Network Traffic Reduction

Stored procedures help reduce the network traffic between applications and MySQL Server by keeping all the programming logic on the server. Instead of sending multiple query results across the network, apps send only the procedure name and the parameter input.

Improved Security

The database administrator grants apps privileges to call and access only specific stored procedures without giving them direct access to tables. Stored procedures help prevent script injection attacks since input parameters are treated as values and not as executable code.

Centralized Business Logic

Stored procedures encapsulate the business logic reusable by multiple applications. That helps reduce duplicating that same logic in many different applications and makes the database more consistent.

Bosko Marijan

Having worked as an educator and content writer, combined with his lifelong passion for all things high-tech, Bosko strives to simplify intricate concepts and make them user-friendly. That has led him to technical writing at PhoenixNAP, where he continues his mission of spreading knowledge.

Disadvantages of Using Stored Procedures

The disadvantages of stored procedures are:

Resource Usage

Next you should read

Using many stored procedures and logical operations causes the memory and CPU usage to increase significantly for every connection.

Backup and Recovery, MySQL

N

I procedures written in a specific language from one installation to a d procedure also ties the user to a particular database.

T ng

M ilities for testing and debugging stored procedures, so it can be
c veloping and maintaining stored procedures require extensive
k nge for new developers and results in added maintenance costs.

(

A u know what stored procedures are and when to use them. You
a odify, see all available stored procedures, and delete the ones you
r

Yes No

How To Use MySQL Triggers

April 12, 2021

MySQL triggers provide control over data validation when inserting, updating or deleting data from a

MySQL, SysAdmin

MySQL Data Types

March 4, 2021

Data types are important to understand as a name and a data type define each column in a database table....

READ MORE

MySQL, SysAdmin

MySQL Commands Cheat Sheet

January 20, 2021

Need a reference sheet for all the important MySQL commands? Check out this MySQL Commands article which...

READ MORE

MySQL

How to Create a Table in MySQL

November 3, 2020

MySQL is a well-known, free and open-source database application. One of the most crucial processes in MySQL...

READ MORE

Live Chat Get a Quote Support | 1-855-330-1509 Sales | 1-877-588-5918

Contact Us
Legal
Privacy Policy
Terms of Use
DMCA
GDPR
Sitemap