## MySQL Programming Lab

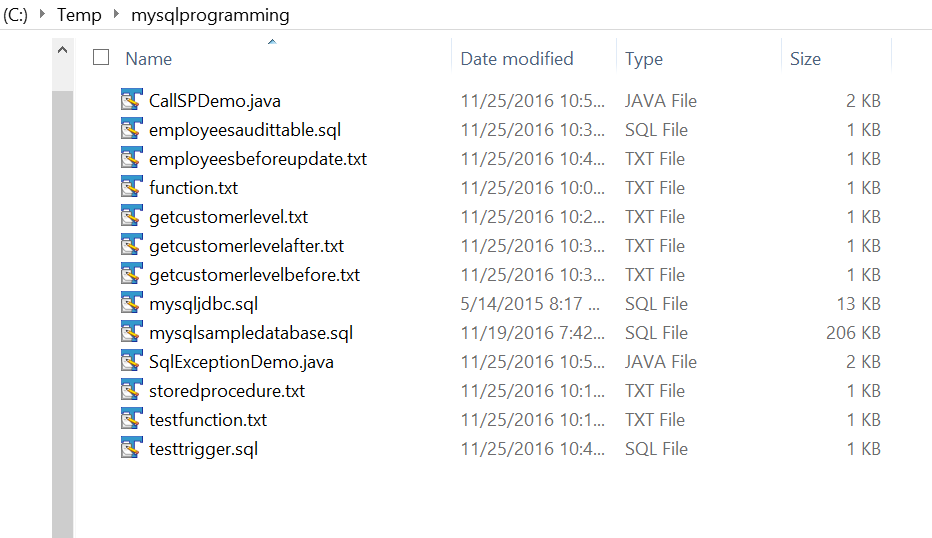
## (taken from several tutorials on the mysqltutorial.org website)

## Loading the Test Database

Step 1. Download the [mysqlprogramming](http://web.csulb.edu/~dbrown/CECS323/Labs/mysqlprogramming.zip).zip from my code page. It contains all the files needed for this lab.

|  |  |
| --- | --- |
| mysqlsampledatabase.sql | Classic Models Database used for the stored procedures, functions and triggers sections |
| storedprocedure.txt | Code for GetAllProducts stored procedure |
| function.txt | Code for CustomerLevel() function |
| testfunction.txt | Code to test the CustomerLevel() function |
| getcustomerlevelbefore.txt | Code for GetCustomerLevel stored procedure before the CustomerLevel() function |
| getcustomerlevelafter.txt | Code for GetCustomerLevel stored procedure after the function |
| employeesaudittable.sql | Code to create the employees\_audit table |
| employeesbeforeupdate.txt | Code to create the employees\_before\_update trigger |
| testtrigger.sql | Code to test the trigger |

Step 2. Unzip the downloaded file into a temporary folder. You can use any folder you want. To make it simple we will unzip it to the C:\temp folder as follows.



Step 3. Launch MySQL Workbench application.

Step 4. Either connect to your local database using an existing connection or create a new one.

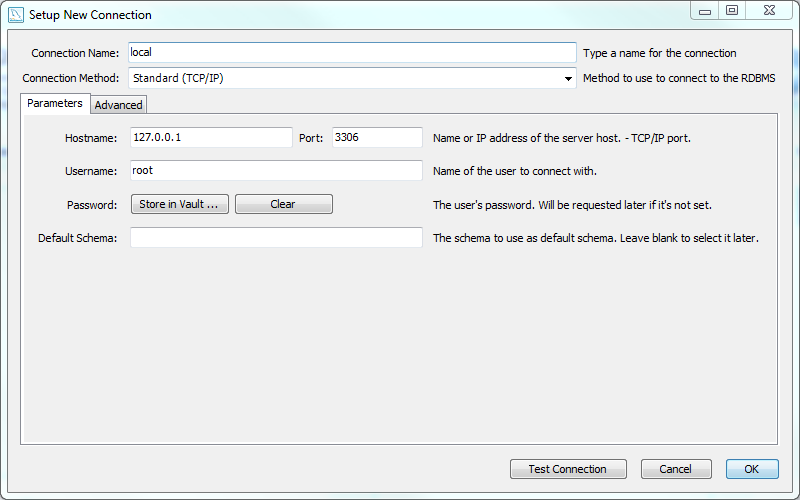
To add a new database connection for querying, click New Connection as follows:

Step 5. Setup a new connection: you must enter all connection parameters on this Setup New Connection window. The following information is required:

* Connection name: the name of the connection. If you connect to the localhost , just type local . In case you connect to a specific host, use the host name for the name of the connection to make it clear.
* Host name: in this case it is 127.0.0.1 i.e., localhost . You can enter either IP address or the name of the database server.
* Username: the user that you use connect to the MySQL database. In this case it is the root user.

You can also provide the following information optionally:

* Password: the password of the user that you use to connect to the database.
* Default Schema: is the database that you want to connect. You can leave it blank and select later using the use database command.

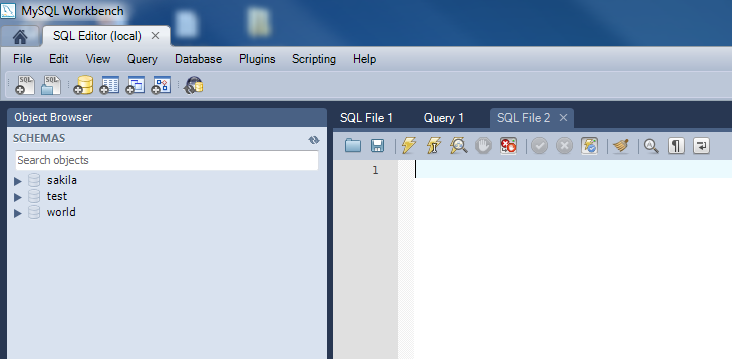


You should click on Test Connection to make sure that the parameters you provided are correct, and then click OK button to create a new connection. Once you complete, you will see connections window as follows:

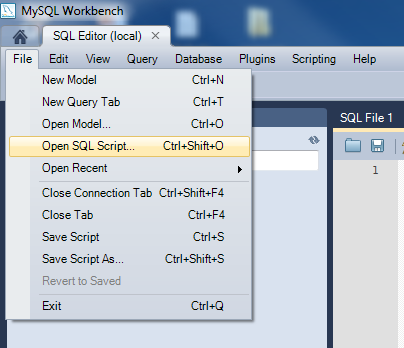
Step 6. Click the local database connection to connect to MySQL database server. Because we didn’t provide the password in the previous step, MySQL asks us to enter the password for the root account. We enter our password and click OK button.



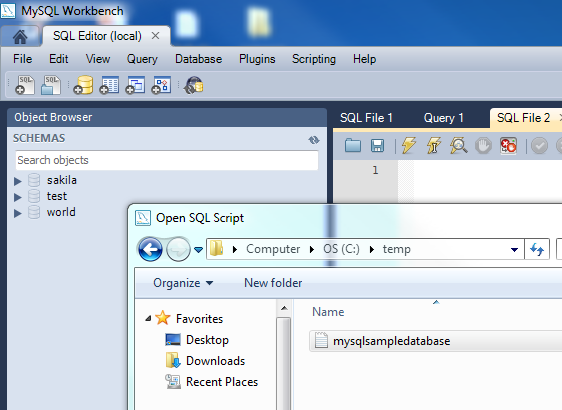
If you enter both user and password correctly, you will see the following window:



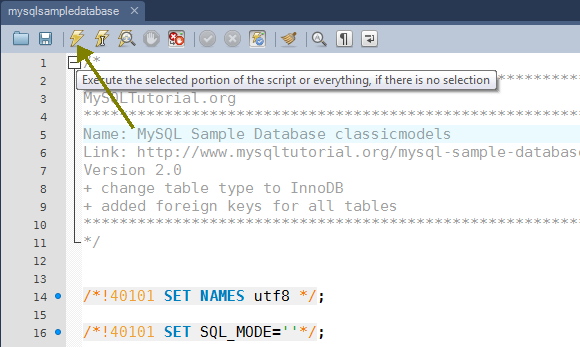
Step 7. Open SQL Script by choosing File > Open SQL Script  or press the Ctrl+Shift+O keyboard shortcut.



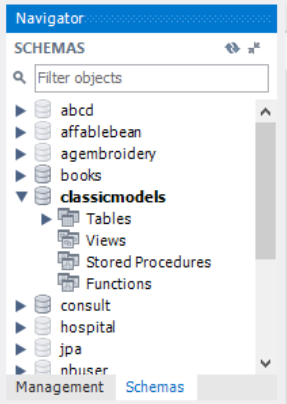
Step 8. Choose SQL Script File by selecting the file **C:\temp\mysqlprogramming\mysqlsampledatabase.sql**



Step 10. To execute SQL Script, you click execute button from the toolbar as following:



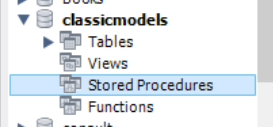
Step 11. Right click inside the Schemas panel and click Refresh All button to update the panel. The **classicmodels** database is loaded successfully into MySQL database server.



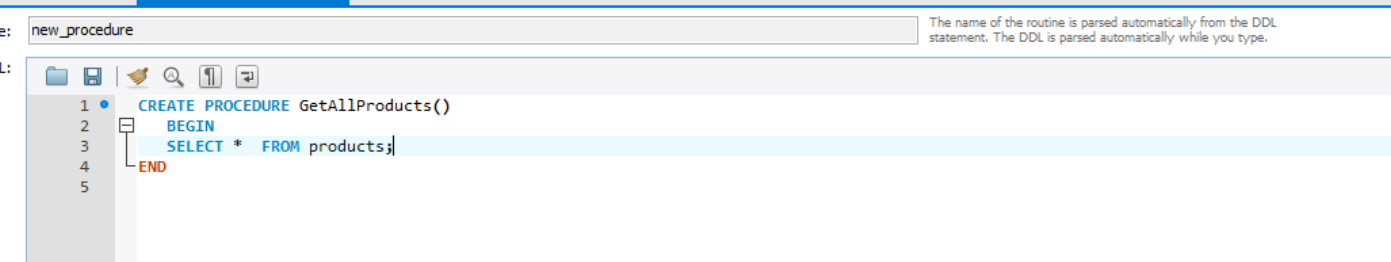
## MySQL Stored Procedure Example

For example, in MySQL Workbench, you can create a new stored procedure as follows:

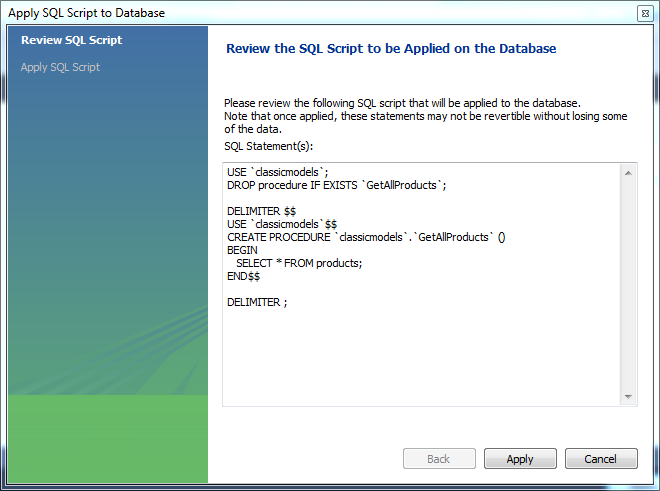
First, right mouse click on the Stored Procedures and choose “Create Procedure…” menu item.



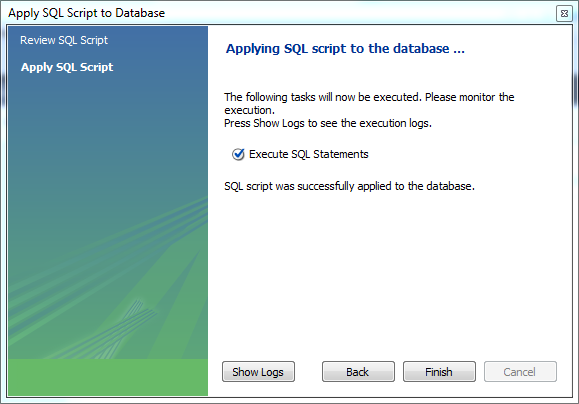
Next, enter the stored procedure code and click the Apply button (**storedprocedure.txt**)



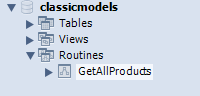
Then, you can review the code before MySQL stores it in the database. Click Apply button if everything is good.



After that, MySQL compiles and puts the stored procedure in the database catalog; click the Finish button.



Finally, you can see a new stored procedure created under *Stored Procedures* of the **classicmodels** database.



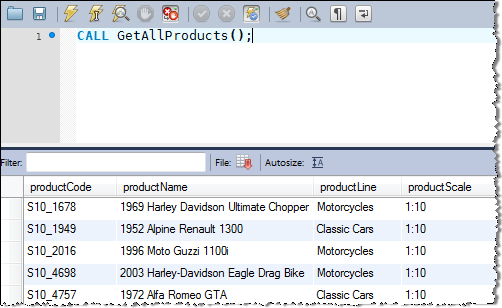
We have created a new stored procedure. Now, it’s time to learn how to use it.

In order to call a stored procedure, you use the following SQL command:

|  |  |
| --- | --- |
|  | **CALL STORED\_PROCEDURE\_NAME();** |

You use the CALL statement to call a stored procedure e.g., to call the GetAllProducts()  stored procedure, you use the following statement:

|  |  |
| --- | --- |
|  | **CALL GetAllProducts();** |

If you execute the statement above, you will get all products in the products table.  
[](http://www.mysqltutorial.org/wp-content/uploads/2009/12/mysql-stored-procedure.png)

## MySQL Stored Function Example

Let’s take a look at an example of using stored function. We will use the customers table in the classics database for the demonstration.

For example, in MySQL Workbench, you can create a new function as follows:

First, right mouse click on the Functions and choose “Create Functions…” menu item.

The following example is a function that returns the level of a customer based on credit limit. We use the IF statement to decide the credit limit. (**function.txt)**

CREATE FUNCTION CustomerLevel(p\_creditLimit double) RETURNS VARCHAR(10) DETERMINISTIC

BEGIN

DECLARE lvl varchar(10);

IF p\_creditLimit > 50000 THEN

SET lvl = 'PLATINUM';

ELSEIF (p\_creditLimit <= 50000 AND p\_creditLimit >= 10000) THEN

SET lvl = 'GOLD';

ELSEIF p\_creditLimit < 10000 THEN

SET lvl = 'SILVER';

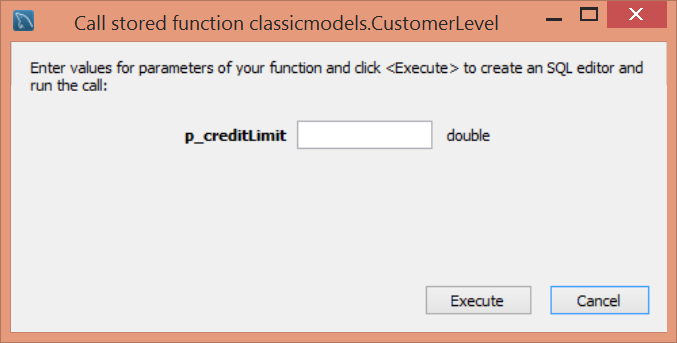
END IF;

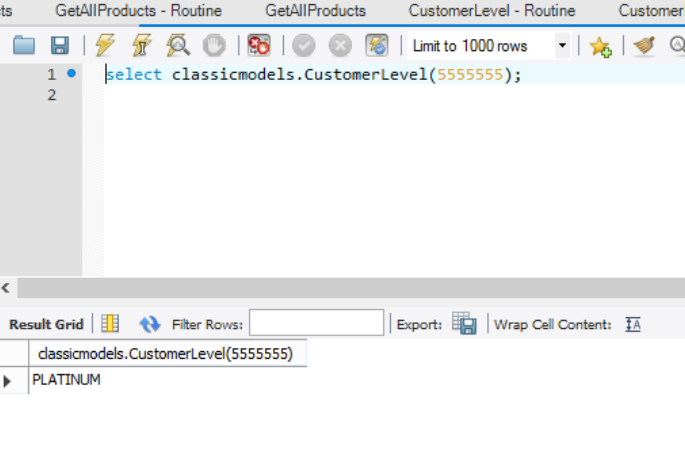
RETURN (lvl);

END

We can test the function as follows:

Click on function then hit the run icon. A dialog box will pop up requesting you to input a credit limit. Then hit Execute.





Now, we can call the CustomerLevel() in an SQL SELECT statement as follows: (**testfunction.txt**)

SELECT

    customerName, CustomerLevel(creditLimit)

FROM

    customers

ORDER BY customerName;



We also rewrite the  GetCustomerLevel() stored procedure that was originally developed using a MySQL IF statement as follows:

|  |  |
| --- | --- |
|  | AFTER  CREATE PROCEDURE GetCustomerLevel(      IN  p\_customerNumber INT(11),      OUT p\_customerLevel  varchar(10)  )  BEGIN      DECLARE creditlim DOUBLE;        SELECT creditlimit INTO creditlim      FROM customers      WHERE customerNumber = p\_customerNumber;        SELECT CUSTOMERLEVEL(creditlim)      INTO p\_customerLevel;    END |

**BEFORE**

CREATE PROCEDURE GetCustomerLevel(

    in  p\_customerNumber int(11),

    out p\_customerLevel  varchar(10))

BEGIN

    DECLARE creditlim double;

    SELECT creditlimit INTO creditlim

    FROM customers

    WHERE customerNumber = p\_customerNumber;

    IF creditlim > 50000 THEN

SET p\_customerLevel = 'PLATINUM';

    ELSEIF (creditlim <= 50000 AND creditlim >= 10000) THEN

        SET p\_customerLevel = 'GOLD';

    ELSEIF creditlim < 10000 THEN

        SET p\_customerLevel = 'SILVER';

    END IF;

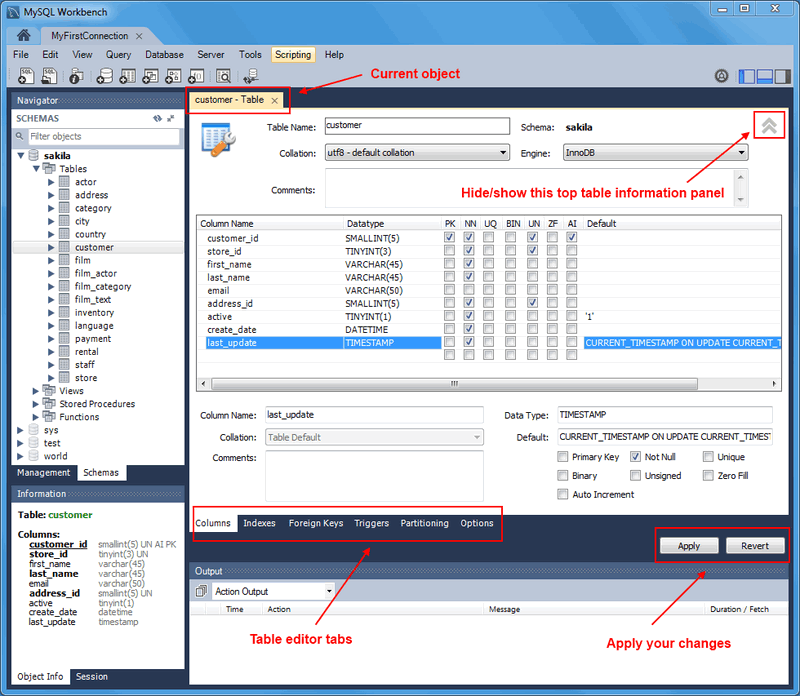
As you can see, the  GetCustomerLevel() stored procedure is much more readable when using the  CustomerLevel() stored function.

Notice that a stored function returns a single value only. If you include a SELECT statement without the INTO clause, you will get an error.

In addition, if a stored function contains SQL statements, you should not use it inside other SQL statements; otherwise, the stored function will slow down the speed of the query.

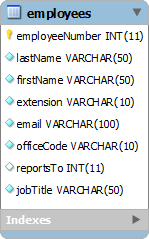
## MySQL trigger example

To create a trigger for an existing table, access the MySQL Table Editor, right-click on a table name in the Object Viewer (the left navigator panel that lists all schemas) and choose ALTER TABLE. This opens a new tab within the main SQL Editor window.



The Triggers tab opens a textbox to create or edit existing triggers. Click the Add Trigger Button.

Let’s start creating a trigger in MySQL to log the changes of the employees table.

[](http://www.mysqltutorial.org/wp-content/uploads/2009/12/employees-table.png)

First, create a new table named employees\_audit to keep the changes of the employee table. The following statement creates the employee\_audit table. (**employeesaudittable.sql**)

CREATE TABLE employees\_audit (

id INT AUTO\_INCREMENT PRIMARY KEY,

employeeNumber INT NOT NULL,

lastname VARCHAR(50) NOT NULL,

changedat DATETIME DEFAULT NULL,

action VARCHAR(50) DEFAULT NULL

);

Next, create a BEFORE UPDATE trigger that is invoked before a change is made to the employees table.

Click the [+] to the right of the BEFORE UPDATE event. An editor window will open with the initial part of the trigger already created. Enter the following block of code to complete the trigger then hit Apply. (**employeesbeforeupdate.txt**)

BEGIN

    INSERT INTO employees\_audit

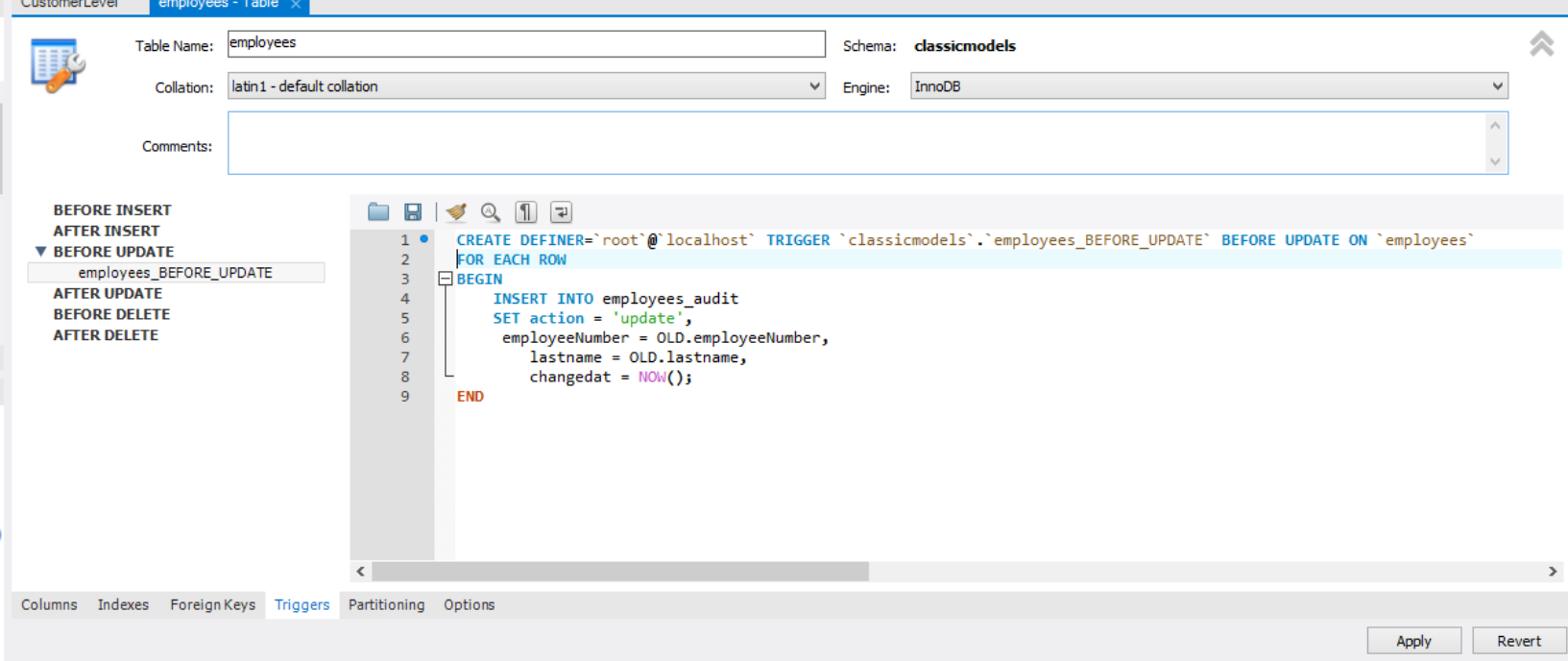
    SET action = 'update',

     employeeNumber = OLD.employeeNumber,

        lastname = OLD.lastname,

        changedat = NOW();

END

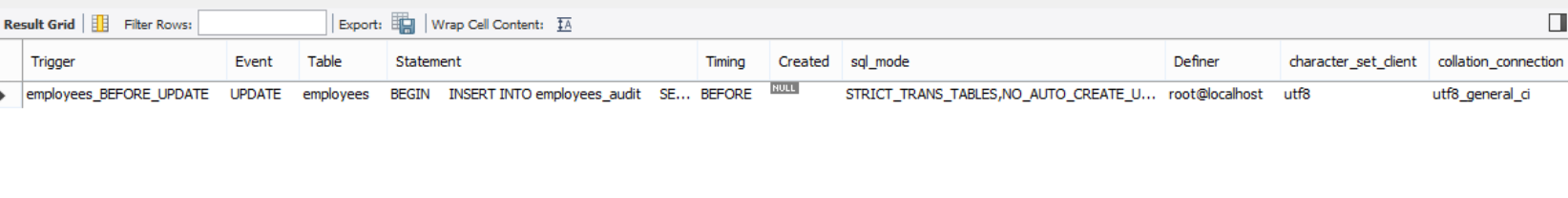


Inside the body of the trigger, we used the OLD keyword to access employeeNumber and lastname column of the row affected by the trigger.

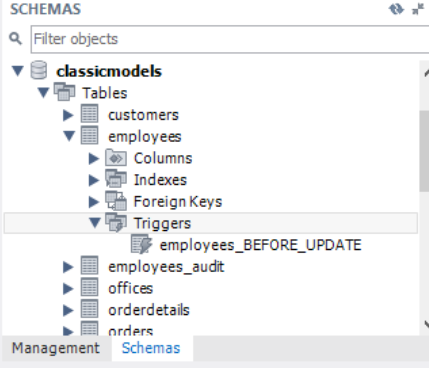
Notice that in a trigger defined for INSERT, you can use NEW keyword only. You cannot use the OLD keyword. However, in the trigger defined for DELETE, there is no new row so you can use the OLD keyword only. In the UPDATE trigger, OLD refers to the row before it is updated and NEW refers to the row after it is updated.

Then, to view all triggers in the current database, you use SHOW TRIGGERS statement as follows:

|  |  |
| --- | --- |
|  | SHOW TRIGGERS; |



In addition, if you look at the schema using MySQL Workbench under the employees > triggers, you will see the employees\_BEFORE\_UPDATE trigger as shown in the screenshot below:



After that, update the employees table to check whether the trigger is invoked. (**testtrigger.sql**)

UPDATE employees

SET

lastName = 'Phan'

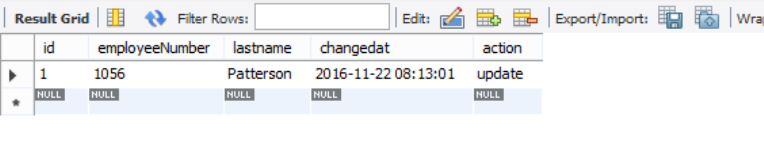
WHERE

employeeNumber = 1056;

Finally, to check if the trigger was invoked by the UPDATE statement, you can query the employees\_audit table using the following query:

SELECT \* FROM employees\_audit;

The following is the output of the query:



As you see, the trigger was really invoked and it inserted a new row into the employees\_audit table.