# MySQL Triggers

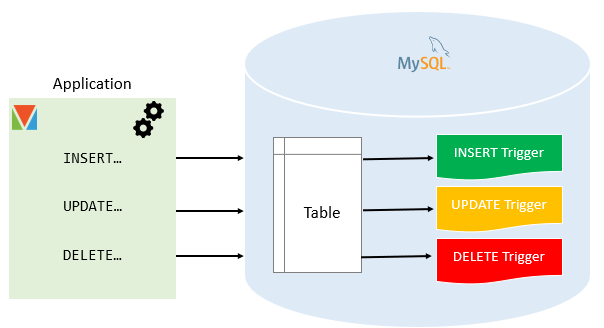
In MySQL, a trigger is a stored program invoked automatically in response to an event such as [insert](https://www.mysqltutorial.org/mysql-insert-statement.aspx), [update](https://www.mysqltutorial.org/mysql-update-data.aspx), or [delete](https://www.mysqltutorial.org/mysql-delete-statement.aspx) that occurs in the associated table. For example, you can define a trigger that is invoked automatically before a new row is inserted into a table.

MySQL supports triggers that are invoked in response to the [INSERT](https://www.mysqltutorial.org/mysql-insert-statement.aspx), [UPDATE](https://www.mysqltutorial.org/mysql-update-data.aspx) or [DELETE](https://www.mysqltutorial.org/mysql-delete-statement.aspx) event.

The SQL standard defines two types of triggers: row-level triggers and statement-level triggers.

* Um **row-level trigger** é ativada para cada registro que é inserido, atualizado, ou apagado. Por exemplo, se uma tabela tiver 100 registros inseridas, atualizadas ou apagadas, o gatilho é automaticamente invocado 100 vezes para os 100 registros afetados.
* Um **statement-level trigger** é executado uma vez para cada transação independentemente de quantas linhas sejam inseridas, atualizadas ou eliminadas.

MySQL supports only row-level triggers. It doesn’t support statement-level triggers.



## Advantages of triggers

* Triggers provide another way to check the integrity of data.
* Triggers handle errors from the database layer.
* Triggers give an alternative way to [run scheduled tasks](https://www.mysqltutorial.org/mysql-triggers/working-mysql-scheduled-event/). By using triggers, you don’t have to wait for the [scheduled events](https://www.mysqltutorial.org/mysql-triggers/working-mysql-scheduled-event/) to run because the triggers are invoked automatically before or after a change is made to the data in a table.
* Triggers can be useful for auditing the data changes in tables.

## Disadvantages of triggers

* Triggers can only provide extended validations, not all validations. **For simple validations, you can use the** [**NOT NULL**](https://www.mysqltutorial.org/mysql-not-null-constraint/)**,** [**UNIQUE**](https://www.mysqltutorial.org/mysql-unique-constraint/)**,** [**CHECK**](https://www.mysqltutorial.org/mysql-check-constraint/) **and** [**FOREIGN KEY**](https://www.mysqltutorial.org/mysql-foreign-key/) **constraints.**
* Triggers can be difficult to troubleshoot because they execute automatically in the database, which may not invisible to the client applications.
* Triggers may increase the overhead of the MySQL Server.

## Managing MySQL triggers

* [Create triggers](https://www.mysqltutorial.org/create-the-first-trigger-in-mysql.aspx) – describe steps of how to create a trigger in MySQL.
* [Drop triggers](https://www.mysqltutorial.org/mysql-triggers/mysql-drop-trigger/) – show you how to drop a trigger.
* [Create a BEFORE INSERT trigger](https://www.mysqltutorial.org/mysql-triggers/mysql-before-insert-trigger/) – show you how to create a BEFORE INSERT trigger to maintain a summary table from another table.
* [Create an AFTER INSERT trigger](https://www.mysqltutorial.org/mysql-triggers/mysql-after-insert-trigger/) – describe how to create an AFTER INSERT trigger to insert data into a table after inserting data into another table.
* [Create a BEFORE UPDATE trigger](https://www.mysqltutorial.org/mysql-triggers/mysql-before-update-trigger/) – learn how to create a BEFORE UPDATE trigger that validates data before it is updated to the table.
* [Create an AFTER UPDATE trigger](https://www.mysqltutorial.org/mysql-triggers/mysql-after-update-trigger/) – show you how to create an AFTER UPDATE trigger to log the changes of data in a table.
* [Create a BEFORE DELETE trigger](https://www.mysqltutorial.org/mysql-triggers/mysql-before-delete-trigger/) – show how to create a BEFORE DELETE trigger.
* [Create an AFTER DELETE trigger](https://www.mysqltutorial.org/mysql-triggers/mysql-after-delete-trigger/) – describe how to create an AFTER DELETE trigger.
* [Create multiple triggers for a table that have the same trigger event and time](https://www.mysqltutorial.org/mysql-triggers/create-multiple-triggers-for-the-same-trigger-event-and-action-time/) – MySQL 8.0 allows you to define multiple triggers for a table that have the same trigger event and time.
* [Show triggers](https://www.mysqltutorial.org/mysql-triggers/mysql-show-triggers/) – list triggers in a database, table by specific patterns.

https://www.mysqltutorial.org/mysql-triggers.aspx

# Create Trigger in MySQL

Summary: in this tutorial, you will learn how to use the MySQL CREATE TRIGGER statement to create a trigger in the database.

## Introduction to MySQL CREATE TRIGGER statement

The CREATE TRIGGER statement creates a new trigger. Here is the basic syntax of the CREATE TRIGGER statement:

CREATE TRIGGER trigger\_name

{BEFORE | AFTER} {INSERT | UPDATE| DELETE }

ON table\_name FOR EACH ROW

trigger\_body;

Code language: SQL (Structured Query Language)   
(sql)

In this syntax:

* First, specify the name of the trigger that you want to create after the CREATE TRIGGER keywords. Note that the trigger name must be unique within a database.
* Next, specify the trigger action time which can be either BEFORE or AFTER which indicates that the trigger is invoked before or after each row is modified.
* Then, specify the operation that activates the trigger, which can be [INSERT](https://www.mysqltutorial.org/mysql-insert-statement.aspx), [UPDATE](https://www.mysqltutorial.org/mysql-update-data.aspx), or [DELETE](https://www.mysqltutorial.org/mysql-delete-statement.aspx).
* After that, specify the name of the table to which the trigger belongs after the ON keyword.
* Finally, specify the statement to execute when the trigger activates. If you want to execute multiple statements, you use the BEGIN END compound statement.

The trigger body can access the values of the column being affected by the DML statement.

Para distinguir entre o valor das colunas ANTES e APÓS o DML ter disparado, usa-se os modificadores NEW e OLD.

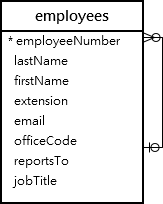
Por exemplo, se atualizar a descrição da coluna, no corpo da trigger, pode acessar o valor da descrição antes da atualização OLD.description e ao novo valor NEW.description.

A tabela seguinte ilustra a disponibilidade dos modificadores OLD e NEW:

|  |  |  |
| --- | --- | --- |
| Trigger Event | OLD | NEW |
| INSERT | No | Yes |
| UPDATE | Yes | Yes |
| DELETE | Yes | No |

## MySQL trigger examples

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



First, [create a new table](https://www.mysqltutorial.org/mysql-create-table/) named employees\_audit to keep the changes to the employees table:

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

);

Code language: SQL (Structured Query Language) (sql)

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

CREATE TRIGGER before\_employee\_update

BEFORE UPDATE ON employees

FOR EACH ROW

INSERT INTO employees\_audit

SET action = 'update',

employeeNumber = OLD.employeeNumber,

lastname = OLD.lastname,

changedat = NOW();

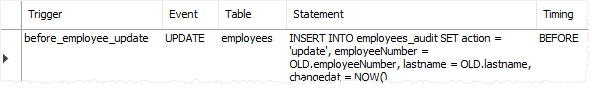
Code language: SQL (Structured Query Language) (sql)

Dentro do corpo do gatilho, utilizamos a palavra-chave OLD para aceder aos valores das colunas employeeNumber e último nome da linha afectada pelo gatilho.

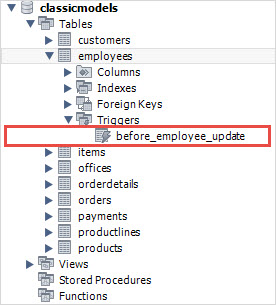
Then, show all triggers in the current database by using the SHOW TRIGGERS statement:

SHOW TRIGGERS;

Code language: SQL (Structured Query Language) (sql)



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



After that, update a row in the employees table:

UPDATE employees

SET

lastName = 'Phan'

WHERE

employeeNumber = 1056;

Code language: SQL (Structured Query Language) (sql)

Finally, query the employees\_audit table to check if the trigger was fired by the UPDATE statement:

SELECT \* FROM employees\_audit;

Code language: SQL (Structured Query Language) (sql)

The following shows the output of the query:



As you see clearly from the output, the trigger was automatically invoked and inserted a new row into the employees\_audit table.

In this tutorial, you have learned how to use the MySQL CREATE TRIGGER statement to create a new trigger in the database.

https://www.mysqltutorial.org/create-the-first-trigger-in-mysql.aspx

# MySQL Triggers with Examples

Share

Triggers are stored programs that are automatically run in response to an event. The event could be an insert, an update, or a delete statement.

There are two types of triggers in SQL.

* Row-level triggers: The triggers are executed after insertion/deletion/update to a single row.
* Statement-level triggers: These triggers are executed once every transaction.

MySQL supports only Row-level triggers.

### **Advantages of triggers:**

* Maintain the integrity of data
* Better error-handling capacity
* Enable edits-auditing
* Faster data updates. We do not need to wait for scheduled tasks and scripts to run ti update data.

Triggers of course come with added processing overhead. Also, triggers allow only a limited number of validations.

## How to create a trigger in MySQL?

The Basic syntax to create a trigger is

CREATE TRIGGER trigger\_name

{BEFORE | AFTER} {INSERT | UPDATE| DELETE }

ON table\_name FOR EACH ROW

trigger\_body;

Let us look at examples related to some of the options.

We will consider a banking system that has an audit log table. We will insert transactions done in the audit\_log table using triggers.

### **Before insert trigger in MySQL:**

BEFORE INSERT triggers are automatically fired before an insert event occurs on the table. For example, Whenever a customer logs in to the system, even before he does any task, we wish to audit the log-in transactions. We can do so as follows:

CREATE TRIGGER login\_trigger

BEFORE INSERT ON customers

FOR EACH ROW

INSERT INTO audit\_log

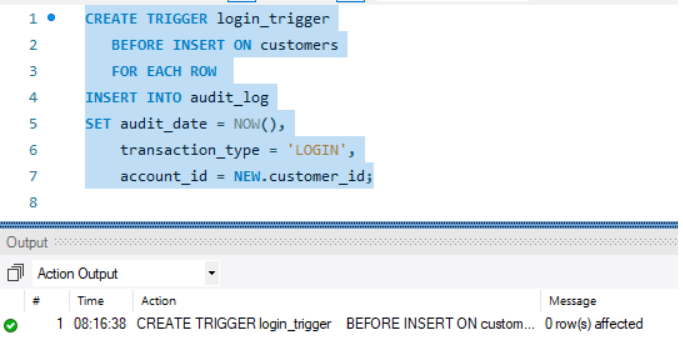
SET audit\_id = (select count(\*) from customers),

audit\_date = NOW(),

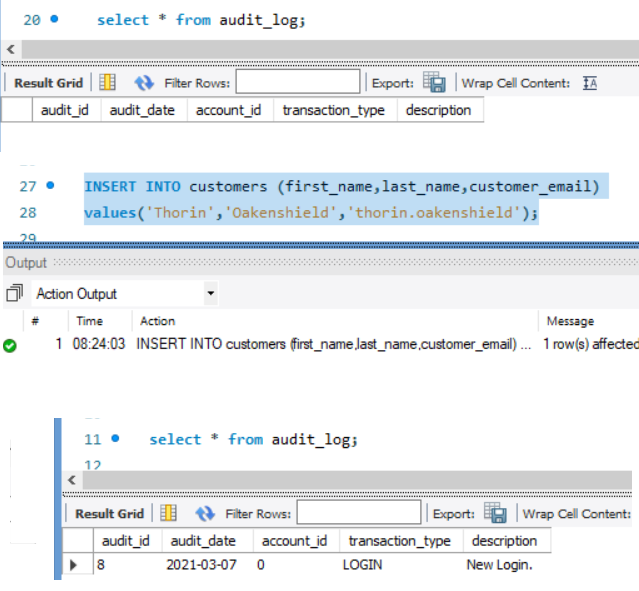
transaction\_type = 'LOGIN',

account\_id = NEW.customer\_id,

description = 'New Login.';



Now before inserting a record in a customer, we will get a record in the audit\_log without explicitly writing an insert.



### **After insert trigger in MySQL:**

AFTER INSERT triggers are automatically invoked after an insert event occurs on the table. Let's say after inserting a record in the transactions table, we want to audit it, we can do:

CREATE TRIGGER audit\_transaction

AFTER INSERT ON transactions

FOR EACH ROW

INSERT INTO audit\_log

SET audit\_id = (select max(transaction\_id) from transactions),

audit\_date = NOW(),

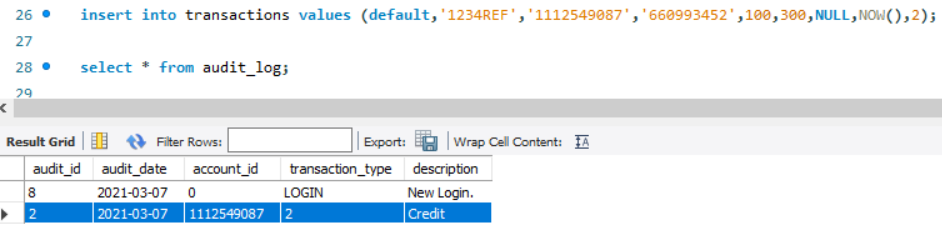
transaction\_type = (select type\_id from transactions having max(transaction\_id)),

account\_id = NEW.sender\_account\_number,

description = (select type\_name from transaction\_types where type\_id in (select type\_id from transactions having max(transaction\_id)));

For example, after inserting the below record, the following will be seen in the audit\_log table.

insert into transactions values (default,'1234REF','1112549087','660993452',100,300,NULL,NOW(),2);



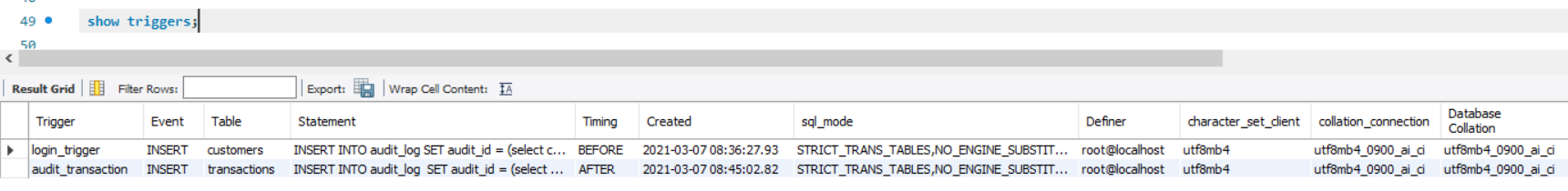
Similarly, we can create CREATE BEFORE Update, CREATE AFTER Update, CREATE Before delete and Create after Delete triggers as well.

The format remains the same.

## How to show triggers in MySQL?

To see what triggers are available, we can use the Show Triggers statement

Show Triggers;



## How to drop triggers in MySQL?

To drop triggers, we use the drop trigger statement. The Basic syntax is

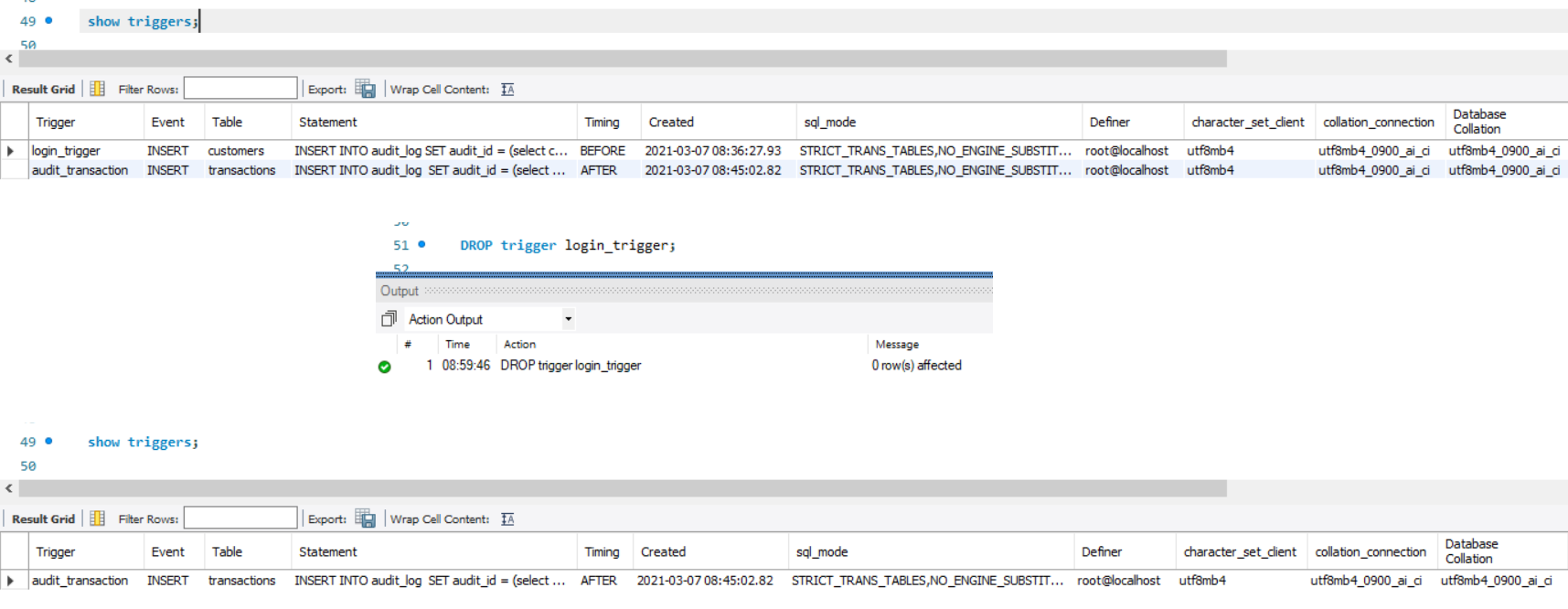
DROP TRIGGER [IF EXISTS] [schema\_name.]trigger\_name;

If we do not use the If Exists clause and try to drop a trigger that does not exist, MySQL throws an error. Otherwise, a warning is issued.

### Example:

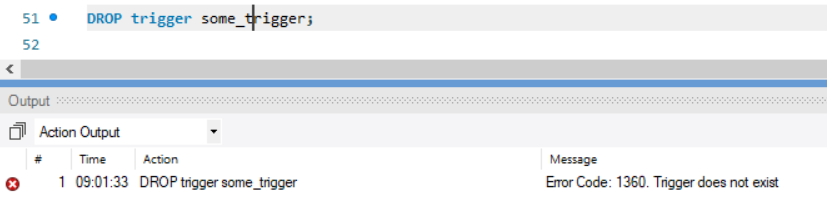
To drop the login\_trigger and remove it, we do:

DROP trigger login\_trigger;



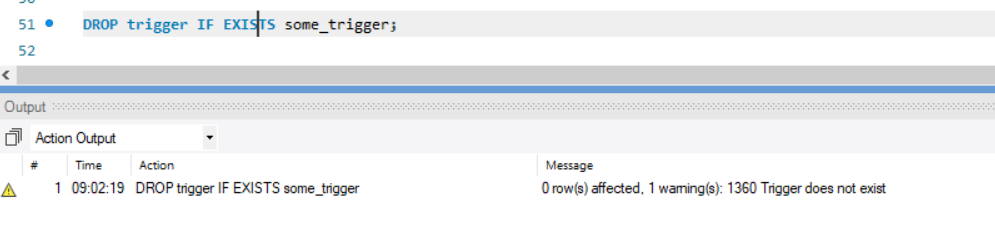
If we try to drop a trigger that does not exist, MySQL throws an error. For example

DROP trigger some\_trigger;



To avoid this error we can use the Drop IF Exists construct. The IF Exists checks if the trigger exists or not. If the trigger does not exist, then only a warning is issued, the execution is not halted due to an error.

DROP trigger IF EXISTS some\_trigger;



# MySQL SHOW TRIGGERS

Summary: in this tutorial, you will learn how to use the MySQL SHOW TRIGGERS statement to show all triggers in a MySQL Server.

## Introduction to MySQL SHOW TRIGGER statement

The SHOW TRIGGERS statement shows all triggers. The following illustrates the basic syntax of the SHOW TRIGGERS statement:

SHOW TRIGGERS

[{FROM | IN} database\_name]

[LIKE 'pattern' | WHERE search\_condition];

Code language: SQL (Structured Query Language) (sql)

In this syntax, if you don’t use the last two clauses, the SHOW TRIGGERS returns all triggers in all databases:

SHOW TRIGGERS;

Code language: SQL (Structured Query Language) (sql)

To show all triggers in a particular database, you specify the name of the database in after the FROM or IN keyword:

SHOW TRIGGERS

FROM database\_name;

Code language: SQL (Structured Query Language) (sql)

or

SHOW TRIGGERS

IN database\_name;

Code language: SQL (Structured Query Language) (sql)

To find trigger using pattern matching, you use the LIKE clause:

SHOW TRIGGERS

LIKE 'pattern';

Code language: SQL (Structured Query Language) (sql)

or

SHOW TRIGGERS

FROM database\_name

LIKE 'pattern';

Code language: SQL (Structured Query Language) (sql)

The meaning of the [LIKE](https://www.mysqltutorial.org/mysql-like/) clause is the same as in the [SELECT](https://www.mysqltutorial.org/mysql-select-statement-query-data.aspx) statement.

To list triggers by a specific search condition, you use the WHERE clause:

SHOW TRIGGERS

WHERE search\_condition;

Code language: SQL (Structured Query Language) (sql)

or

SHOW TRIGGERS

FROM database\_name

WHERE search\_condition;

Code language: SQL (Structured Query Language) (sql)

The SHOW TRIGGERS statement returns a result set that includes the following columns:

* Trigger: the name of the trigger
* Event: the event that invokes the trigger e.g., INSERT, UPDATE, or DELETE.
* Table: the table to which the trigger belongs.
* Statement: the body of the trigger.
* Timing: the activation time of the trigger, either BEFORE or AFTER.
* created: the created time of the trigger.
* sql\_mode: the SQL\_MODE when the trigger executes.
* Definer: the user account that created the trigger.
* character\_set\_client
* collation\_connection
* Database Collation

Notice that to execute the SHOW TRIGGERS statement, you must have the SUPER privilege.

## MySQL SHOW TRIGGER statement examples

To show all triggers in all databases in a MySQL Server, you use this statement:

SHOW TRIGGERS;

Code language: SQL (Structured Query Language) (sql)

For example, to show all triggers in the classicmodels database, you use this statement:

SHOW TRIGGERS

FROM classicmodels;

Code language: SQL (Structured Query Language) (sql)

This example shows all triggers associated with the employees table:

SHOW TRIGGERS

FROM classicmodels

WHERE table = 'employees';

Code language: SQL (Structured Query Language) (sql)

In this tutorial, you have learned how to use the MySQL SHOW TRIGGERS statement to show all triggers in a database server.