Introduction to MySQL AFTER INSERT triggers

MySQL AFTER INSERT triggers are automatically invoked after an insert event occurs on the table.

The following shows the basic syntax of creating a MySQL AFTER INSERT trigger:

**CREATE** **TRIGGER** trigger\_name

**AFTER** **INSERT**

**ON** table\_name **FOR** **EACH** **ROW**

trigger\_body

In this syntax:

First, specify the name of the trigger that you want to create after the [CREATE TRIGGER](https://www.mysqltutorial.org/create-the-first-trigger-in-mysql.aspx) keywords.

Second, use AFTER INSERT clause to specify the time to invoke the trigger.

Third, specify the name of the table on which you want to create the trigger after the ON keyword.

Finally, specify the trigger body which consists of one or more statements that execute when the trigger is invoked.

In case the trigger body has multiple statements, you need to use the BEGIN END block and change the default [delimiter](https://www.mysqltutorial.org/mysql-stored-procedure/mysql-delimiter/):

DELIMITER $$

**CREATE** **TRIGGER** trigger\_name

**AFTER** **INSERT**

**ON** table\_name **FOR** **EACH** **ROW**

**BEGIN**

*-- statements*

**END**$$

DELIMITER ;

In an AFTER INSERT trigger, you can access the NEW values but you cannot change them. Also, you cannot access the OLD values because there is no OLD on INSERT triggers.

MySQL AFTER INSERT trigger example

Consider the following AFTER INSERT trigger example.

Setting up a sample table

First, [create a new table](https://www.mysqltutorial.org/mysql-create-table/) called members:

**DROP** **TABLE** **IF** **EXISTS** members;

**CREATE** **TABLE** members (

**id** INT AUTO\_INCREMENT,

**name** VARCHAR(100) **NOT** NULL,

email VARCHAR(255),

birthDate DATE,

PRIMARY **KEY** (**id**)

);

Second, create another table called reminders that stores reminder messages to members.

**DROP** **TABLE** **IF** **EXISTS** reminders;

**CREATE** **TABLE** reminders (

**id** INT AUTO\_INCREMENT,

memberId INT,

message VARCHAR(255) **NOT** NULL,

PRIMARY **KEY** (**id** , memberId)

);

Creating AFTER INSERT trigger example

The following statement creates an AFTER INSERT trigger that inserts a reminder into the reminders table if the birth date of the member is NULL.

DELIMITER $$

**CREATE** **TRIGGER** after\_members\_insert

**AFTER** **INSERT**

**ON** members **FOR** **EACH** **ROW**

**BEGIN**

**IF** NEW.birthDate **IS** NULL **THEN**

**INSERT** **INTO** reminders(memberId, message)

**VALUES**(new.id,**CONCAT**('Hi ', NEW.name, ', please update your date of birth.'));

**END** **IF**;

**END**$$

DELIMITER ;

In this trigger:

First, the name of the trigger is after\_members\_insert specified in the CREATE TRIGGER clause:

**CREATE** **TRIGGER** after\_members\_insert

Second, the triggering event is:

AFTER **INSERT**

hird, the table that the trigger associated with is members table:

ON members FOR EACH ROW

Finally, inside the trigger body, insert a new row into the reminder table if the birth date of the member is NULL.

Testing the MySQL AFTER INSERT trigger

First, [insert two rows](https://www.mysqltutorial.org/mysql-insert-multiple-rows/) into the members table:

**INSERT** **INTO** members(**name**, email, birthDate)

**VALUES**

('John Doe', 'john.doe@example.com', NULL),

('Jane Doe', 'jane.doe@example.com','2000-01-01');

Second, [query data](https://www.mysqltutorial.org/mysql-select-statement-query-data.aspx) from the members table:

**SELECT** \* **FROM** members;

MySQL AFTER INSERT Trigger example

Third, query data from reminders table:

**SELECT** \* **FROM** reminders;

MySQL AFTER INSERT Trigger Output

We inserted two rows into the members table. However, only the first row that has a birth date value NULL, therefore, the trigger inserted only one row into the reminders table.