**Lab 6: Triggers**

## MySQL Triggers

## Triggers:

In MySQL, a trigger is a stored program invoked automatically in response to an event such as [insert](http://www.mysqltutorial.org/mysql-insert-statement.aspx), [update](http://www.mysqltutorial.org/mysql-update-data.aspx), or [delete](http://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](http://www.mysqltutorial.org/mysql-insert-statement.aspx),  [UPDATE](http://www.mysqltutorial.org/mysql-update-data.aspx)  or  [DELETE](http://www.mysqltutorial.org/mysql-delete-statement.aspx)  event.

## MySQL BEFORE INSERT triggers

MySQL BEFORE INSERT [triggers](http://www.mysqltutorial.org/mysql-triggers.aspx) are automatically fired before an [insert](http://www.mysqltutorial.org/mysql-insert-statement.aspx) event occurs on the table.

We will create a BEFORE INSERT trigger to maintain a summary table from another table.

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

CREATE TABLE WorkCenters (

    id INT AUTO\_INCREMENT PRIMARY KEY,

    name VARCHAR(100) NOT NULL,

    capacity INT NOT NULL

);

Second, create another table called WorkCenterStats that stores the summary of the capacity of the work centers:

CREATE TABLE WorkCenterStats(

    totalCapacity INT NOT NULL

);

**Creating BEFORE INSERT trigger example**

The following trigger updates the total capacity in the WorkCenterStats table before a new work center is inserted into the WorkCenter table:

DELIMITER $$

CREATE TRIGGER before\_workcenters\_insert

BEFORE INSERT

ON WorkCenters FOR EACH ROW

BEGIN

    DECLARE rowcount INT;

    SELECT COUNT(\*)

    INTO rowcount

    FROM WorkCenterStats;

    IF rowcount > 0 THEN

        UPDATE WorkCenterStats

        SET totalCapacity = totalCapacity + new.capacity;

    ELSE

        INSERT INTO WorkCenterStats(totalCapacity)

        VALUES(new.capacity);

    END IF;

END $$

DELIMITER ;

In this trigger:

If the table WorkCenterStats has a row, the trigger adds the capacity to the  totalCapacity  column. Otherwise, it inserts a new row into the WorkCenterStats table.

In the above example, there is new keyword '**NEW**' which is a MySQL extension to triggers. There are two MySQL extension to triggers '**OLD**' and '**NEW**'. OLD and NEW are not case sensitive.

* Within the trigger body, the old and new keywords enable you to access columns in the rows affected by a trigger
* In an INSERT trigger, only new.col\_name can be used.
* In a UPDATE trigger, you can use OLD.col\_name to refer to the columns of a row before it is updated and NEW.col\_name to refer to the columns of the row after it is updated.
* In a DELETE trigger, only OLD.col\_name can be used; there is no new row.

### Testing the MySQL BEFORE INSERT trigger

First, [insert a new row](http://www.mysqltutorial.org/mysql-insert-statement.aspx) into the WorkCenter table:

INSERT INTO WorkCenters(name, capacity)

VALUES('Mold Machine',100);

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

SELECT \* FROM WorkCenterStats;

http://www.mysqltutorial.org/wp-content/uploads/2019/09/MySQL-BEFORE-INSERT-Trigger-Example.png

The trigger has been invoked and inserted a new row into the WorkCenterStats table.

Third, insert a new work center:

INSERT INTO WorkCenters(name, capacity)

VALUES('Packing',200);

Finally, query data from the WorkCenterStats:

SELECT \* FROM WorkCenterStats;

MySQL BEFORE INSERT Trigger Example 2

## MySQL AFTER INSERT Triggers

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

Consider the following AFTER INSERT trigger example.

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

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 ;

### Testing the MySQL AFTER INSERT trigger

First, [insert two rows](http://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](http://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.

## MySQL BEFORE DELETE trigger

Let’s see the following BEFORE DELETE trigger example.

**CREATE** **TABLE** Salaries (

employeeNumber INT PRIMARY **KEY**,

validFrom DATE **NOT** NULL,

amount DECIMAL(12 , 2 ) **NOT** NULL **DEFAULT** 0

);

Let’s insert some values;

**INSERT** **INTO** salaries(employeeNumber,validFrom,amount)

**VALUES**

(1002,'2000-01-01',50000),

(1056,'2000-01-01',60000),

(1076,'2000-01-01',70000);

**CREATE** **TABLE** SalaryArchives (

**id** INT PRIMARY **KEY** AUTO\_INCREMENT,

employeeNumber INT,

validFrom DATE **NOT** NULL,

amount DECIMAL(12 , 2 ) **NOT** NULL **DEFAULT** 0,

deletedAt TIMESTAMP **DEFAULT** **NOW**()

);

### Creating BEFORE DELETE trigger example

The following BEFORE DELETE trigger inserts a new row into the SalaryArchives table before a row from the Salaries table is deleted.

DELIMITER $$

**CREATE** **TRIGGER** before\_salaries\_delete

**BEFORE** **DELETE**

**ON** salaries **FOR** **EACH** **ROW**

**BEGIN**

**INSERT** **INTO** SalaryArchives(employeeNumber,validFrom,amount)

**VALUES**(OLD.employeeNumber,OLD.validFrom,OLD.amount);

**END**$$

DELIMITER ;

### Testing the MySQL BEFORE DELETE trigger

First, delete a row from the Salaries table:

**DELETE** **FROM** salaries

**WHERE** employeeNumber = 1002;

Second, query data from the SalaryArchives table:

**SELECT** \* **FROM** SalaryArchives;



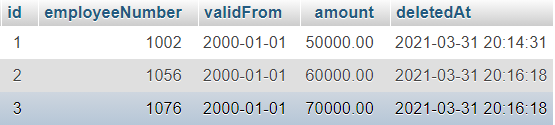
The trigger was invoked and inserted a new row into the SalaryArchives table.

Third, delete all rows from the Salaries table:

**DELETE** **FROM** salaries;

Finally, query data from the SalaryArchives table:

**SELECT** \* **FROM** SalaryArchives;



The trigger was trigger twice because the DELETE statement deleted two rows from the Salaries table.

## MySQL AFTER DELETE trigger

Consider the following AFTER DELETE trigger example.

### Setting up a sample table

**INSERT** **INTO** Salaries(employeeNumber,amount)

**VALUES**

(10002,5000),

(10056,7000),

(10076,8000);

Third, create another table called SalaryBudgets that stores the total of salaries from the Salaries table:

**CREATE** **TABLE** SalaryBudgets(

total DECIMAL(15,2) **NOT** NULL

);

Fourth, use the [SUM()](https://www.mysqltutorial.org/mysql-sum/) function to get the total salary from the Salaries table and insert it into the SalaryBudgets table:

**INSERT** **INTO** SalaryBudgets(total)

**SELECT** **SUM**(amount)

**FROM** Salaries;

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

**SELECT** \* **FROM** SalaryBudgets;



### Creating AFTER DELETE trigger example

The following AFTER DELETE trigger updates the total salary in the SalaryBudgets table after a row is deleted from the Salaries table:

**CREATE** **TRIGGER** after\_salaries\_delete

**AFTER** **DELETE**

**ON** Salaries **FOR** **EACH** **ROW**

**UPDATE** SalaryBudgets

**SET** total = total - old.amount;

### Testing the MySQL AFTER DELETE trigger

First, delete a row from the Salaries table:

**DELETE** **FROM** Salaries

**WHERE** employeeNumber = 10002;

Second, query total salary from the SalaryBudgets table:

**SELECT** \* **FROM** SalaryBudgets;



As you can see from the output, the total is reduced by the deleted salary.

Third, delete all rows from the salaries table:

**DELETE** **FROM** Salaries;

Finally, query the total from the SalaryBudgets table:

**SELECT** \* **FROM** SalaryBudgets;

The trigger updated the total to 0.00 in SalaryBudgets.

