MySQL Triggers – Log of Record (LAB Lecture)

Department of Information Technology
Database Systems II (IT216)
Spring 2024-2025
Week 15 – May 18, 2025
Lecturer: Soma Soleimanzadeh

Contents

- Triggers in MySQL
- Syntax of Creating a Trigger in MySQL
- Triggers for
  1. Log of Record (The type of triggers in this lecture)
  2. Validating Input Data

Why We Need Triggers in Database?

- Triggers are useful in many situations. Some of the main reasons for using triggers are:
  - Keeping a **Log of Records**
  - **Validating Input Data**
  - **Enforcing Business Rules**

3

Syntax of Creating TRIGGER in MySQL

```
DELIMITER //
CREATE TRIGGER trigger_name
(BEFORE | AFTER) (INSERT | UPDATE | DELETE) ON table_name
FOR EACH ROW
BEGIN
    <Trigger Statements>
END//
DELIMITER ;
```

4

Log of Records Scenario

- Suppose there is a table to store employees' data in our database.
- We want to make a log of records that let us know about any modifications that happen on the table (**Employee** table).
- We will create an empty table (for example, the **Emp_Log** table) to store the log of records. For example:
    - Which **user** did which **type of modification** at which **date** and **time** on the **Employee** table?

5

Log of Records Scenario

Employee Table

EID	Ename	Age

Employee Table

EID	Ename	Age
1	Hasan	44
2	Lana	36 → 37

```
insert into Employee(Ename, Age) values ('Hasan', 44), ('Lana', 36);
update Employee set Age = 37 where EID = 2;
```

Emp_Log Table

id	Action_Name	Old_Age	New_Age	By_User	Action_Date	Action_Time
1	Insert	Null	44	root	10 May 2025	06:12:45 PM
2	Insert	Null	36	root	14 May 2025	10:07:26 AM
3	Update	36	37	root	15 May 2025	09:00:31 AM

Log of Records Scenario

- We need **3 triggers** that make a log of records when any modification (INSERT, UPDATE or DELETE) happens on **Employee** table.

  - First trigger is activated when **INSERT** happens.
  - Second trigger is activated when **DELETE** happens.
  - Third trigger is activated when **UPDATE** happens.

7

Let's Create Database and Tables

- Create **Company** database, and activate it.
- Create both **Employee** and **Emp_Log** tables.

Employee Table

EID	Ename	Age

Emp_Log Table

id	Action_Name	Old_Age	New_Age	By_User	Action_Date	Action_Time

8

Let's Create Database and Tables

```
create database company;
use company;


create table Employee
    (EID int auto_increment,
     EName varchar(100),
     Age  int,
     primary key (EID));
```

```
create table Emp_Log
    (id int auto_increment,
     Action_Name varchar(15),
     Old_Age int,
     New_Age int,
     By_User varchar(50),
     Action_Date date,
     Action_Time time,
     primary key(id));
```

9

INSERT Trigger

- Create **INSERT Trigger** on the **Employee** table, which is activated when any **insert** happens on the **Employee** table. This trigger is only for the log of records.

```
DELIMITER //
CREATE TRIGGER insert_emp_log
AFTER INSERT ON Employee
FOR EACH ROW
BEGIN
        insert into Emp_Log
        set
            Action_Name = 'Insert',
            New_Age = new.Age,
            By_User = user(),
            Action_Date = current_date(),
            Action_Time = current_time();
END//
DELIMITER ;
```

The **user()** function is a built-in function in MySQL that returns **the current user name and hostname** for the MySQL connection being used by the user.

The **current_date()** and **current_time()** functions are built-in functions in MySQL that return the current date and current time respectively.

10

DELETE Trigger

- Create **DELETE Trigger** on the **Employee** table, which is activated when any **delete** happens on the **Employee** table. This trigger is only for the log of records.

```
DELIMITER //
CREATE TRIGGER delete_emp_log
AFTER DELETE ON Employee
FOR EACH ROW
BEGIN
        insert into Emp_Log
        set
            Action_Name = 'Delete',
            Old_Age = old.Age,
            By_User = user(),
            Action_Date = current_date(),
            Action_Time = current_time();
END//
DELIMITER ;
```

11

UPDATE Trigger

- Create **UPDATE Trigger** on the **Employee** table, which is activated when any **update** happens on the **Employee** table. This trigger is only for the log of records.

```
DELIMITER //
CREATE TRIGGER update_emp_log
AFTER UPDATE ON Employee
FOR EACH ROW
BEGIN
        insert into Emp_Log
        set
            Action_Name = 'Update',
            Old_Age = old.Age,
            New_Age = new.Age,
            By_User = user(),
            Action_Date = current_date(),
            Action_Time = current_time();
END//
DELIMITER ;
```

12

Testing Triggers

- Now you can test how the triggers work by executing INSERT, DELETE and UPDATE statements on the **Employee** table and checking the log of records in the **Emp_Log** table.

```
INSERT INTO Employee(EName, Age) VALUES ('Ali' , 26);

INSERT INTO Employee(EName, Age) VALUES ('Kawa' , 30);

UPDATE Employee SET Age = 25 WHERE EID = 1;

DELETE FROM Employee WHERE EID = 2;
```

```
SELECT * FROM Emp_Log;
```

13