MySQL Triggers – Validating Data (LAB Lecture)

Department of Information Technology
Database Systems II (IT216)
Spring 2024-2025
Week 16 – May 25, 2025
Lecturer: Soma Soleimanzadeh

1

Contents

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

2

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

Validating Input Data Scenario

- Suppose there is a table to store employees' data in our database.
- The **employee** table has a column named **Age** to store the age of employees.
- The **Age** column can not have negative values, so we are going to create a trigger that doesn't allow entering negative values in the **Age** column.
    - Whenever a user tries to **insert** a row in **Employee** table that contains negative number in **Age** column, the trigger **will generate an Error message** and doesn't allow the **insert** happens.

5

Validating Input Data Scenario

Employee Table

EID	Ename	Age
1	Hasan	44
2	Lana	36

Employee Table

EID	Ename	Age
1	Hasan	44
2	Lana	36

insert into Employee(Ename, Age) **values** ('Kawa', - 70);

ERROR: Age Can not be Negative!

6

Let's Create Database and Table

- Create **Company** database, and activate it.
- Create **Employee** table.

Employee Table

EID	Ename	Age

```
create database company;
use company;


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

Validate_Age Trigger

- Create a **Trigger** on the **Employee** table, which is activated when any **insert** happens on the **Employee** table. The trigger checks the inserted value for the **Age** column, and if it is negative, it shows an Error message to the client and doesn't allow the insertion to happen.

```
DELIMITER $$
CREATE TRIGGER validate_age_tg
BEFORE INSERT ON Employee
FOR EACH ROW
BEGIN
    IF NEW.Age < 0 THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Age cannot be negative!' ;
    END IF;
END$$
DELIMITER ;
```

8

Testing the Trigger

- Now you can test how the trigger works by executing an INSERT statement to enter a row with a negative **age** value into the **Employee** table.

```
INSERT INTO Employee(EName, Age) VALUES ('Sara' , -17);
INSERT INTO Employee(EName, Age) VALUES ('Dana' , 22);
```

```
SELECT * FROM Employee;
```

9

Example 2

- Create a trigger that considers the following limitation whenever any update is going to happen in the **Employee** table.

Limitation for Age: **10 <= Age <= 60**

- This trigger considers the above limitation for **age**,
  - if the updated age becomes more than 60, the trigger sets the **Age** to 60,
  - if the updated age becomes less than 10, the trigger sets the **Age** to 10.

10

Example 2

```
delimiter //
CREATE TRIGGER age_limitation_tg
BEFORE UPDATE ON employee
FOR EACH ROW
BEGIN
    IF NEW.age > 60 THEN SET NEW.age = 60;
    ELSEIF NEW.age < 10 THEN SET NEW.age = 10;
    END IF;
END//
delimiter ;
```

11

Testing the Trigger

- Now you can test how the trigger works by executing UPDATE statements on the **Employee** table.

```
UPDATE Employee SET Age = 70 WHERE EID = 1;
UPDATE Employee SET Age = 45 WHERE EID = 2;
UPDATE Employee SET Age = 5 WHERE EID = 2;
```

```
SELECT * FROM Employee;
```

12