IT Department

Triggers in MySQL (Log of Records)

Soma Soleiman Zadeh

Database Systems II (IT 216)

Spring 2025 - 2026

Week 13

April 27, 2026

Outline

- **Triggers** in MySQL
- Syntax of Creating **Trigger** in MySQL
- **BEFORE** Row_level Triggers vs. **AFTER** Row_level Triggers
- **Triggers** for Log of Record

What is Trigger?

- A **trigger** is a special type of **stored procedure** that is *invoked automatically in response to an event*.
  - Each trigger is associated with a table,
  - which is activated on any **DML statement** such as **INSERT**, **UPDATE**, or **DELETE**.

Why We Need Triggers in Database?

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

Trigger vs. Procedure

- **Trigger** is a special procedure, but
  - **Procedure** must be called by the user.
  - **Trigger** is activated (called) automatically when a data modification event is made on a table.

Row-Level Trigger vs. Statement-Level Trigger

- **Row-Level Trigger :**
  - This type of trigger is executed once for each row affected by a data modification operation, such as **INSERT**, **UPDATE**, or **DELETE**.
- **Statement-Level Trigger :**
  - This type of trigger fires once for each SQL statement executed, **regardless of the number of rows that have been changed**.

Types of Triggers

1. Before Insert : It is activated **before the insertion** of data into the table.
2. After Insert : It is activated **after the insertion** of data into the table.

3. Before Update : It is activated **before the update** of data in the table.
4. After Update : It is activated **after the update** of the data in the table.

5. Before Delete : It is activated **before the deletion** of data from the table.
6. After Delete : It is activated **after the deletion** of data from the table.

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 ;
```

Log of Records Scenario

```
insert into Product values (1, 'Tablet', 150);
```

Product Table

PID	Pname	Price
1	Tablet	150
..	..	..
..	..	..

Product_Log Table

id	actionName	oldPrice	newPrice	byUser	actionDate
1	Insert	Null	150	root	24 Apr 2026

Log of Records Scenario

```
insert into Product values (1, 'Tablet', 150);
insert into Product values (2, 'Laptop', 1200);
```

Product Table

PID	Pname	Price
1	Tablet	150
2	Laptop	1200
..	..	..

Product_Log Table

id	actionName	oldPrice	newPrice	byUser	actionDate
1	Insert	Null	150	root	24 Apr 2026
2	Insert	Null	1200	ali	26 Apr 2026

Product Table

```
create table Product
(PID int auto_increment primary key,
PName varchar(100),
Price int);
```

PID	Pname	Price

Product_Log Table

```
create table product_log
(id int auto_increment primary key,
actionName varchar(10),
oldPrice int,
newPrice int,
byUser varchar(30),
actionDate date);
```

id	actionName	oldPrice	newPrice	byUser	actionDate

Creating Trigger for Log of Records

- Now, we create Triggers for the **Product** table and save the log of data into the **Product_log** table.

Product Table

PID	Pname	Price
1	Tablet	150
2	Laptop	1000
..	..	..

Product_Log Table

id	actionName	oldPrice	newPrice	byUser	actionDate
1	Insert	Null	150	root	24 Apr 2026
2	Insert	Null	1200	ali	26 Apr 2026
3	Update	1200	1000	ali	27 Apr 2026

Creating Insertion Trigger

```
delimiter //
create trigger product_insert
after insert on product
for each row
begin
    insert into product_log
    set
        actionName='Insert',
        newPrice=new.price,
        byUser=user(),
        actionDate = current_date();
end//
delimiter ;
```

The Effect of Insertion Trigger

```
insert into product(PName, Price) values ('Tablet', 150);
insert into product(PName, Price) values ('Laptop', 1200);
```

Product_Log Table

id	actionName	newPrice	oldprice	byUser	actionDate
1	Insert	150	NULL	root@localhost	2026-05-02
2	Insert	1200	NULL	root@localhost	2026-05-02

Creating Deletion Trigger

```
delimiter //
create trigger product_delete
after delete on product
for each row
begin
    insert into product_log
    set
        actionName='Delete',
        oldPrice=old.price,
        byUser=user(),
        actionDate = current_date();
end//
delimiter ;
```

The Effect of Deletion Trigger

```
insert into product(PName, Price) values ('Tablet', 150);
insert into product(PName, Price) values ('Laptop', 1200);
```

```
delete from product where PName = 'Laptop';
```

Product_Log Table

id	actionName	newPrice	oldprice	byUser	actionDate
1	Insert	150	NULL	root@localhost	2026-05-02
2	Insert	1200	NULL	root@localhost	2026-05-02
3	Delete	NULL	1200	root@localhost	2026-05-02

Creating Updating Trigger

```
delimiter //
create trigger product_update
after update on product
for each row
begin
    insert into product_log
    set
        actionName='Update',
        newPrice=new.price,
        oldPrice=old.price,
        byUser=user(),
        actionDate = current_date();
end//
delimiter ;
```

The Effect of Updating Trigger

```
insert into product(PName, Price) values ('Tablet', 150);
insert into product(PName, Price) values ('Laptop', 1200);
```

```
delete from product where PName = 'Laptop';
```

```
update product set Price = 100 where PName='Tablet';
```

Product_Log Table

id	actionName	newPrice	oldprice	byUser	actionDate
1	Insert	150	NULL	root@localhost	2026-05-02
2	Insert	1200	NULL	root@localhost	2026-05-02
3	Delete	NULL	1200	root@localhost	2026-05-02
4	Update	100	150	root@localhost	2026-05-02