Multiple Triggers

In this lesson, we will see how to write multiple triggers for a table which have the same action time and event.

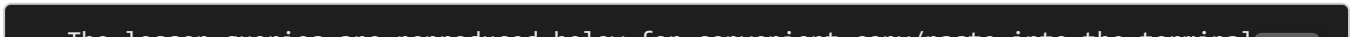
Multiple Triggers

It is possible to create triggers on a table whose action time and event are the same. Such triggers are fired in a sequence that is specified at the time of creation of the triggers. The **FOLLOWS** and **PRECEDES** keywords are used to define the sequence in which triggers associated with a table having the same action time and event execute.

Syntax

> CREATE TRIGGER **trigger_name** [BEFORE | AFTER] [INSERT | UPDATE | DELETE]
>
> ON **table_name**
>
> [FOLLOWS | PRECEDES] **existing_trigger_name**
>
> FOR EACH ROW
>
> **trigger_body**

> Connect to the terminal below by clicking in the widget. Once connected, the command line prompt will show up. Enter or copy-paste the command **./DataJek/Lessons/50lesson.sh** and wait for the mysql prompt to start-up.



```
-- The lesson queries are reproduced below for convenient copy/paste into the terminal.

-- Query 1

CREATE TABLE GenderSummary (
  TotalMales INT NOT NULL,
  TotalFemales INT NOT NULL
);

CREATE TABLE MaritalStatusSummary (
  TotalSingle INT NOT NULL,
  TotalMarried INT NOT NULL,
  TotalDivorced INT NOT NULL
);

CREATE TABLE ActorsTableLog (
  RowId INT AUTO_INCREMENT PRIMARY KEY,
  ActorId INT NOT NULL,
  Detail VARCHAR(100) NOT NULL,
  UpdatedOn TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP
);

-- Query 2
INSERT INTO GenderSummary (TotalMales, TotalFemales)
Values ((SELECT COUNT(Gender) FROM Actors WHERE Gender = 'Male'),
        (SELECT COUNT(Gender) FROM Actors WHERE Gender = 'Female'));

SELECT * FROM GenderSummary;

INSERT INTO MaritalStatusSummary (TotalSingle, TotalMarried, TotalDivorced)
Values ((SELECT COUNT(MaritalStatus) FROM Actors WHERE MaritalStatus = 'Single'),
        (SELECT COUNT(MaritalStatus) FROM Actors WHERE MaritalStatus = 'Married'),
        (SELECT COUNT(MaritalStatus) FROM Actors WHERE MaritalStatus = 'Divorced'));

SELECT * FROM MaritalStatusSummary;

-- Query 3
DELIMITER **
CREATE TRIGGER UpdateGenderSummary
AFTER INSERT
ON Actors
FOR EACH ROW
BEGIN
DECLARE count INT;
IF NEW.Gender = 'Male' THEN
    UPDATE GenderSummary
   SET TotalMales = TotalMales+1;
   INSERT INTO ActorsTableLog (ActorId, Detail)
   VALUES (NEW.Id, 'TotalMales value of GenderSummary table changed.');
ELSE
   UPDATE GenderSummary
   SET TotalFemales = TotalFemales+1;
   INSERT INTO ActorsTableLog (ActorId, Detail)
   VALUES (NEW.Id, 'TotalFemales value of GenderSummary table changed.');
END IF;
END  **
DELIMITER ;

-- Query 4
DELIMITER **
CREATE TRIGGER UpdateMaritalStatusSummary
AFTER INSERT
ON Actors
```

```
FOR EACH ROW
FOLLOWS UpdateGenderSummary
BEGIN
DECLARE count INT;
IF NEW.MaritalStatus = 'Single' THEN
    UPDATE MaritalStatusSummary
   SET TotalSingle = TotalSingle+1;
   INSERT INTO ActorsTableLog (ActorId, Detail)
   VALUES (NEW.Id, 'TotalSingle value of MaritalStatusSummary table changed.');
ELSEIF  NEW.MaritalStatus = 'Married' THEN
    UPDATE MaritalStatusSummary
   SET TotalMarried = TotalMarried+1;
   INSERT INTO ActorsTableLog (ActorId, Detail)
   VALUES (NEW.Id, 'TotalMarried value of MaritalStatusSummary table changed.');
ELSE
   UPDATE MaritalStatusSummary
   SET TotalDivorced = TotalDivorced+1;
   INSERT INTO ActorsTableLog (ActorId, Detail)
   VALUES (NEW.Id, 'TotalDivorced value of MaritalStatusSummary table changed.');
END IF;
END  **
DELIMITER ;

-- Query 5
INSERT INTO Actors (FirstName, SecondName, DoB, Gender, MaritalStatus,  NetWorthInMillions)
VALUES ('Tom', 'Hanks', '1956-07-09', 'Male', 'Married', 350);
SELECT * FROM ActorsTableLog;

-- Query 6
SHOW TRIGGERS;

-- Query 7
SELECT
    trigger_name,
    action_order
FROM
    information_schema.triggers
WHERE
    trigger_schema = 'MovieIndustry';
```

Terminal

1. To demonstrate the order in which two triggers execute for the same event, we will create a simple example. Suppose that we want to perform two tasks when a new record is inserted in the **Actors** table. First, based on the gender of the actor, we want to update the **GenderSummary** table. Second, based on his/her marital status, we want to update the **MaritalStatusSummary** table. We will log these actions in a separate table **ActorsTableLog** to show the order of execution of triggers. To create these tables, execute the following queries:

queries:

```
CREATE TABLE GenderSummary (
  TotalMales INT NOT NULL,
  TotalFemales INT NOT NULL
);

CREATE TABLE MaritalStatusSummary (
  TotalSingle INT NOT NULL,
  TotalMarried INT NOT NULL,
  TotalDivorced INT NOT NULL
);

CREATE TABLE ActorsTableLog (
  RowId INT AUTO_INCREMENT PRIMARY KEY,
  ActorId INT NOT NULL,
  Detail VARCHAR(100) NOT NULL,
  UpdatedOn TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP
);
```

Now run the following queries to enter data in the tables:

```
INSERT INTO GenderSummary (TotalMales, TotalFemales)
Values ((SELECT COUNT(Gender) FROM Actors WHERE Gender = 'Male'),
        (SELECT COUNT(Gender) FROM Actors WHERE Gender = 'Female'
));

SELECT * FROM GenderSummary;

INSERT INTO MaritalStatusSummary (TotalSingle, TotalMarried, Tota
lDivorced)
Values ((SELECT COUNT(MaritalStatus) FROM Actors WHERE MaritalSta
tus = 'Single'),
        (SELECT COUNT(MaritalStatus) FROM Actors WHERE MaritalSta
tus = 'Married'),
        (SELECT COUNT(MaritalStatus) FROM Actors WHERE MaritalSta
tus = 'Divorced'));

SELECT * FROM MaritalStatusSummary;
```

It can be seen that summary data has been entered in the tables.

2. Now we will create the first trigger that updates **GenderSummary**

table after a row is inserted in the **Actors** table as follows:

```
DELIMITER **

CREATE TRIGGER UpdateGenderSummary
AFTER INSERT
ON Actors
FOR EACH ROW
BEGIN

DECLARE count INT;

IF NEW.Gender = 'Male' THEN
    UPDATE GenderSummary
   SET TotalMales = TotalMales+1;

   INSERT INTO ActorsTableLog (ActorId, Detail)
   VALUES (NEW.Id, 'TotalMales value of GenderSummary table chang
ed.');

ELSE
   UPDATE GenderSummary
   SET TotalFemales = TotalFemales+1;

   INSERT INTO ActorsTableLog (ActorId, Detail)
   VALUES (NEW.Id, 'TotalFemales value of GenderSummary table cha
nged.');

END IF;
END  **

DELIMITER ;
```

In this trigger, we first check the **Gender** of the newly inserted actor and increment the **TotalMales** or **TotalFemales** value accordingly. Then a row is inserted in the **ActorsTableLog** which describes which value in the **GenderSummary** table was changed.

3. Next, we will create another trigger **UpdateMaritalStatusSummary** that will execute after the **UpdateGenderSummary** trigger as follows:

```
DELIMITER **

CREATE TRIGGER UpdateMaritalStatusSummary
AFTER INSERT
ON Actors
FOR EACH ROW
FOLLOWS UpdateGenderSummary

BEGIN

DECLARE count INT;

IF NEW.MaritalStatus = 'Single' THEN
    UPDATE MaritalStatusSummary
   SET TotalSingle = TotalSingle+1;

   INSERT INTO ActorsTableLog (ActorId, Detail)
   VALUES (NEW.Id, 'TotalSingle value of MaritalStatusSummary tab
le changed.');

ELSEIF  NEW.MaritalStatus = 'Married' THEN
    UPDATE MaritalStatusSummary
   SET TotalMarried = TotalMarried+1;

   INSERT INTO ActorsTableLog (ActorId, Detail)
   VALUES (NEW.Id, 'TotalMarried value of MaritalStatusSummary ta
ble changed.');

ELSE
   UPDATE MaritalStatusSummary
   SET TotalDivorced = TotalDivorced+1;

   INSERT INTO ActorsTableLog (ActorId, Detail)
   VALUES (NEW.Id, 'TotalDivorced value of MaritalStatusSummary t
able changed.');

END IF;
END  **

DELIMITER ;
```

The **FOLLOWS** keyword is used to define the order of execution of the trigger to be after the **UpdateGenderSummary** trigger.

In this trigger, **IF THEN ELSEIF ELSE** statements are used to check

the **MaritalStatus** of the newly inserted actor and corresponding value in **MaritalStatusSummary** table is updated. Then a row is inserted in the **ActorsTableLog** which describes which value in the **MaritalStatusSummary** table was changed.

4. Both triggers are associated with the same event **AFTER INSERT ON ACTORS**. To test if the triggers are executed in the order defined, we will insert a row in the **Actors** table and then check the **ActorsTableLog** table.

```
INSERT INTO Actors (FirstName, SecondName, DoB, Gender, MaritalSt
atus,  NetWorthInMillions)
VALUES ('Tom', 'Hanks', '1956-07-09', 'Male', 'Married', 350);

SELECT * FROM ActorsTableLog;
```

As it can be seen from the **ActorsTableLog**, the **UpdateGenderSummary** trigger executes first and a row is inserted in the table. Then the **UpdateMaritalStatusSummary** trigger runs and inserts a row in the **ActorsTableLog**.

5. The SHOW TRIGGERS statement is used to display the triggers in the database.

```
SHOW TRIGGERS;
```

This statement does not return any information on the order of execution of triggers if a table has multiple triggers associated with the same event. That information is stored in the **triggers** table in the **information_schema** database. Use the following query to display relevant column of the table:

```
SELECT
    trigger_name,
    action_order
FROM
    information_schema.triggers
```

```
WHERE
    trigger_schema = 'MovieIndustry';
```