LAB Assignment 9

Introduction of Triggers.

MySQL, a trigger is a stored program invoked automatically in response to an event such as insert, update, or delete 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, UPDATE or DELETE event. The SQL standard defines two types of triggers: row-level triggers and statement-level triggers.

A row-level trigger is activated for each row that is inserted, updated, or deleted. For example, if a table has 100 rows inserted, updated, or deleted, the trigger is automatically invoked 100 times for the 100 rows affected.

A statement-level trigger is executed once for each transaction regardless of how many rows are inserted, updated, or deleted.

MySQL supports only row-level triggers. It doesn't support statement-level triggers.

Managing MySQL triggers

- Create triggers – describe steps of how to create a trigger in MySQL.
- Drop triggers – show you how to drop a trigger.
- Create a BEFORE INSERT trigger – show you how to create a `BEFORE INSERT` trigger to maintain a summary table from another table.
- Create an AFTER INSERT trigger – describe how to create an `AFTER INSERT` trigger to insert data into a table after inserting data into another table.
- Create a BEFORE UPDATE trigger – learn how to create a `BEFORE UPDATE` trigger that validates data before it is updated to the table.
- Create an AFTER UPDATE trigger – show you how to create an `AFTER UPDATE` trigger to log the changes of data in a table.
- Create a BEFORE DELETE trigger – show how to create a `BEFORE DELETE` trigger.
- Create an AFTER DELETE trigger – describe how to create an `AFTER DELETE` trigger.
- Create multiple triggers for a table that have the same trigger event and time – MySQL 8.0 allows you to define multiple triggers for a table that have the same trigger event and time.
- Show triggers – list triggers in a database, table by specific patterns.

Advantages of triggers

- Triggers provide another way to check the integrity of data.
- Triggers handle errors from the database layer.
- Triggers give an alternative way to run scheduled tasks. By using triggers, you don't have to wait for the scheduled events to run because the triggers are invoked automatically *before* or *after* a change is made to the data in a table.
- Triggers can be useful for auditing the data changes in tables.

Disadvantages of triggers

- Triggers can only provide extended validations, not all validations. For simple validations, you can use the `NOT NULL`, `UNIQUE`, `CHECK` and `FOREIGN KEY` constraints.
- Triggers can be difficult to troubleshoot because they execute automatically in the database, which may not invisible to the client applications.
- Triggers may increase the overhead of the MySQL Server.

Please follow the url as given below and try all the example.

https://www.w3resource.com/mysql/mysql-triggers.php

Lab Questions:

First, consider the database tables of **lab assignment 8** as per the following schema diagram. All three SQL Scripts are attached on LMS and perform the lab questions.

Table entries are as following.

Salesman Table:

salesman_id	name	city	commission
5001	James Hoog	New York	0.15
5002	Nail Knite	Paris	0.13
5005	Pit Alex	London	0.11
5006	Mc Lyon	Paris	0.14
5007	Paul Adam	Rome	0.13
5003	Lauson Hen	San Jose	0.12

Customer Table:

customer_id | cust_name | city | grade | salesman_id

3002 | Nick Rimando | New York | 100 | 5001

3007 | Brad Davis | New York | 200 | 5001

3005 | Graham Zusi | California | 200 | 5002

3008 | Julian Green | London | 300 | 5002

3004 | Fabian Johnson | Paris | 300 | 5006

3009 | Geoff Cameron | Berlin | 100 | 5003

3003 | Jozy Altidor | Moscow | 200 | 5007

3001 | Brad Guzan | London | | 5005

Orders Table:

ord_no	purch_amt	ord_date	customer_id	salesman_id
70001	150.5	2012-10-05	3005	5002
70009	270.65	2012-09-10	3001	5005
70002	65.26	2012-10-05	3002	5001
70004	110.5	2012-08-17	3009	5003
70007	948.5	2012-09-10	3005	5002
70005	2400.6	2012-07-27	3007	5001
70008	5760	2012-09-10	3002	5001
70010	1983.43	2012-10-10	3004	5006
70003	2480.4	2012-10-10	3009	5003
70012	250.45	2012-06-27	3008	5002
70011	75.29	2012-08-17	3003	5007
70013	3045.6	2012-04-25	3002	5001

Q1. Create a trigger which gives error when data with customer-id 0000 is added in customer table.

Q2. Create a trigger which gives error when data with grade 0 is added into customer table.

Q 3. Define a trigger to update a separate table (a new table – anything other than the customer table), after each insertion into the original customer table (i.e. whenever a new entry is added into the customer table, all the changes should be reflected in the new table). The separate table is user-defined, you can name it

according to your convenience.

Q4. Write and execute a DELETE statement that tries to delete the last salesman's all detail (tuple) from the salesman. Check the remaining rows in the salesman table. Is the salesman you tried to delete still in salesman and customer table? Why?