CS 207: Applied Database Practicum
Week 6

Varun Dutt

School of Computing and Electrical Engineering
School of Humanities and Social Sciences
Indian Institute of Technology Mandi, India

Scaling the Heights

MySQL - Triggers

- A SQL trigger is a set of SQL statements stored in the database catalog. A SQL trigger is executed or fired whenever an event associated with a table occurs e.g., **insert**, **update** or **delete**.
- A SQL trigger is a special type of stored procedure. It is special because it is not called directly like a stored procedure. The main difference between a trigger and a stored procedure is that a trigger is called automatically when a data modification event is made against a table whereas a stored procedure must be called explicitly.

Advantages of triggers

- Provides an alternative way to check the integrity of data.
- Provide a way to automatically run scheduled tasks because triggers are invoked automatically before or after a change is made in the table.
- Very useful to audit the changes in the table.
- Provide transparent event logging
- Prevent invalid transactions.

Types of triggers

A trigger can be defined to be invoked either before or after the data is changed by INSERT, UPDATE or DELETE statement.

- BEFORE INSERT - activated before data is inserted into the table.
- AFTER INSERT - activated after data is inserted into the table.
- BEFORE UPDATE - activated before data in the table is updated.
- AFTER UPDATE - activated after data in the table is updated.

Types of Triggers

- BEFORE DELETE - activated before data is removed from table.
- AFTER DELETE - activated after data is removed from table.

Syntax for Triggers

CREATE
[DEFINER = { *user* | CURRENT_USER }] (Optional)
TRIGGER *trigger_name*
trigger_time trigger_event
ON table_name FOR EACH ROW
trigger_body

trigger_time: { BEFORE | AFTER }

trigger_event: { INSERT | UPDATE | DELETE }

Within the trigger body, the OLD and NEW keywords enable you to access columns in the rows affected by a trigger. OLD and NEW are not case-sensitive.

Example on triggers

Tables description-

```
mysql> desc blog;
+----------+------------+------+-----+-------------------+-----------------------------+
| Field    | Type       | Null | Key | Default           | Extra                       |
+----------+------------+------+-----+-------------------+-----------------------------+
| id       | int(8)     | NO   | PRI | NULL              | auto_increment              |
| title    | text       | YES  |     | NULL              |                             |
| content  | text       | YES  |     | NULL              |                             |
| deleted  | tinyint(1) | NO   |     | 0                 |                             |
| blogTime | timestamp  | NO   |     | CURRENT_TIMESTAMP | on update CURRENT_TIMESTAMP |
+----------+------------+------+-----+-------------------+-----------------------------+
5 rows in set (0.00 sec)

mysql> desc audit;
+------------+-----------------------------+------+-----+-------------------+-----------------------------+
| Field      | Type                        | Null | Key | Default           | Extra                       |
+------------+-----------------------------+------+-----+-------------------+-----------------------------+
| id         | int(8)                      | NO   | PRI | NULL              | auto_increment              |
| blog_id    | int(8)                      | NO   | MUL | NULL              |                             |
| changetype | enum('NEW','EDIT','DELETE') | NO   |     | NULL              |                             |
| changetime | timestamp                   | NO   |     | CURRENT_TIMESTAMP | on update CURRENT_TIMESTAMP |
+------------+-----------------------------+------+-----+-------------------+-----------------------------+
4 rows in set (0.00 sec)

mysql>
```

Example using after-insert

```
ysql> delimiter $$
ysql> create
    ->   trigger blog_after_ins after insert
    ->   on blog
    ->   for each row begin
    ->     if NEW.deleted then
    ->       set @changetype = 'delete';
    ->     else
    ->       set @changetype = 'new';
    ->     end if;
    ->     insert into audit (blog_id, changetype) values (NEW.id, @changetype);
    ->   end$$
uery OK, 0 rows affected (0.09 sec)

ysql> delimiter ;
ysql> insert into blog (title,content) values ("after-insert trigger eg", "compare the timestamp of corresponding entries in blog-audit table"
);
uery OK, 1 row affected (0.08 sec)

ysql> select * from blog;
+----+------------------------+-----------------------------------------------------------------------+---------+---------------------+
| id | title                  | content                                                               | deleted | blogTime            |
+----+------------------------+-----------------------------------------------------------------------+---------+---------------------+
|  2 | after-insert trigger eg | compare the timestamp of corresponding entries in blog-audit table |       0 | 2018-09-09 01:23:02 |
+----+------------------------+-----------------------------------------------------------------------+---------+---------------------+
1 row in set (0.00 sec)

ysql> select * from audit;
+----+---------+------------+---------------------+
| id | blog_id | changetype | changetime          |
+----+---------+------------+---------------------+
|  2 |       2 | NEW        | 2018-09-09 01:23:02 |
+----+---------+------------+---------------------+
1 row in set (0.00 sec)

ysql>
```

Example Explanation

In above example whenever some new entry is inserted in blog table, new entry in audit table gets create automatically to keep note the changetype and timestamp. Next example deal with same table and show how any modification in table blog causes new entry in table audit referring to corresponding modified entry.

Example of after-update

Creation of trigger

```
mysql> delimiter $$
mysql> create
    ->   trigger blog_after_upd after update
    ->   on blog
    ->   for each row begin
    ->     if new.deleted then
    ->       set @changetype = 'delete';
    ->     else
    ->       set @changetype = 'edit';
    ->     end if;
    ->
    ->
    ->     insert into audit (blog_id, changetype) values (NEW.id, @changetype);
    ->   end $$
Query OK, 0 rows affected (0.08 sec)

mysql> delimter ;
    ->
```

Example of after-update

Realization of trigger -

```
mysql> update blog set content = "Nice blog" where id = 2;
Query OK, 1 row affected (0.17 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from audit;
+----+---------+------------+---------------------+
| id | blog_id | changetype | changetime          |
+----+---------+------------+---------------------+
|  2 |       2 | NEW        | 2018-09-09 01:23:02 |
|  3 |       2 | EDIT       | 2018-09-09 01:35:29 |
+----+---------+------------+---------------------+
2 rows in set (0.00 sec)

mysql> select * from blog;
+----+--------------------------+-----------+---------+---------------------+
| id | title                    | content   | deleted | blogTime            |
+----+--------------------------+-----------+---------+---------------------+
|  2 | after-insert trigger eg | Nice blog |       0 | 2018-09-09 01:35:29 |
+----+--------------------------+-----------+---------+---------------------+
```

Above example shows the modification of entry in blog table causes new entry in audit table automatically.

Example on before delete

Using same table as above:

Trigger creation and realization -

```
mysql> delimiter $$
mysql> create
    ->   trigger blog_before_del before delete
    ->   on blog
    ->   for each row begin
    ->     delete from audit where audit.blog_id = OLD.id;
    ->   end $$
Query OK, 0 rows affected (0.08 sec)

mysql> delimiter ;
mysql> delete from blog where id=2;
Query OK, 1 row affected (0.06 sec)

mysql> select * from blog;
Empty set (0.00 sec)

mysql> select * from audit;
Empty set (0.00 sec)
```

Example

Above example shows how one can save him/her from error.

As per the example, before deleting entry of primary key, we have to be sure that we first delete foreign key entries of it in another table. So we can use trigger before -delete here, so before deleting primary key entry clear all entries of it as foreign key in another table.

Example on before insert

For next two examples we'll using following table:
Table description-

```
ysql> create table people (age int, name varchar(100));
uery OK, 0 rows affected (0.46 sec)

ysql> desc people;
+-------+--------------+------+-----+---------+-------+
| Field | Type         | Null | Key | Default | Extra |
+-------+--------------+------+-----+---------+-------+
| age   | int(11)      | YES  |     | NULL    |       |
| name  | varchar(100) | YES  |     | NULL    |       |
+-------+--------------+------+-----+---------+-------+
 rows in set (0.00 sec)
```

Example on before insert

Trigger creation and realization

```
ysql> delimiter $$
ysql> create
    ->   trigger agecheck before insert
    ->   on people
    ->   for each row begin
    ->     if NEW.age < 0 then
    ->       set NEW.age = 0;
    ->     end if;
    ->   end $$
uery OK, 0 rows affected (0.09 sec)

ysql> delimiter ;
ysql> insert into people values (-100,"J. Cena"), (30, "R. Reigns");
uery OK, 2 rows affected (0.07 sec)
ecords: 2  Duplicates: 0  Warnings: 0

ysql> select * from people;
------+-----------+
 age  | name      |
------+-----------+
    0 | J. Cena   |
   30 | R. Reigns |
------+-----------+
 rows in set (0.00 sec)

ysql>
```

Example on before insert

In above example upon insertion of invalid negative age, MySQL with triggers modify it to zero.

Example on before-update

Trigger creation and realization-

```
ysql> delimiter $$
ysql> create
    ->   trigger upd_ageCheck before update
    ->   on people
    ->   for each row begin
    ->     if NEW.age < 0 then
    ->       set NEW.age = 0;
    ->     end if;
    ->   end $$
uery OK, 0 rows affected (0.08 sec)

ysql> delimiter ;
ysql> select * from people;
------+-----------+
 age  | name      |
------+-----------+
    0 | J. Cena   |
   30 | R. Reigns |
------+-----------+
 rows in set (0.00 sec)

ysql> update people set age = -10 where name = "R. Reigns";
uery OK, 1 row affected (0.04 sec)
ows matched: 1  Changed: 1  Warnings: 0

ysql> select * from people;
------+-----------+
 age  | name      |
------+-----------+
    0 | J. Cena   |
    0 | R. Reigns |
------+-----------+
 rows in set (0.00 sec)
```

Example on before-update

In above eg. upon modification of entry in table people MySQL trigger checks if there is invalid modification (age < 0), set age to zero.

References

http://w3schools.com/
http://www.mysqltutorial.org/mysql-triggers.aspx