University of Central Punjab

(Incorporated by Ordinance No. XXIV of 2002 promulgated by Government of the Punjab)

FACULTY OF INFORMATION TECHNOLOGY

Introduction to Database System

Lab 15

Topic : Triggers and its types

Spring-20

Read Instructions Carefully :

1-You need to run all queries one by one on your system.

2-Each query should have a screenshot of output.

3-Write Difference between Before and After insert , update and Delete in your own words.

4-You will not get any marks if you will copy from anywhere.

Different types of MySQL Triggers (with examples)

A MySQL **trigger** is a stored program (with queries) which is executed automatically to respond to a specific event such as insertion, updation or deletion occurring in a table. There are 6 different types of triggers in MySQL:

Syntax:

```
create trigger [trigger_name]
[before | after]
{insert | update | delete}
on [table_name]
[for each row]
[trigger_body]
```

Explanation of syntax:

1. create trigger [trigger_name]: Creates or replaces an existing trigger with the trigger_name.
2. [before | after]: This specifies when the trigger will be executed.
3. {insert | update | delete}: This specifies the DML operation.
4. on [table_name]: This specifies the name of the table associated with the trigger.
5. [for each row]: This specifies a row-level trigger, i.e., the trigger will be executed for each row being affected.
6. [trigger_body]: This provides the operation to be performed as trigger is fired

Note: Please attempt and understand all questions carefully.

Q#1(a). Before Update Trigger:

As the name implies, it is a trigger which enacts before an update is invoked. If we write an update statement, then the actions of the trigger will be performed before the update is implemented.

Example:

Considering tables:

```
create table customer (acc_no integer primary key,cust_name varchar(20),
                                    avail_balance decimal);
```

```
create table mini_statement (last_update datetime, acc_no integer,
avail_balance decimal, foreign key(acc_no) references customer(acc_no) on
delete cascade);
```

Inserting values in them:

```
insert into customer values (1000, "Fanny", 7000);

insert into customer values (1001, "Peter", 12000);
```

Trigger to insert (old) values into a mini_statement record (including account number and available balance as parameters) before updating any record in customer record/table:

```
delimiter //
create trigger update_cus
before update
on
customer
for each row
begin
insert into mini_statement values (CURRENT_TIMESTAMP(),
old.acc_no,old.avail_balance);
end;
//
```

Making updates to invoke trigger:

```
Delimiter ;
update customer set avail_balance = avail_balance + 3000 where acc_no =
1001;
update customer set avail_balance = avail_balance + 3000 where acc_no =
1000;
```

Output:

```
select *from mini_statement;

+--------+---------------+

| acc_no | avail_balance |

+--------+---------------+

|   1001 |         12000 |

|   1000 |          7000 |

+--------+---------------+

2 rows in set (0.0007 sec)
```

Q#1(b). After Update Trigger:

As the name implies, this trigger is invoked after an update occurs. (i.e., it gets implemented after an update statement is executed.).

Example:

We create another table:

```
create table micro_statement (last_update datetime,acc_no integer,
avail_balance decimal,foreign key(acc_no) references customer(acc_no) on
delete cascade);
```

Insert another value into customer:

```
insert into customer values (1002, "Janitor", 4500);

Query OK, 1 row affected (0.0786 sec)
```

Trigger to insert (new) values of account number and available balance into micro_statement record after an update has occurred:

```
delimiter //
create trigger update_after
after update
on
customer
for each row
begin
insert into micro_statement values(CURRENT_TIMESTAMP(), new.acc_no,
new.avail_balance);
end;
//
```

Making an update to invoke trigger:

```
delimiter ;

update customer set avail_balance = avail_balance + 1500 where acc_no =
1002;
```

Output:

```
select *from micro_statement;

+--------+---------------+

| acc_no | avail_balance |

+--------+---------------+

|   1002 |          6000 |

+--------+---------------+
```

```
1 row in set (0.0007 sec)
```

Q#1(c) Explain in your own words Difference between Before update and After update (Minimum 5 - 10 lines) ?

Q#2(a) Before Insert Trigger:

As the name implies, this trigger is invoked before an insert, or before an insert statement is executed.

Example:

Considering tables:

```
create table contacts (contact_id INT (11) NOT NULL AUTO_INCREMENT,
last_name VARCHAR (20) NOT NULL, first_name VARCHAR (25),birthday DATE,
created_date DATE, created_by VARCHAR(20), CONSTRAINT contacts_pk PRIMARY
KEY (contact_id));
```

Trigger to insert contact information such as name, birthday and creation-date/user into a table contact before an insert occurs:

```
delimiter //
create trigger contacts_before_insert
before insert
on
contacts
for each row
begin
DECLARE vUser varchar(50);
-- Find username of person performing INSERT into table
select USER() into vUser;
-- Update create_date field to current system date
SET NEW.created_date = SYSDATE();
-- Update created_by field to the username of the person performing the
INSERT
SET NEW.created_by = vUser;
end;
//
```

Making an insert to invoke the trigger:

```
Delimiter ;
insert into contacts values (1, "Newton", "Enigma",
str_to_date ("19-08-1999", "%d-%m-%Y"),
str_to_date ("17-03-2018", "%d-%m-%Y"), "xyz");
```

Output:

```
select *from contacts;

+------------+----------+-----------+----------+-------------+----------+

| contact_id |last_name |first_name | birthday |created_date |ceated_by |
```

```
+--------- --+----------+-----------+----------+-------------+----------+
|          1 | Newton   | Enigma    |1999-08-19| 2019-05-11  |
root@localhost |
+------------+-----------+------------+------------+--------------+-------
---------+
```

Q#2(b) After Insert Trigger:

As the name implies, this trigger gets invoked after an insert is implemented.

Example:

Consider tables:

```
create table contacts1 (contact_id int (11) NOT NULL AUTO_INCREMENT,
last_name VARCHAR(30) NOT NULL, first_name VARCHAR(25), birthday DATE,
CONSTRAINT contacts_pk PRIMARY KEY (contact_id));

create table contacts1_audit (contact_id integer,created_date date,
created_by varchar (30));
```

Trigger to insert contact_id and contact creation-date/user information into contacts_audit record after an insert occurs:

```
delimiter //

create trigger contacts_after_insert
after insert
on
contacts1
for each row
begin
DECLARE vUser varchar(30);
-- Find username of person performing the INSERT into table
SELECT USER() into vUser;
-- Insert record into audit table
INSERT into contacts1_audit ( contact_id, created_date, created_by)
VALUES ( NEW.contact_id, SYSDATE(), vUser );
END;
//
```

Making an insert to invoke the trigger:

```
insert into contacts1 values (1, "Asif", "Majeed",

str_to_date("20-06-1999", "%d-%m-%Y"));
```

Output:

```
select *from contacts_audit;

+------------+-------------+----------------+
```

```
| contact_id | created_date | created_by     |
+------------+--------------+----------------+
|          1 | 2019-05-11   | root@localhost |
+------------+--------------+----------------+
1 row in set (0.0006 sec)
```

Q#2(c) Explain in your own words Difference between Before Insert and After Insert ? (Minimum 5 - 10 lines) ?

Q#3(a) Before Delete Trigger:

As the name implies, this trigger is invoked before a delete occurs, or before the deletion statement is implemented.

Example:

Consider tables:

```
create table contacts (contact_id int (11) NOT NULL AUTO_INCREMENT,
last_name VARCHAR (30) NOT NULL, first_name VARCHAR (25),
birthday DATE, created_date DATE, created_by VARCHAR(30),
CONSTRAINT contacts_pk PRIMARY KEY (contact_id));

create table contacts_audit (contact_id integer, deleted_date date,
deleted_by varchar(20));
```

Trigger to insert contact_id and contact deletion-date/user information into contacts_audit record before a delete occurs:

```
delimiter //
create trigger contacts_before_delete
before delete
on
contacts
for each row
begin
DECLARE vUser varchar(50);
-- Find username of person performing the DELETE into table
SELECT USER() into vUser;
-- Insert record into audit table
INSERT into contacts_audit ( contact_id, deleted_date, deleted_by)
VALUES ( OLD.contact_id, SYSDATE(), vUser );
end;
//
```

Making an insert and then deleting the same to invoke the trigger:

```
delimiter;
```

```
insert into contacts values (1, "Bond", "Ruskin",
str_to_date ("19-08-1995", "%d-%m-%Y"),
str_to_date ("27-04-2018", "%d-%m-%Y"), "xyz");
delete from contacts where last_name="Bond";
```

Output:

```
select *from contacts_audit;

+------------+--------------+----------------+

| contact_id | deleted_date | deleted_by     |

+------------+--------------+----------------+

|          1 | 2019-05-11   | root@localhost |

+------------+--------------+----------------+

1 row in set (0.0007 sec)
```

Q#3(b). After Delete Trigger:

As the name implies, this trigger is invoked after a delete occurs, or after a delete operation is implemented.

Example:

Consider the tables:

```
create table contacts (contact_id int (11) NOT NULL AUTO_INCREMENT,
last_name VARCHAR (30) NOT NULL, first_name VARCHAR (25),
birthday DATE, created_date DATE, created_by VARCHAR (30),
CONSTRAINT contacts_pk PRIMARY KEY (contact_id));
create table contacts_audit (contact_id integer, deleted_date date,
deleted_by varchar(20));
```

Trigger to insert contact_id and contact deletion-date/user information into contacts_audit record after a delete occurs:

```
create trigger contacts_after_delete
after delete
on contacts
for each row
begin
DECLARE vUser varchar(50);
-- Find username of person performing the DELETE into table
SELECT USER() into vUser;
-- Insert record into audit table
INSERT into contacts_audit ( contact_id, deleted_date, deleted_by)
VALUES ( OLD.contact_id, SYSDATE(), vUser );
end;
//
```

Making an insert and deleting the same to invoke the trigger:

```
delimiter;
insert into contacts values (1, "Newton", "Isaac",
str_to_date ("19-08-1985", "%d-%m-%Y"),
str_to_date ("23-07-2018", "%d-%m-%Y"), "xyz");
delete from contacts where first_name="Isaac";
```

Output:

```
select *from contacts_audit;

+------------+--------------+----------------+

| contact_id | deleted_date | deleted_by     |

+------------+--------------+----------------+

|          1 | 2019-05-11   | root@localhost |

+------------+--------------+----------------+

1 row in set (0.0009 sec)
```

Q#3(c) Explain in your own words Difference between Before Delete and After Delete ? (Minimum 5 - 10 lines) ?