TRIGGERS

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.

Contd..

```
create table customer (acc_no integer primary key,
                       cust_name varchar(20),
                       avail_balance decimal);
create table mini_statement (acc_no integer, avail_balance
                             decimal,foreign key(acc_no)
                             references customer(acc_no)
                             on delete cascade);
```

Contd..

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:

Contd..

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

Contd..

```
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)
```

After Update Trigger:

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

Contd..

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


insert into customer values (1002, "Janitor", 4500);
Query OK, 1 row affected (0.0786 sec)
```

Contd..

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(new.acc_no,new.avail_balance);
    -> end; //
```

Contd..

delimiter ;

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

select *from micro_statement;

Output:

```
+--------+--------------+
| acc_no | avail_balance |
+--------+--------------+
|   1002 |         6000 |
+--------+--------------+
```

Before Insert Trigger:

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

```
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), PRIMARY KEY (contact_id));
```

Contd..

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

Contd..

```
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; //
```

Contd..

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

After Insert Trigger:

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

Contd..

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

Contd..

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

Contd..

```
delimiter //
create trigger contacts_after_insert
        -> after insert
        -> on contacts for each row
        -> begin
        ->    DECLARE vUser varchar(50);
        ->    -- Find username of person performing the INSERT into table
        ->    SELECT USER() into vUser;
        ->    -- Insert record into audit table
        ->    INSERT into contacts_audit ( contact_id, created_date, created_by) VALUES
              ( NEW.contact_id, SYSDATE(),vUser );
        -> END; //
```

Contd..

```
insert into contacts values (1, "Kumar", "Rupesh",
                        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)
```

Before Delete Trigger:

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

Contd..

```
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:

Contd..

```
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; //
```

Contd..

```
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";
```

Contd..

```
select *from contacts_audit;
+-----------+-------------+---------------+
| contact_id | deleted_date | deleted_by     |
+-----------+-------------+---------------+
|          1 | 2019-05-11   | root@localhost |
+-----------+-------------+---------------+
1 row in set (0.0007 sec)
```

After Delete Trigger:

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

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

Contd..

```
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; //
```

Contd..

```
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";
```

Contd..

```
select *from contacts_audit;
+-----------+-------------+---------------+
| contact_id | deleted_date | deleted_by     |
+-----------+-------------+---------------+
|          1 | 2019-05-11   | root@localhost |
+-----------+-------------+---------------+
1 row in set (0.0009 sec)
```