IPN, ESCOM, ING. EN SISTEMAS COMPUTACIONALES

MATERIA: DISTRIBUTED DATA BASES

GRUPO: 3CM10

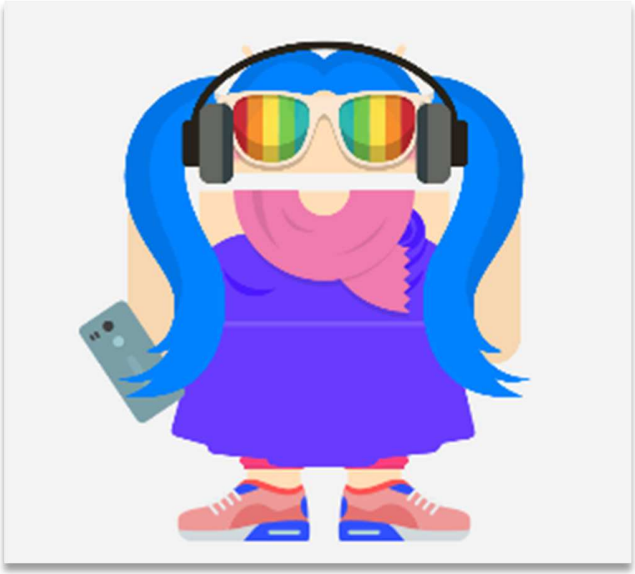
PROFESOR: HERNÁNDEZ CONTRERAS EULER

NOMBRE: SALDAÑA AGUILAR GABRIELA

NO. DE PRÁCTICA: 7

PRACTICA: TRIGGERS

FECHA DE ENTREGA: 12/05/2016



INDICE

MARCO TEORICO

The MySQL trigger is a database object that is associated with a table. It will be activated when a defined action is executed for the table. The trigger can be executed when you run one of the following MySQL statements on the table: INSERT, UPDATE and DELETE. It can be invoked before or after the event.

Some uses for triggers are to perform checks of values to be inserted into a table or to perform calculations on values involved in an update. These row operations are trigger events. For example, rows can be inserted by INSERT or LOAD DATA statements, and an insert trigger activates for each inserted row. A trigger can be set to activate either before or after the trigger event. For example, you can have a trigger activate before each row that is inserted into a table or after each row that is updated.

MySQL triggers activate only for changes made to tables by SQL statements. They do not activate for changes in views, nor by changes to tables made by APIs that do not transmit SQL statements to the MySQL Server.

SINTAXIS

```
CREATE [DEFINER={usuario|CURRENT_USER}]
TRIGGER nombre_del_trigger {BEFORE|AFTER} {UPDATE|INSERT|DELETE}
ON nombre_de_la_tabla
FOR EACH ROW
<bloque_de_instrucciones>
```

- DEFINER={usuario|CURRENT_USER}
  : Indica al **gestor de bases de datos** qué usuario tiene privilegios en su cuenta, para la invocación de los triggers cuando surjan los eventos **DML**. Por defecto este

característica tiene el valor CURRENT_USER que hace referencia al usuario actual que esta creando el Trigger.

- nombre_del_trigger:
  Indica el nombre de nuestro trigger. Existe una **nomenclatura** muy práctica para nombrar un trigger, la cual nos da mejor legibilidad en la administracion de la base de datos. Primero ponemos el nombre de tabla, luego especificamos con la inicial de la operación DML y seguido usamos la inicial del momento de ejecución(AFTER o BEFORE). Por ejemplo:

```
-- BEFORE INSERT
clientes_BI_TRIGGER
```

- BEFORE|AFTER: Especifica si el Trigger se ejecuta antes o después del evento DML.
- UPDATE|INSERT|DELETE:
  Aquí eliges que sentencia usarás para que se ejecute el Trigger.
- ON nombre_de_la_tabla:
  En esta sección estableces el nombre de la tabla asociada.
- FOR EACH ROW: Establece que el Trigger se ejecute por cada fila en la tabla asociada.
- <bloque_de_instrucciones>: Define el bloque de sentencias que el Trigger ejecutará al ser invocado.

Identificadores NEW Y OLD En Triggers

Si queremos relacionar el trigger con columnas especificas de una tabla debemos usar los identificadores OLD y NEW.
OLD indica el valor antiguo de la columna y NEW el valor nuevo que pudiese tomar. Por ejemplo: OLD.idproducto ó NEW.idproducto.
Si usamos la sentencia UPDATE podremos referirnos a un valor OLD y NEW, ya que modificaremos registros existentes por nuevos valores. En cambio si usamos INSERT solo usaremos NEW, ya que su naturaleza es únicamente de insertar nuevos valores a las columnas. Y si usamos DELETE usaremos OLD debido a que borraremos valores que existen con anterioridad.

Triggers BEFORE Y AFTER

Estas cláusulas indican si el Trigger se ejecuta **antes** o **después** del evento DML. Hay ciertos eventos que no son compatibles con estas sentencias.
Por ejemplo, si tuvieras un **Trigger AFTER** que se ejecuta en una sentencia UPDATE, sería ilógico editar valores nuevos NEW, sabiendo que el evento ya ocurrió. Igual sucedería con la sentencia INSERT, el Trigger tampoco podría referenciar valores NEW, ya que los valores que en algún momento fueron NEW, han pasado a ser OLD.

Para ver información sobre un trigger usamos:

SHOW CREATE TRIGGER

Para ver los triggers dentro de la BD:

SHOW TRIGGERS;

Para eliminar un trigger:

DROP TRIGGER nombre_trigger;

DESARROLLO

1.Construir las siguientes relaciones

```
create table uno(
        a1 int
);
create table dos(
        a2 int
);
create table tres(
        a3 int not null primary key auto_increment
);
create table cuatro(
        a4 int not null primary key auto_increment,
```

b1 int default 0

);

```
mysql> create database triggers;
Query OK, 1 row affected (0.00 sec)

mysql> use triggers;
Database changed
mysql> create table uno(
    -> a1 int
    -> );
Query OK, 0 rows affected (0.09 sec)

mysql> create table dos(
    -> a2 int
    -> );
Query OK, 0 rows affected (0.06 sec)

mysql> create table tres(
    -> a3 int not null primary key auto_increment
    -> );
Query OK, 0 rows affected (0.07 sec)

mysql> create table cuatro(
    -> a4 int not null primary key auto_increment,
    -> b1 int default 0
    -> );
Query OK, 0 rows affected (0.12 sec)
```

2.- Crear el siguiente trigger:

```
DELIMITER $
CREATE TRIGGER BI_uno
BEFORE INSERT ON uno
FOR EACH ROW
BEGIN
INSERT into dos
SET a2=NEW.a1;
DELETE from tres
WHERE a3=NEW.a1;
UPDATE cuatro
SET b1= b1+1
WHERE a4=NEW.a1;
END; $
DELIMITER ;
```

Este trigger genra una copia en A2 del nuevo valor a ser insertado en a1, vamos a borrar de tres los a3 que sean iguales al valor a ser insertado, en cuatro vamos a aumentar b1 donde a4 sea igual a a1.

3.- Llenar con datos la relacion tres

insert into tres

values(null),(null),(null),(null),(null)

,(null),(null),(null),(null),(null);

insert into cuatro (a4)

values(0),(0),(0),(0),(0)

,(0),(0),(0),(0),(0);

insert into uno values(1),(3),(1),(7),(8),(4),(4);

```
mysql> select * from tres;
+----+
| a3 |
+----+
|  1 |
|  2 |
|  3 |
|  4 |
|  5 |
|  6 |
|  7 |
|  8 |
|  9 |
| 10 |
+----+
10 rows in set (0.00 sec)

mysql> insert into cuatro (a4)
    -> values(0),(0),(0),(0),(0)
    -> ,(0),(0),(0),(0),(0);
Query OK, 10 rows affected (0.00 sec)
Enregistrements: 10  Doublons: 0  Avertissements: 0
```

```
mysql> select * from cuatro;
+----+------+
| a4 | b1   |
+----+------+
|  1 |    0 |
|  2 |    0 |
|  3 |    0 |
|  4 |    0 |
|  5 |    0 |
|  6 |    0 |
|  7 |    0 |
|  8 |    0 |
|  9 |    0 |
| 10 |    0 |
+----+------+
10 rows in set (0.00 sec)

mysql> insert into uno values(1),(3),(1),(7),(8),(4),(4);
Query OK, 7 rows affected (0.07 sec)
Enregistrements: 7  Doublons: 0  Avertissements: 0
```

```
mysql> insert into uno values(1),(3),(1),(7),(8),(4),(4);
Query OK, 7 rows affected (0.07 sec)
Enregistrements: 7  Doublons: 0  Avertissements: 0

mysql> select * from uno;
+------+
| a1   |
+------+
|    1 |
|    3 |
|    1 |
|    7 |
|    8 |
|    4 |
|    4 |
+------+
7 rows in set (0.00 sec)

mysql> select * from dos;
+------+
| a2   |
+------+
|    1 |
|    3 |
|    1 |
|    7 |
|    8 |
|    4 |
|    4 |
+------+
7 rows in set (0.00 sec)

mysql>
```

4.-Crear las siguientes relaciones

```
create table cliente(
        id int not null primary key auto_increment,
        nombreCliente varchar(100),
        depto varchar(20)
);


create table movCliente(

        id int not null primary key auto_increment,
        nombreCliente varchar(100),
        deptoMod varchar(40),
        fechaMod datetime,
        nombreUsuario varchar(30)
);
```

5.-Crear los siguientes triggers

```
create trigger  cliente_au
after update on cliente
for each row
        insert into movcliente
        (nombrecliente,deptoMod,fechaMod,nombreUsuario)
        values(old.nombreCliente,old.depto,now(), current_user());
```

6.-Llenar la relacion cliente.... con una BD la de homedepot

```
insert into cliente
(nombreCliente,depto)
select s.nombre,h.nombre
```

from home.socio s, home.homedepot h,

home.hdsocio x

where s.idsocio=x.socio_idsocio

and x.homedepot_idHd=h.idHD

order by s.nombre, h.nombre;

```
mysql> create table movCliente(
    ->
    -> id int not null primary key auto_increment,
    -> nombreCliente varchar(100),
    -> deptoMod varchar(40),
    -> fechaMod datetime,
    -> nombreUsuario varchar(30)
    -> );
Query OK, 0 rows affected (0.09 sec)

mysql> create trigger  cliente_au
    -> after update on cliente
    -> for each row
    -> insert into movcliente
    -> (nombrecliente,deptoMod,fechaMod,nombreUsuario)
    -> values(old.nombreCliente,old.depto,now(), current_user());
Query OK, 0 rows affected (0.02 sec)

mysql> insert into cliente
    -> (nombreCliente,depto)
    -> select s.nombre,h.nombre
    -> from home.socio s, home.homedepot h,
    -> home.hdsocio x
    -> where s.idsocio=x.socio_idsocio
    -> and x.homedepot_idHd=h.idHD
    -> order by s.nombre, h.nombre;
Query OK, 199 rows affected (0.18 sec)
Enregistrements: 199  Doublons: 0  Avertissements: 0
```

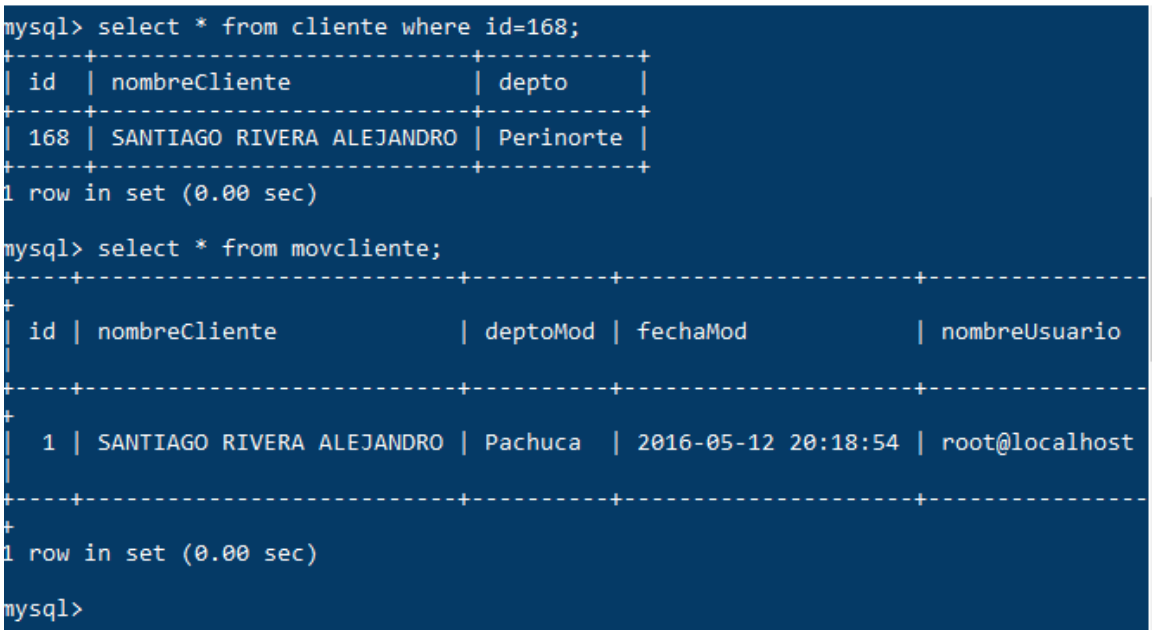
update cliente

set depto="Perinorte"

where id=168;

select * from cliente where id=168;

select * from movcliente;



```
mysql> select * from cliente where id=168;
+-----+---------------------------+-----------+
| id  | nombreCliente             | depto     |
+-----+---------------------------+-----------+
| 168 | SANTIAGO RIVERA ALEJANDRO | Perinorte |
+-----+---------------------------+-----------+
1 row in set (0.00 sec)

mysql> select * from movcliente;
+----+---------------------------+----------+---------------------+----------------
+
| id | nombreCliente             | deptoMod | fechaMod            | nombreUsuario
|
+----+---------------------------+----------+---------------------+----------------
+
|  1 | SANTIAGO RIVERA ALEJANDRO | Pachuca  | 2016-05-12 20:18:54 | root@localhost
|
+----+---------------------------+----------+---------------------+----------------
+
1 row in set (0.00 sec)

mysql>
```

CONCLUSIÓN

En esta práctica interactuamos de forma más dinámica con la BD, puesto que el usar triggers es trabajar con eventos que realizarán funciones específicas sobre las tablas de la BD, esto es de gran ayuda especialmente cuando hay que realizar operaciones que mantengan la integridad de la BD o que sea necesario obtener información.

BIBLIOGRAFÍA

http://www.hermosaprogramacion.com/2014/06/mysql-trigger/

http://dev.mysql.com/doc/refman/5.7/en/triggers.html

https://dev.mysql.com/doc/refman/5.5/en/trigger-syntax.html