

**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 TEÓRICO…………………………. 2

DESARROLLO………………………………… 4

CONCLUSIÓN Y BIBLIOGRAFÍA….. 10

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](http://dev.mysql.com/doc/refman/5.7/en/insert.html) or [LOAD DATA](http://dev.mysql.com/doc/refman/5.7/en/load-data.html) 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

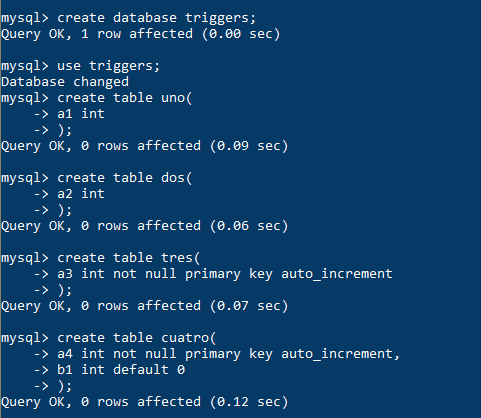
);

create table cuatro(

a4 int not null primary key auto\_increment,

b1 int default 0

);



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)

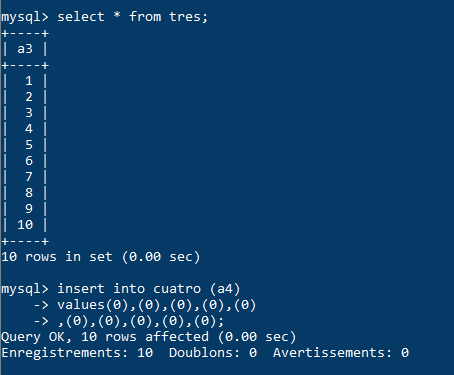
,(null),(null),(null),(null),(null);

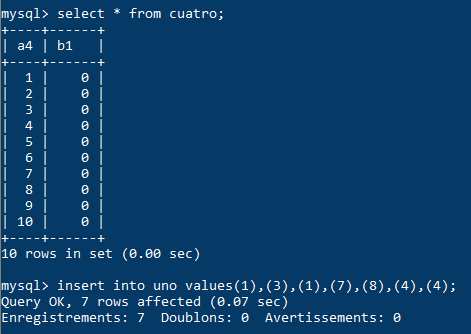
insert into cuatro (a4)

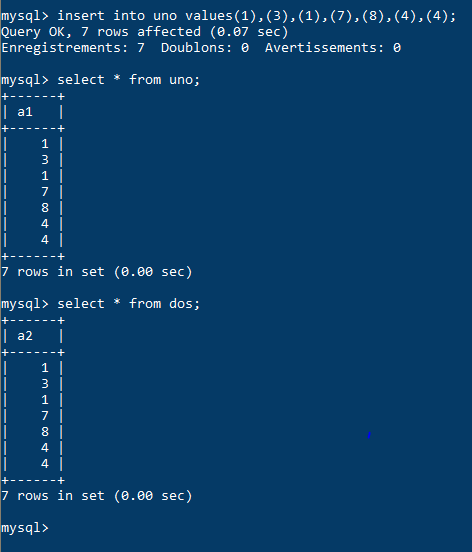
values(0),(0),(0),(0),(0)

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

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







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

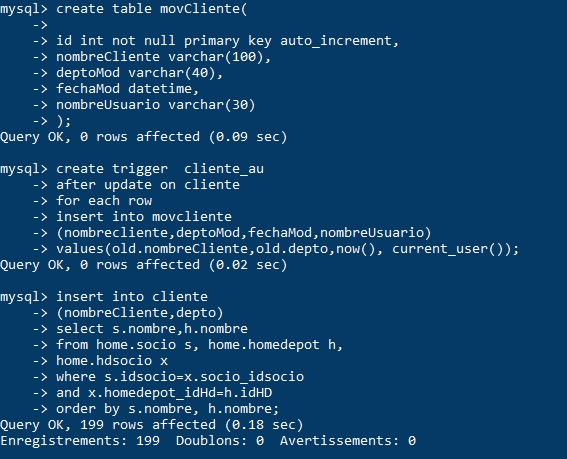
from home.socio s, home.homedepot h,

home.hdsocio x

where s.idsocio=x.socio\_idsocio

and x.homedepot\_idHd=h.idHD

order by s.nombre, h.nombre;



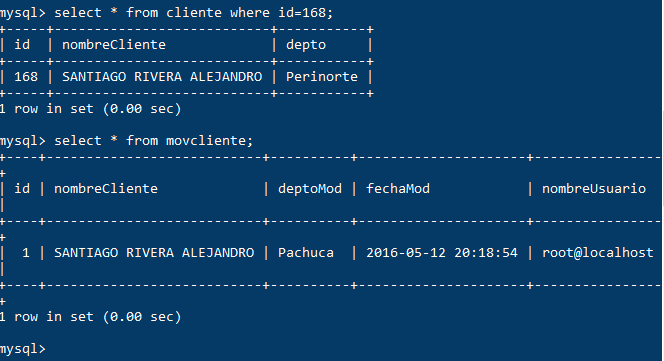
update cliente

set depto="Perinorte"

where id=168;

select \* from cliente where id=168;

select \* from movcliente;



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>