Instituto Politécnico Nacional

Escuela Superior de Cómputo

Distributed Data Base

Reporte de laboratorio #3.2

Nombre: De la rosa Hernández Samuel

Grupo: 3CM5

Fecha de entrega: 31/05/2019

Indice

Marco teorico

¿Qué son los triggers de MySQL y cómo se usan?

El trigger o disparador MySQL es un objeto de la base de datos que está asociado con tablas. El trigger se puede ejecutar cuando se ejecuta una de las siguientes sentencias MySQL en la tabla: INSERT, UPDATE y DELETE. Se puede invocar antes o después del evento.

es un objeto de base de datos con nombre que está asociado con una tabla y que se activa cuando ocurre un evento en particular para la tabla. Algunos usos de los activadores son realizar verificaciones de valores para insertarlos en una tabla o realizar cálculos en valores involucrados en una actualización.

Un activador se define para activarse cuando una declaración inserta, actualiza o elimina filas en la tabla asociada. Estas operaciones de fila son eventos de activación. Por ejemplo, las filas se pueden insertar mediante las instrucciones INSERT o LOAD DATA, y se activa un activador de inserción para cada fila insertada. Se puede configurar un activador para que se active antes o después del evento de activación. Por ejemplo, puede tener un activador de activación antes de cada fila que se inserta en una tabla o después de cada fila que se actualiza.

Instrucciones

cargar bd elektra.sql

trabajar con el cliente....

1. **modificar la pk... auto_increment;**

set foreign_key_checks=0;

alter table cliente modify column

idCliente int auto_increment;

```
mysql> alter table cliente modify column idcliente int auto_increment;
Query OK, 160 rows affected (0.35 sec)
Records: 160  Duplicates: 0  Warnings: 0
```

2. **crear la relacion socio ---> Socio(idsocio,nombre,ap,am);**

create table socio(

idSocio int not null primary key auto_increment,

nombre varchar(30),

ap varchar(30),

am varchar(30)

);

```
mysql> create table socio(
    -> idSocio int not null primary key auto_increment,
    -> nombre varchar(45),
    -> appaterno varchar(45),
    -> apmaterno varchar(45)
    -> );
Query OK, 0 rows affected (0.16 sec)
```

3. **Crear la relación empleado….**

create table empleado(

idempleado int not null primary key auto_increment,

nombre varchar(30),

ap varchar(30),

am varchar(30)

);

```
mysql> create table empleado(
    -> idEmpleado int not null primary key auto_increment,
    -> nombre varchar(45),
    -> appaterno varchar(45),
    -> apmaterno varchar(45)
    -> );
Query OK, 0 rows affected (0.16 sec)
```

4. **llenar la relacion empleado con los datos que tiene la relación cliente**

insert into empleado (nombre,ap,am)

select nombre, apPaterno,apMaterno

from Cliente;

```
mysql> insert into empleado (nombre,appaterno,apmaterno)select nombre,appaterno,apmaterno from cliente;
Query OK, 160 rows affected (0.09 sec)
Records: 160  Duplicates: 0  Warnings: 0
```

5. **crear un sp que permita dar de alta a un cliente determinado**

delimiter #

create procedure e1(in n varchar (40),

in p varchar(40),in m varchar(40))

begin insert into cliente (nombre,apPaterno,apMaterno) values (n,p,m);

select *from cliente

where nombre=n and apPaterno=p;

end #

```
mysql> delimiter #
mysql> create procedure e1(in ns varchar(45), in ap varchar(45), in am varchar(45))
    -> begin
    -> insert into cliente(nombre,appaterno,apmaterno) values(ns,ap,am);
    -> end #
Query OK, 0 rows affected (0.14 sec)

mysql> delimiter ;
```

6. Ahora, registramos un cliente llamando al SP e1: call e1("Euler","Hernandez","Contreras");

```
mysql> delimiter ;
mysql> call e1("Euler","Hernandez","Contreras");
Query OK, 1 row affected (0.13 sec)
```

7. Crea un disparador para reaizar las siguientes acciones.

A) Insertar el mismo registro (cliente) a la relación socio.
B) Actualizar la fecha de pago de un cliente.
C) Eliminar de la relación el mismo cliente de la relación empleado.

```
delimiter #
create trigger tgr1 after insert on cliente
for each row
insert into socio(nombre,appaterno,apmaterno)
values(NEW.nombre,NEW.appaterno,NEW.apmaterno);

update pago set fechapago=curdate() where idcliente= (NEW.idcliente);

delete from empleado where idEmpleado=(NEW.idcliente-100);

end #
delimiter ;
```

```
mysql> delimiter #
mysql> create trigger tgr1 after insert on cliente
    -> for each row
    -> insert into socio(nombre,appaterno,apmaterno) values(NEW.nombre,NEW.appaterno,NEW.apmaterno);
    ->
    -> update pago set fechapago=curdate() where idcliente= (NEW.idcliente);
    ->
    -> delete from empleado where idEmpleado=(NEW.idcliente-100);
    ->
    -> end #
Query OK, 0 rows affected (0.19 sec)

ERROR 1054 (42S22): Unknown column 'NEW.idcliente' in 'where clause'
mysql> delimiter ;
```

8. Ahora, llamamos a SP e1:

llamada e1 ("Samuel", "De la rosa", "Hernandez");

De nuevo, la relación socio se actualiza automáticamente.

9. **Cree la relación bitacora para conocer la vista previa y el valor actual de los cambios.**

```
create table biActualizar(
        id int not null primary key auto_increment,
        nombre varchar(45),
        valorprevio  date,
```

fechacambio datetime,

valornuevo date

);

```
mysql> create table biActualizar(
    -> id int not null primary key auto_increment,
    -> nombre varchar(45),
    -> valorprevio  date,
    -> fechacambio datetime,
    -> valornuevo date
    -> );
Query OK, 0 rows affected (0.16 sec)
```

10. crear trigger para llenar la bitacora

delimiter #

create trigger tgr3 AFTER UPDATE on pago

for each row

begin

insert into biActualizar(nombre,valorprevio,fechacambio,valornuevo)

values (current_user(),OLD.fechapago,now(),NEW.fechapago);

end #

delimiter ;

```
mysql> delimiter #
mysql> create trigger tgr3 AFTER UPDATE on pago
    -> for each row
    -> begin
    -> insert into biActualizar(nombre,valorprevio,fechacambio,valornuevo)
    -> values (current_user(),OLD.fechapago,now(),NEW.fechapago);
    -> end #
Query OK, 0 rows affected (0.16 sec)

mysql> delimiter ;
```

11. update pago set fechapago="2015-02-15" where idPago=1;

```
mysql> delimiter ;
mysql> update pago set fechapago="2015-02-15" where idPago=1;
Query OK, 1 row affected (0.11 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

Conclusiones

El uso de desencadenadores me parece muy útil, especialmente en el caso de que queremos llenar tablas sin tener que hacer una consulta adicional, principalmente como se vio en la práctica, para llenar la tabla de registros. De esta manera podemos controlar los cambios realizados en nuestra base de datos y, como vimos durante el curso, controlar las transacciones en una base de datos distribuida.

Referencias

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

https://www.siteground.es/kb/que-son-triggers-mysql-uso/

"MySQL 5.5 Reference Manual :: 20.3.1 Trigger Syntax and Examples" recovered on June 7, 2017 from https://dev.mysql.com/doc/refman/5.5/en/trigger-syntax.html

"Capitulo 20: disparadores" recovered on June 7, 2017 from https://manuales.guebs.com/mysql-5.0/triggers.html

"How to Create Database Triggers in MySQL" recovered on June 7, 2017 from https://manuales.guebs.com/mysql-5.0/triggers.html