GUIA DE TRIGGERS (*Disparadores*) EN MYSQL

Conceptos, historia y contexto

A partir de MySQL 5.0.2 se incorporó el soporte básico para disparadores (triggers). *Un disparador es un objeto con nombre dentro de una base de datos el cual se asocia con una tabla y se activa cuando ocurre en ésta un evento en particular.*

Definición de un trigger

```
CREATE TRIGGER nombre_disp momento_disp evento_disp
    ON nombre_tabla FOR EACH ROW sentencia_disp
```

Eliminación de un trigger

Drop trigger *nombre_disp*;

nombre_disp: será el nombre del trigger.

El disparador queda asociado a la tabla *nombre_tabla*. Esta debe ser una tabla permanente, no puede ser una tabla TEMPORARY ni una vista.

momento_disp es el momento en que el disparador entra en acción. Puede ser BEFORE (antes) o AFTER (despues), para indicar que el disparador se ejecute antes o después que la sentencia que lo activa.

evento_disp indica la clase de sentencia que activa al disparador. Puede ser INSERT, UPDATE, o DELETE. Por ejemplo, un disparador BEFORE para sentencias INSERT podría utilizarse para validar los valores a insertar.
No puede haber dos disparadores en una misma tabla que correspondan al mismo momento y sentencia. Por ejemplo, no se pueden tener dos disparadores BEFORE UPDATE. Pero sí es posible tener los disparadores BEFORE UPDATE y BEFORE INSERT o BEFORE UPDATE y AFTER UPDATE.

sentencia_disp es la sentencia que se ejecuta cuando se activa el disparador (lo que quiero que sea automático, es un algoritmo). Si se desean ejecutar múltiples sentencias, deben colocarse entre BEGIN ... END, el constructor de sentencias compuestas. Esto además posibilita emplear las mismas sentencias permitidas en rutinas(procedimientos y funciones) almacenadas.

En estas sentencias se puede tener acceso a los datos dependiendo del *evento_disp* y del *momento_disp*:

momento_disp	*evento_disp*	*Toma de datos*	*observaciones*
BEFORE	INSERT	New.nombredelcampo: campo nuevo Ejemplo dentro del trigger yo puedo hacer referencia al nuevo código así new.codigo	
AFTER	INSERT	New.nombredelcampo: campo nuevo	No se puede trabajar sobre la misma tabla
BEFORE	UPDATE	New.nombredelcampo: campo nuevo old.nombredelcampo: campo viejo. Ejemplo en el trigger se podría preguntar If(old.costo>new.costo) Se refiere a que compare el que está actualmente y el Nuevo por el que va a ser remplazado	
AFTER	UPDATE	New.nombredelcampo: campo nuevo old.nombredelcampo: campo viejo.	No se puede trabajar sobre la misma tabla

BEFORE	DELETE	old.nombredelcampo: campo viejo	
AFTER	DELETE	old.nombredelcampo: campo viejo	

Ejemplo1 de un triggers: La "idea" es crear un trigger que *antes* de *insertar* un *producto* calcule el valor de venta.

Supongamos la siguiente situación: En una empresa vende productos y se gana un porcentaje fijo por cada producto.

Cree la siguiente tabla:

PRODUCTOS
CODIGO NOMBRE PORCENTAJE_DE_GANANCIA COSTO VALOR VENTA CANTIDAD

PASO PARA EL EJEMPLO

```
-- Creando la base de datos
    Create database pruebadetrigger;
-- seleccionando la base de datos
    Use pruebadetrigger;
-- creando la base de datos
    Create table productos(
    codigo varchar(12) primary key,
    nombre varchar(30),
    porgana int(2),
    Costo int,
    Valorventa int,
    cantidad int
    );
```

-- Insertando información: observe que enviamos el valorventa en cero(0)

```
    Insert into productos values('1','arroz 1
    kilo',15,1800,0,45);
```

-- observamos lo que se guardo:

```
    select * from productos;
```

efectivamen guardo en cero(0) aun no hemos creado nada que me calcule el valor venta(ningún procedimierno, función o trigger)

-- **creemos el trigger**

```
Drop  trigger calvalorventa;
Delimiter //
Create trigger calvalorventa before insert  on productos
for each row
begin
set new.valorventa= new.costo +new.costo*new. Porgana/100;
end;
//
Delimiter ;
```

-- **eliminando e Insertando información: observe que enviamos el valorventa en cero(0)**

```
delete from productos;
Insert into productos values('1','arroz 1
kilo',15,1800,0,45);
```

-- **observamos lo que se guardo: ahora si calculo el valor de venta "automaticamente"**

```
select * from productos;
```

Ejemplo2 de un triggers: La "idea" es crear un trigger que *antes* de *actualizar* un *producto* vuelva a calcule el valor de venta.

PASOS CREANDO EL TRIGGER

-- **actualizando un producto**

```
Update productos set costo=5000 where codigo='1';
```

-- **observamos lo que se actualizo:**

```
select * from productos;
```

Efectivamente solo se actualizo el costo a 5000 y el valorventa sigue siendo 2070, aun no hemos creado nada que me calcule el valor venta cuando se actualice(ningún procedimierno, función o trigger)

-- **creemos el trigger**

```
Delimiter //
Create trigger  actvalorventa before update  on productos
for each row
begin
set new.valorventa= new.costo + new.costo*new.Porgana/100;
end;
//

Delimiter ;
```

-- eliminando e Insertando información y actualizamos: observe que enviamos el valorventa en cero(0)

```
delete from productos;
Insert into productos values('1','arroz 1
kilo',15,1800,0,45);
Update productos set costo=5000 where codigo='1';
```

Observamos lo que se actualizo:

```
select * from productos;
```

Ahora si volvió a calcular el valorventa "automaticamente".