**Ejercicios MySQL**

**1-Necesitamos almacenar en una tabla llamada "usuarios" los datos de los usuarios de un sitio web. Cada vez que el usuario cambia su clave se debe almacenar en otra tabla llamada "clavesanteriores" el dato de la clave vieja.**

**Crear estas tablas con las siguientes estructuras:**

create table usuarios(

nombre varchar(30),

clave varchar(30),

primary key (nombre)

);

create table clavesanteriores(

numero int auto\_increment,

nombre varchar(30),

clave varchar(30),

primary key (numero)

);

**Creamos el disparador 'before\_usuarios\_update'**

**Ejecutemos ahora un insert en la tabla 'usuarios':**

insert into usuarios(nombre,clave) values ('marcos','123abc');

El trigger no se ejecuta ya que solo hemos definido que se dispare para el comando 'update'.

**Ahora procedamos a modificar la clave del usuario mediante el comando 'update':**

update usuarios set clave='999zzz' where nombre='marcos';

Cuando se ejecuta el 'update' además de actualizarse la clave del usuario en la tabla 'usuarios' se dispara el trigger donde se efectúa la inserción de una fila en la tabla 'clavesanteriores'.

**Listemos la tabla 'clavesanteriores'**

select \* from clavesanteriores;

Podemos comprobar que tenemos ahora una fila que contiene los datos:

nombre: marcos

clave: 123abc

Si volvemos a cambiar la clave del usuario 'marcos':

update usuarios set clave='123456' where nombre='marcos';

Listamos nuevamente la tabla 'clavesanteriores':

select \* from clavesanteriores;

Podemos comprobar que tenemos ahora dos filas que contiene los datos:

nombre: marcos

clave: 123abc

nombre: marcos

clave: 999zzz

Con el trigger logramos tener el historial de todas las claves empleadas por los usuarios. Tener en cuenta que la última clave elegida por el usuario se encuentra en la tabla 'usuarios':

select \* from usuarios;

**2-Administrar los datos de dos tablas llamadas: "libros" y "ventas". Cada vez que se produzca la venta de libros reducir el campo stock de la tabla "libros" mediante un trigger definido en la tabla ventas**.

Creamos las dos tablas con las siguientes estructuras:

create table libros(

codigo int auto\_increment,

titulo varchar(50),

autor varchar(50),

editorial varchar(30),

precio float,

stock int,

primary key (codigo)

);

create table ventas(

numero int auto\_increment,

codigolibro int,

precio float,

cantidad int,

primary key (numero)

);

Insertamos algunas filas de prueba en la tabla "libros":

insert into libros(titulo, autor, editorial, precio, stock)

values('Uno','Richard Bach','Planeta',15,100);

insert into libros(titulo, autor, editorial, precio, stock)

values('Ilusiones','Richard Bach','Planeta',18,50);

insert into libros(titulo, autor, editorial, precio, stock)

values('El aleph','Borges','Emece',25,200);

insert into libros(titulo, autor, editorial, precio, stock)

values('Aprenda PHP','Mario Molina','Emece',45,200);

**Creamos el disparador 'before\_ventas\_insert':**

El disparador 'before\_ventas\_insert' se asocia a la tabla 'ventas' y se dispara cada vez que ejecutamos el comando SQL insert:

**Procedemos a insertar una fila en la tabla 'ventas':**

insert into ventas(codigolibro, precio, cantidad) values(1, 15, 1);

Luego si controlamos la cantidad de libros en stock veremos que se ha reducido en 1:

**3-Una librería almacena los datos de sus libros en una tabla denominada "libros" y en otra tabla llamada "ventas", las ventas de los mismos.**

Creamos las dos tablas con las siguientes estructuras:

create table libros(

codigo int auto\_increment,

titulo varchar(50),

autor varchar(50),

editorial varchar(30),

precio float,

stock int,

primary key (codigo)

);

create table ventas(

numero int auto\_increment,

codigolibro int,

precio float,

cantidad int,

primary key (numero)

);

Insertamos algunas filas de prueba en la tabla "libros":

insert into libros(titulo, autor, editorial, precio, stock)

values('Uno','Richard Bach','Planeta',15,100);

insert into libros(titulo, autor, editorial, precio, stock)

values('Ilusiones','Richard Bach','Planeta',18,50);

insert into libros(titulo, autor, editorial, precio, stock)

values('El aleph','Borges','Emece',25,200);

insert into libros(titulo, autor, editorial, precio, stock)

values('Aprenda PHP','Mario Molina','Emece',45,200);

**Creamos un disparador cuando se produce una venta para disminuir el stock de libros:**

**Creamos un nuevo disparador para actualizar el campo "stock" de la tabla "libros" cuando se elimina un registro de la tabla "ventas" (por ejemplo, si el comprador devuelve el o los libros comprados):**

Procedemos a efectuar una venta y luego controlar que se ha reducido en 1 el stock de dicho libro en la tabla 'libros':

insert into ventas(codigolibro, precio, cantidad) values(1, 15, 1);

select \* from libros;

Finalmente eliminamos la fila de la tabla 'ventas' por la devolución del libros, nuevamente podemos ver que gracias a la ejecución del trigger 'before\_ventas\_delete' se ha incrementado el stock en la tabla 'libros':

delete from ventas where numero=1;

select \* from libros;

**4-Creamos las siguiente tabla:**

CREATE TABLE empleados

(dni char(4) PRIMARY KEY,

Nomemp varchar2(15),

Mgr char(4),

Salario integer DEFAULT 1000,

Usuario integer DEFAULT 1000,

Fecha date );

ALTER TABLE EMP ADD(  
FOREIGN KEY (mgr) REFERENCES empleados (DNI) );

**- Crear un trigger sobre la tabla empleados para que no se permita que un empleado sea jefe (MGR) de más de cinco empleados:**

**- Crear un trigger para impedir que se aumente el salario de un empleado en más de un 20%.**

**Bibliografía:**

<https://help.clouding.io/hc/es/articles/4408406585618-C%C3%B3mo-administrar-usuarios-y-permisos-en-MySQL>

<https://www.studocu.com/co/document/universidad-del-pacifico-colombia/gestion-de-tecnologia/ejercicios-de-cuentas-de-usuario-resuelto/31421037>