**Estudiante:** Kevin Gracia Orejuela

**Nivel:** Quinto Semestre

**BASES DE DATOS II**

Actualizar la base classicmodels a nivel de BDD para obtener las siguientes funcionalidades:

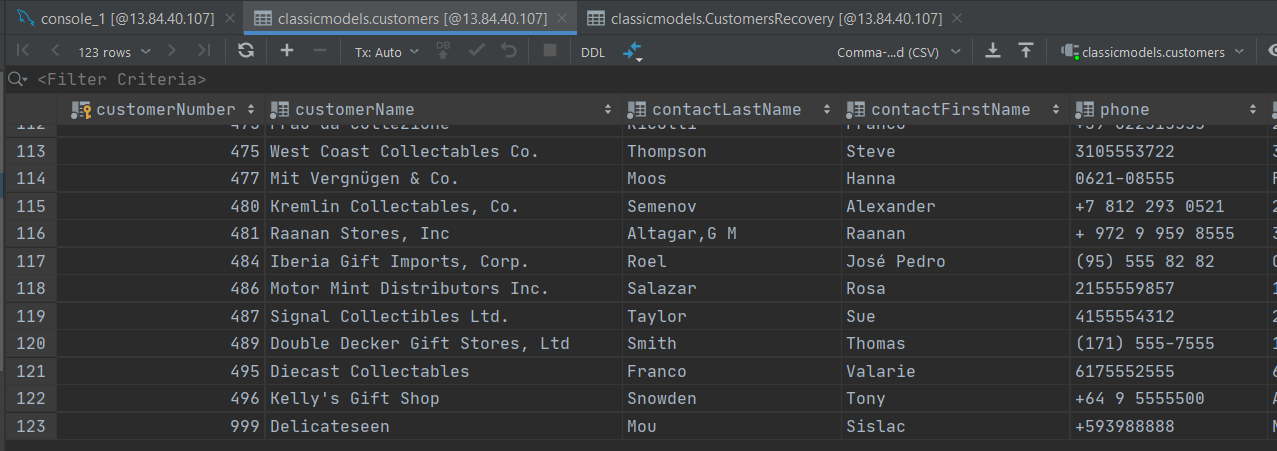
**1. Generar un registro en caso de que se elimina algo de las siguientes tablas: productlines, product, customers**

Para la solución a este requerimiento primero traté de hacer un solo disparador para las tres tablas, pero leyendo en distintos foros y documentación me encontré que un disparador únicamente puede estar asociado hacia una única tabla por lo que generé tres disparadores distintos para cada tabla, el proceso fue el siguiente:

* Crear una tabla con los mismos campos de la tabla a respaldar y con un campo adicional deleteAt para saber cuándo se borró algo, un nombre distinto en este caso use la palabra Recovery luego del nombre en la tabla original con formato UpperCamelCase
* Empezar la creación de trigger con los delimitadores, usar el naming adecuado para crearlo con formato SnakeCase: tiempodeaccion\_tabla\_accion
* Indicamos el tiempo y la acción del trigger en este caso BEFORE DELETE
* Luego la sentencia ON tabla\_ascoiada FOR EACH ROW
* BEGIN – inicio de las acciones a realizar
* INSERT INTO tabla\_creada\_paso1 (campos)
* VALUES usamos la palabra reservada OLD.campo\_tabla\_original
* END$$ – DELIMITER ;

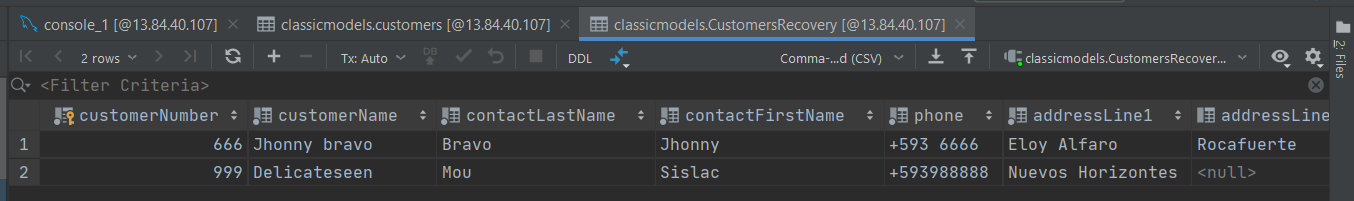
**Ejemplo**

Se encuentra el registro con el numero 999 en la tabla customer



DELETE FROM customers WHERE customerNumber=999

Si se ejecuta la sentencia de borrar se añade el registro a la tabla CustomersRecorey



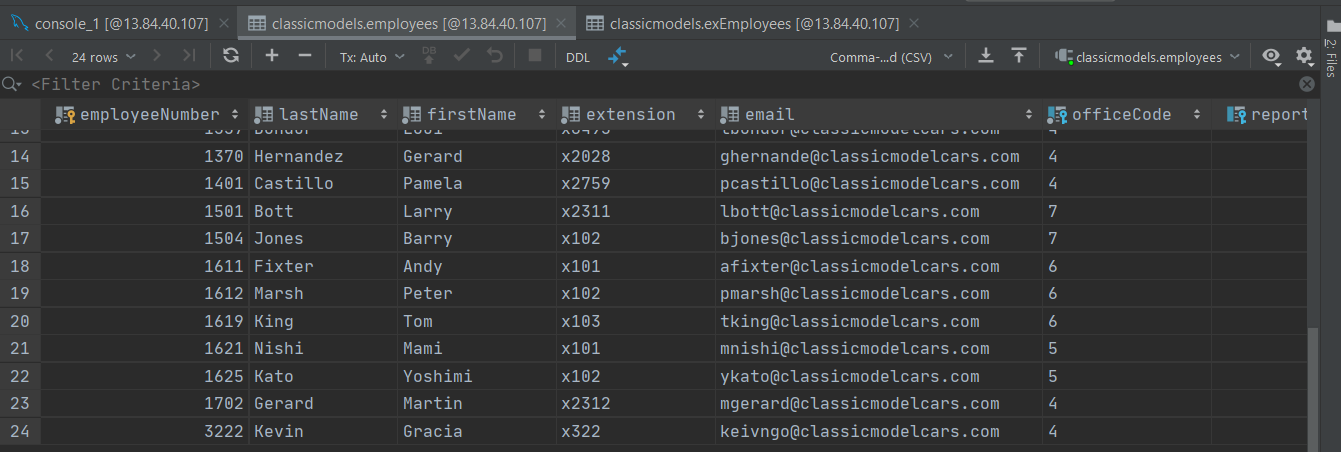
**Registros que se elimina en la tabla employees tiene que pasar a una tabla exEmployees al borrarlos**

Para cumplir con este requerimiento seguimos las *buenas prácticas* indicadas en el anterior y el proceso es similar

* Crear la tabla exEmployees con los mismos campos de employees añadiendo un campo deleteAt
* DELIMITER $$
* Creamos el trigger damos el nombre
* Indicamos el tiempo y acción BEFORE DELETE
* Sobre que tabla estará asociado ON employees FOR EACH ROW
* BEGIN – inicio de acciones
* Trigger de inserción INSERT INTO exEmployees(campos)
* VALUES (OLD.campo)
* END $$
* DELIMITER;

**Ejemplo**

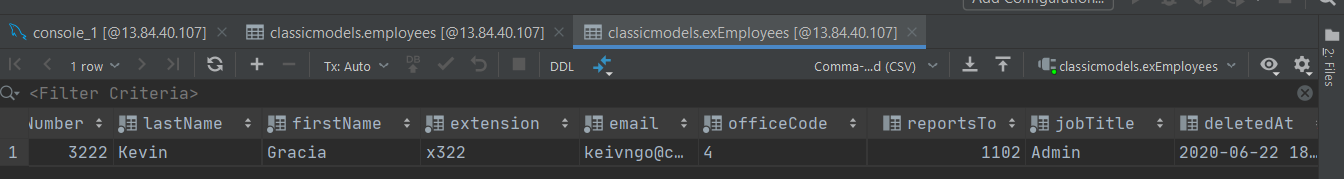
Tenemos al empleado con el numero 3222 en employes



Lo eliminamos

DELETE FROM employees WHERE employeeNumber=3222

Vemos que se almaceno en la tabla exEmployes luego de ser eliminado y se encuentra la fecha de cuando ocurrió esa eliminación



**Actualizaciones o inserts a la tabla products tiene que registrarse en una tabla logProducto**

Este requerimiento en mysql necesita ser manejado en disparadores diferentes debido a que no Podemos ejecutar en un solo disparador las dos acciones en este caso after insert y after update

* Crear la table logProductos con los campos: **id, productCode, dateLog** (fecha exacta del momento en que se ejecute la accion DEFAULT NOW(), **descripcion** (breve descripcon de la accion que se hizo)
* Creamos el trigger con el naming adecuado
* Establecemos la accion AFTER INSERT para el primer disparador y AFTER UPDATE Segundo disparador FOR EACH ROW
* BEGIN
* INSERT INTO logProducts
* VALUES usamos la function reservada CONCAT para formatear la **descripcion** e indicar la accion realizada
* En Segundo disparador(UPDATE) use condicionales para saber cual es el campo que se estaba actualizando y como cambiaba su valor todo esto Tambien lo envie al campo **descripcion** para que se diferencie el OLD.value a NEW.value
* END

**Ejemplo**

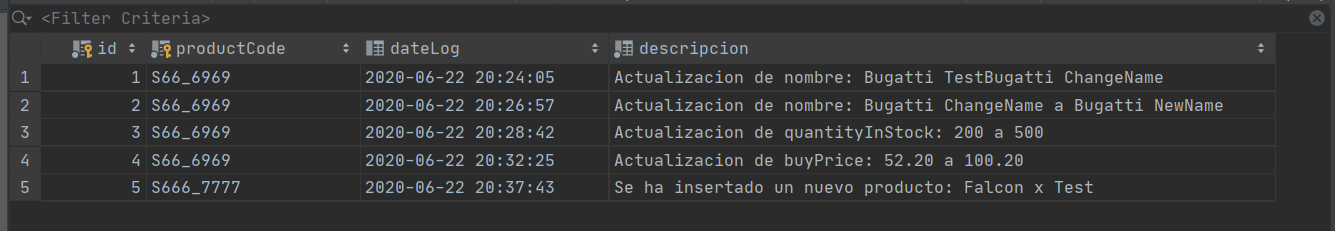
INSERT INTO products(productCode, productName, productLine, productScale, productVendor, productDescription, quantityInStock, buyPrice, MSRP)

VALUES("S66\_7777","Falcon X Test","Sport Cars",1:72,"Dubai AutoSports","Created in Dubai",7,250.25,260.30);

UPDATE products

SET buyPrice = 100.20

WHERE productCode = 'S66\_6969';



**Impedir que se puede borrar algo de las tablas orders y payments y registrar en caso que hay alteraciones a estas tablas - tambien registrar el usuario (employee) que está realizando cambios**

Esta solicitud la hice por partes primero para impeder el borrado de algun dato en la table orders y payments en donds disparadores distintos para los que el prcoeso fue el siguiente

* Se trata de un before delete
* Validamos que los campos no sean nulos
* Mandamos una señal donde se indica que hay un error

Registrar en caso de que hay alteracion es y el usuario employee que esta realizando las mismas

* Hice un after update para orders y para payments
* Primero creamos ambas tablas para almacenar dichos registros: logOrders y logPayments en esta vamos a llevar el campo primario de cada tabla, una fecha en que se hizo el registro **dateLog,** y una descripcion para definir la accion que se ejecuto y un campo **employesLog** donde almacenaernos el nombre del empleado que hizo los cambios
* Creamos el trigger con el naming correspondiente
* Definimos la accion AFTER UPDATE ON table\_correspondiente FOR EACH ROW
* BEGIN
* Realizamos condiciones para saber cual es el campo que se esta modificando comparando valores OLD & NEW
* INSERT INTO en **campoclave**, **descripcion** y employesLog
* VALUES OLD.**campoclave** , **descripcion** usamos CONCAT para establecer la diferencia entre el valor OLD & NEW y del campo donde se realizo la accion, **employesLog** ejecute una subconsulta de las distintas tablas con las que se puede establecer una relacion del **employee** con la **orders** pasando por la table **customers** su campo saleRepEmployeeNumbers

**SubConsulta**

SELECT lastName FROM employees, customers, orders

WHERE customers.salesRepEmployeeNumber = employees.employeeNumber

AND orders.customerNumber = customers.customerNumber

AND orders.orderNumber = OLD.orderNumber

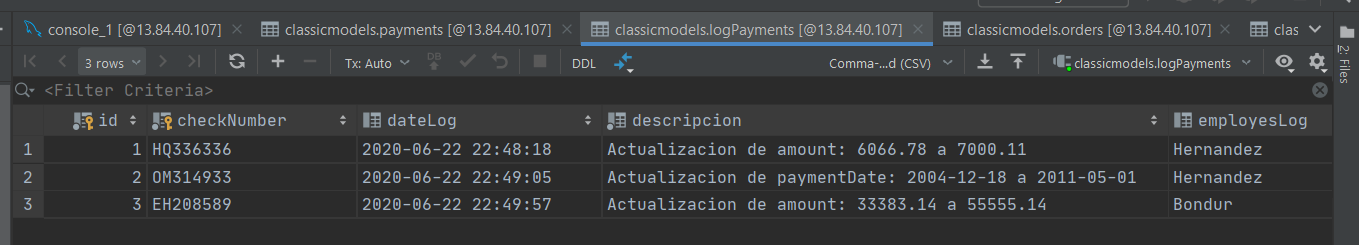
De esta forma obtenemos el employee que realizo ese cambio

**Ejemplo de logPayments**

UPDATE payments

SET amount = 7000.11

WHERE checkNumber = 'HQ336336’;



**Ejemplo de logOrders**

UPDATE orders

SET status = ‘Cancelled’

WHERE orderNumber = 10200;

