Coderhouse

Curso: SQL

Alumno: Tomas Rave

Tema: Departamento de Bromatología

Resumen

En este documento se detalla el modelo de negocio, situación problemática, el objetivo de este trabajo, así como también referente a la parte técnica, los Diagramas de Entidad-Relación, estructura de tablas, funciones, procedimientos, vistas y triggers creados/utilizados para solucionar los problemas listados.

Introducción

El modelo utilizado es el del Departamento de Bromatología de la Municipalidad de General Pueyrredón, Provincia de Buenos Aires, Argentina. Se trabajo con el área del curso de manipulación de alimentos, área de habilitación de vehículos que transportan alimentos, el área de inspección en vía publica y el área de habilitación de comercios que trafican alimentos.

Objetivo

Crear un sistema de base de datos para poder realizar un seguimiento del trabajo diario del departamento, que nos permita obtener métricas de control y seguimiento.

Problemática

La falta de datos y métricas para poder realizar un seguimiento eficiente del trabajo realizado, así como poder llevar auditorías internas.

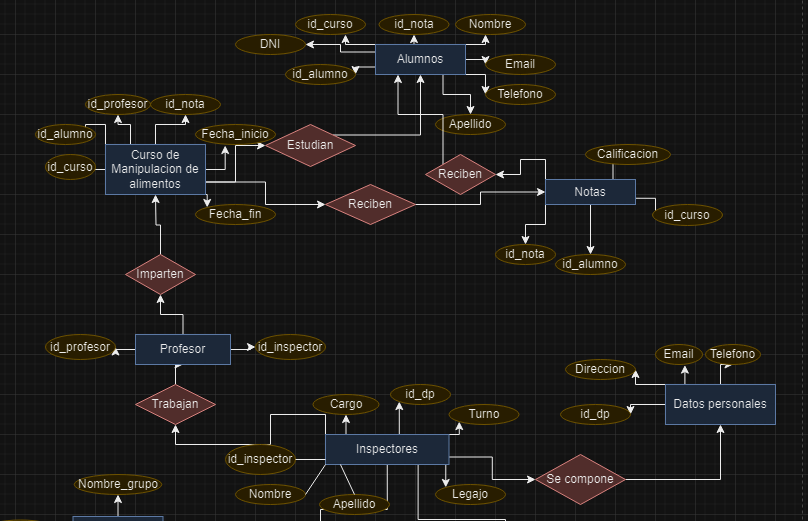
Modelo de negocio

El modelo elegido para este trabajo es una dependencia municipal que consta de la siguiente estructura.

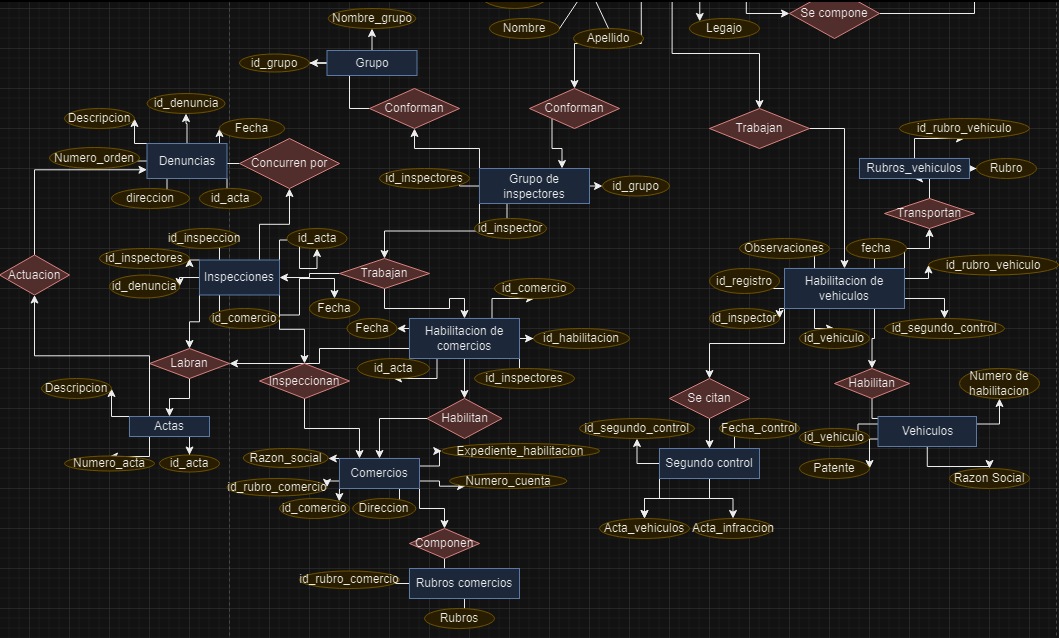
* Un (1) jefe de departamento.
* Trece (15) inspectores, de los cuales cuatro (4) están habilitados a dar el curso de manipulación de alimentos.
* Cuatro (4) administrativos.

El departamento abre de lunes a viernes de 8:00 a 22:00. Trabajando en dos turnos.

Diagrama entidad-relación (conceptual)



El resto del DER se une a través de la tabla inspectores y grupo de inspectores.



Tablas creadas por los triggers.

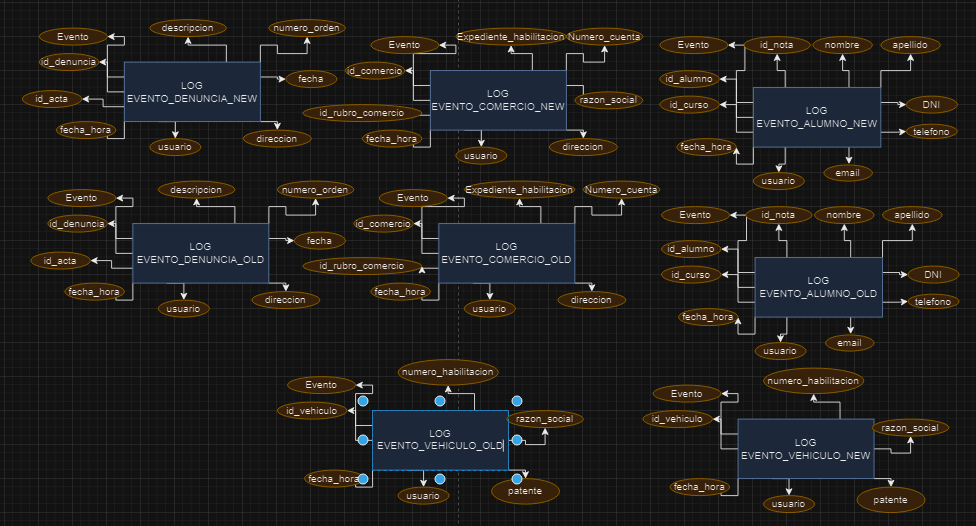
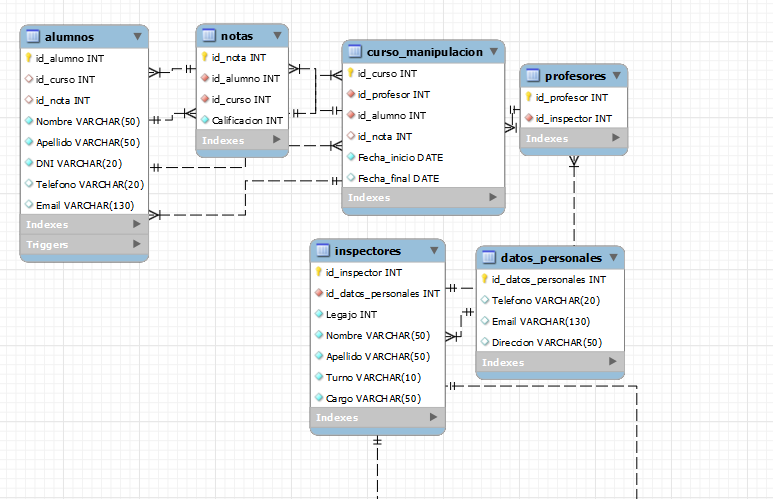
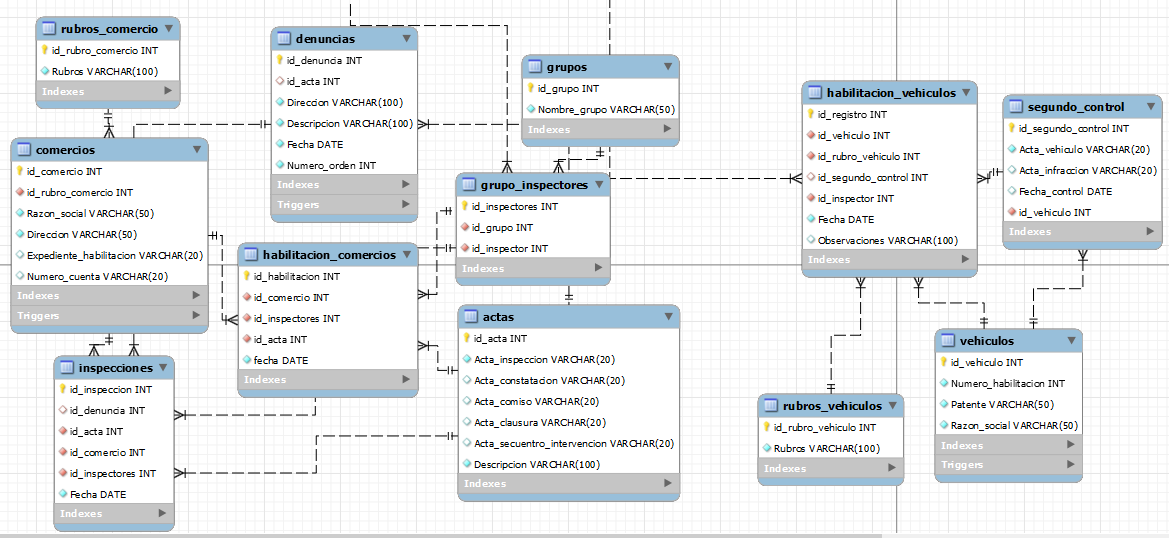


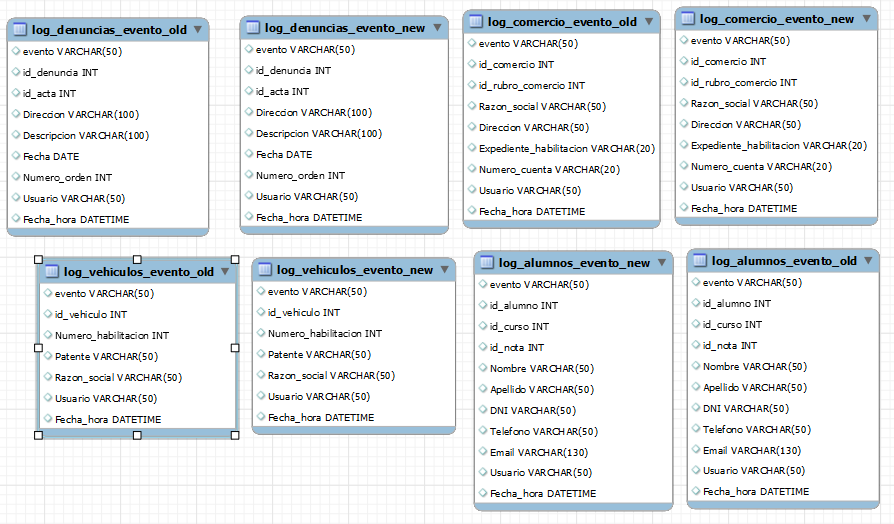
Diagrama entidad-relación (esquemático)



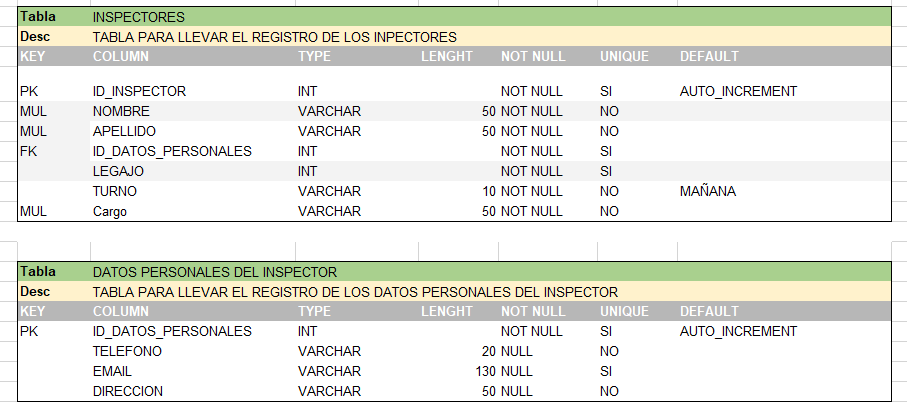
El resto del DER se une a través de la tabla inspectores y grupo de inspectores.

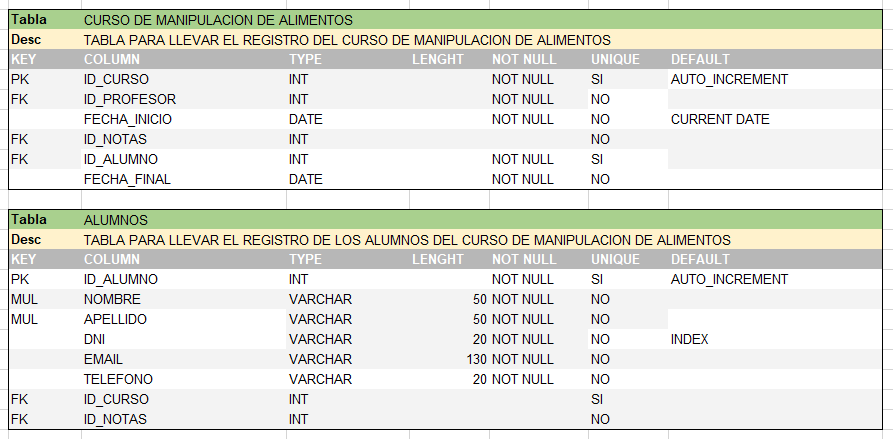


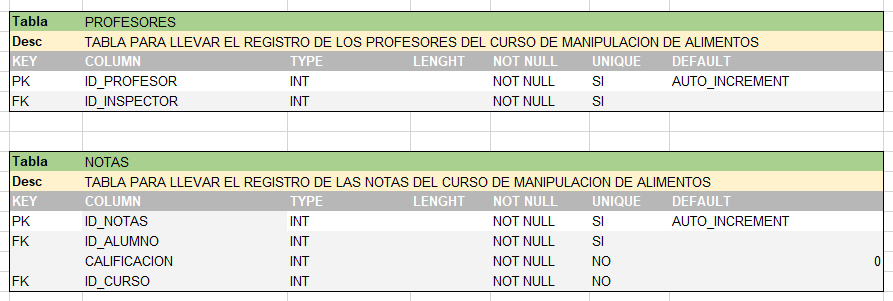
Tablas hechas para el seguimiento de los triggers.

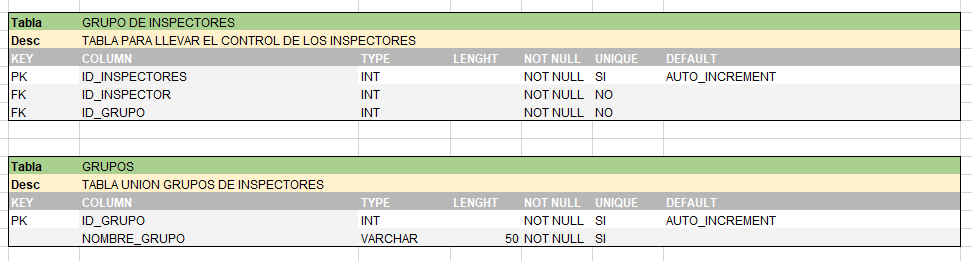


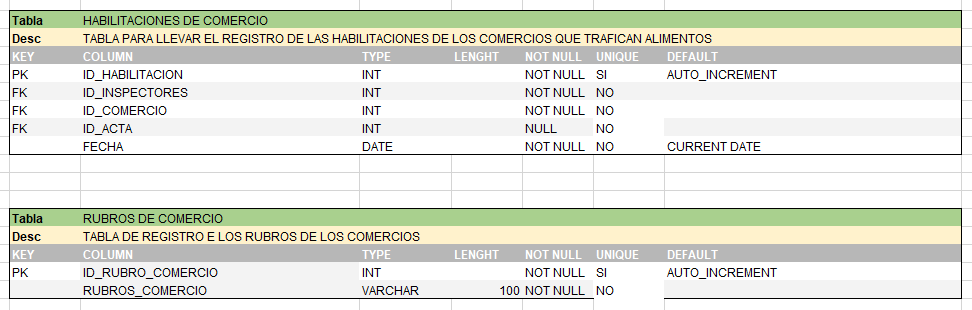
Visualización de tablas

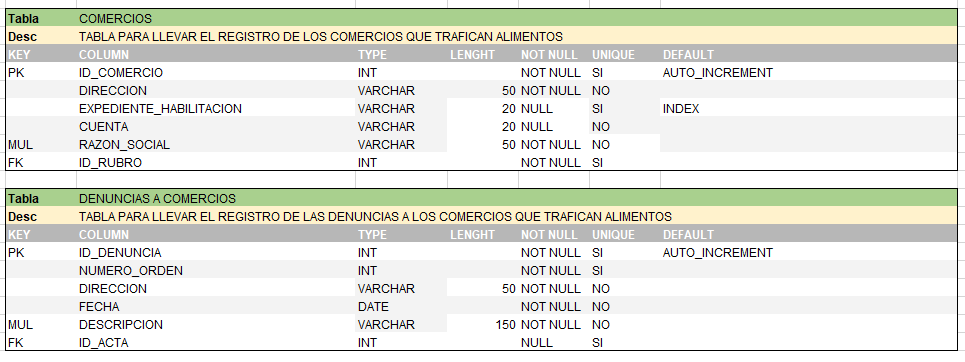


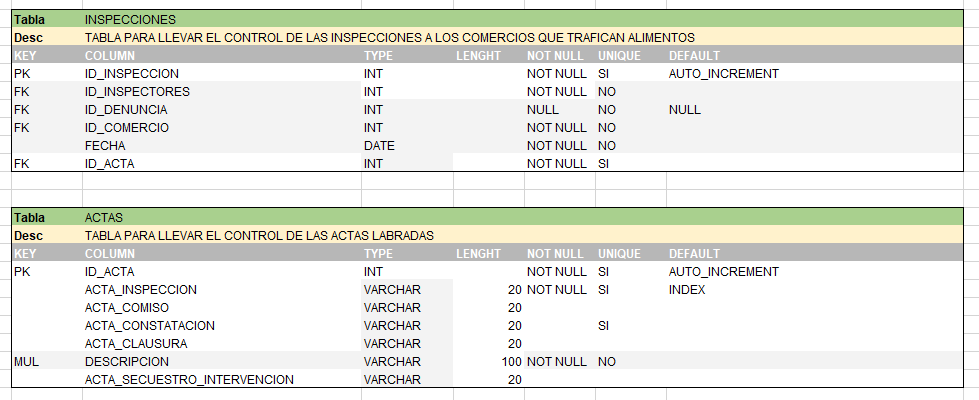


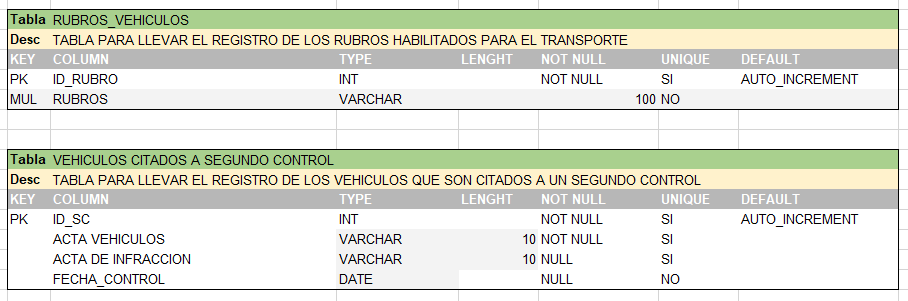
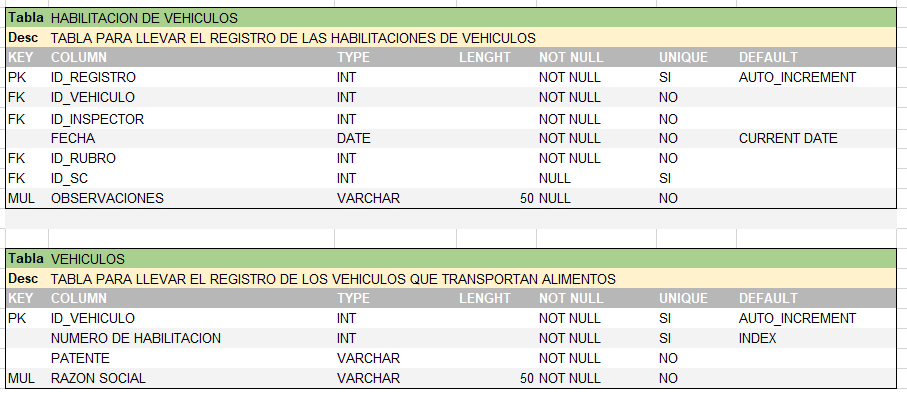


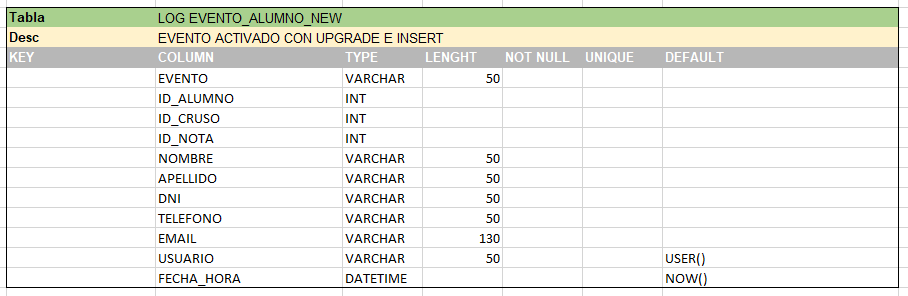


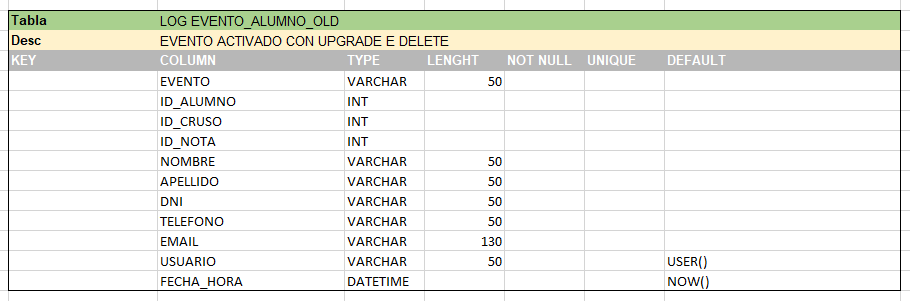


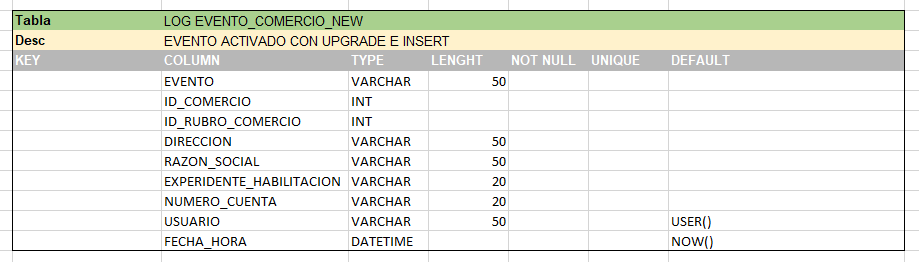


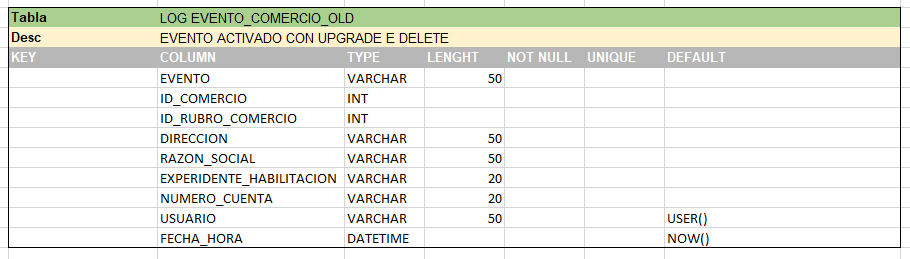


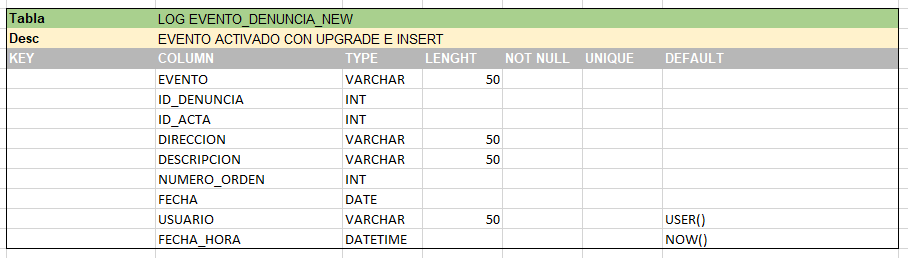


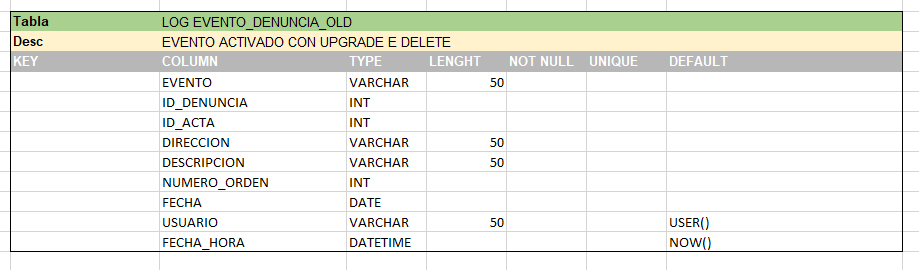


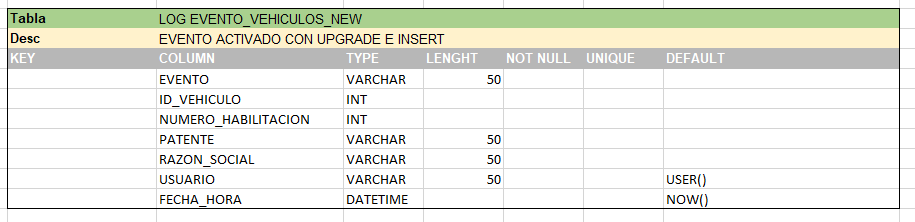


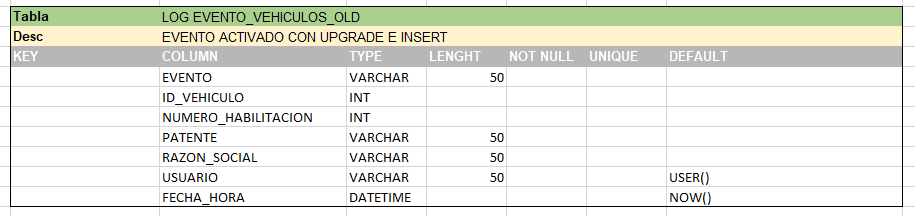












Funcionalidad

A continuación, se describen los elementos creados a partir de la estructura de la base de datos para asistir en el trabajo con la misma:

1. Vistas:

Se crearon las siguientes vistas:

* View\_Habilitacion\_Vehiculos: VIEW para visualizar las habilitaciones de vehículos, se crea para acceder rápidamente a toda la información de las habilitaciones de vehículos.
* View\_Habilitacion\_comercios: VIEW visualización habilitación de comercios, se crea para acceder rápidamente a toda la información de las habilitaciones de comercio.
* View\_Comercio: VIEW Inspecciones, se crea para acceder rápidamente a toda la información de las inspecciones.
* View\_Curso\_manipulacion: VIEW Cursos, se crea para acceder rápidamente a toda la información de los cursos de manipulación de alimentos.
* View\_grupo\_inspectores: VIEW Grupo-inspectores, se crea para acceder rápidamente a toda la información de los inspectores.

1. Funciones:

Se crearon las siguientes funciones:

* contar\_habilitacion\_vehiculos(Fecha DATE): Función para hacer un conteo rápido de todas las habilitaciones de vehículos hechas en un periodo de tiempo.
* contar\_habilitacion\_comercios(Fecha DATE): Función para hacer un conteo rápido de todas las habilitaciones de comercios hechas en un periodo de tiempo.
* cursos\_profesor(profesor INT): Función para saber la cantidad de cursos dictados por cada profesor.
* denuncias\_pendientes(Fecha DATE): Función creada para saber la cantidad de denuncias sin realizar.

1. Stored procedure:

* sp\_vw\_comercio\_orden: Ordena la habilitación de comercios de forma ascendente o descendente.
* sp\_alumnos\_nota: Actualiza las tablas de curso de manipulación de alimentos y de alumnos cada vez q se cargue una nota.

1. Triggers:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Trigger | Tabla monitoreada | Acciones que monitorea | Tabla donde registra | Descripción |
| tr\_vehiculos\_update\_OLD | vehículos | Before | log\_vehiculos\_evento\_OLD | Crea un backup de los datos modificados |
| tr\_vehiculos\_delete\_OLD | vehículos | Before | log\_vehiculos\_evento\_OLD | Crea un backup de los datos eliminados. |
| tr\_denuncias\_update\_OLD | denuncias | Before | log\_denuncias\_evento\_OLD | Crea un backup de los datos modificados |
| tr\_denuncias\_delete\_OLD | denuncias | Before | log\_denuncias\_evento\_OLD | Crea un backup de los datos eliminados. |
| tr\_alumno\_update\_OLD | alumnos | Before | log\_alumnos\_evento\_OLD | Crea un backup de los datos modificados |
| tr\_alumno\_delete\_OLD | alumnos | Before | log\_alumnos\_evento\_OLD | Crea un backup de los datos eliminados. |
| tr\_comercio\_update\_OLD | comercios | Before | log\_comercio\_evento\_OLD | Crea un backup de los datos modificados |
| tr\_comercio\_delete\_OLD | comercios | Before | log\_comercio\_evento\_OLD | Crea un backup de los datos eliminados. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Trigger | Tabla monitoreada | Acciones que monitorea | Tabla donde registra | Descripción |
| tr\_vehiculos\_update\_NEW | vehículos | After | log\_vehiculos\_evento\_NEW | Registra cada registro que se modifica. |
| tr\_vehiculos\_insert\_NEW | vehículos | After | log\_vehiculos\_evento\_NEW | Registra cada registro que se inserta. |
| tr\_denuncias\_update\_NEW | denuncias | After | log\_denuncias\_evento\_NEW | Registra cada registro que se modifica. |
| tr\_denuncias\_ insert \_NEW | denuncias | After | log\_denuncias\_evento\_NEW | Registra cada registro que se inserta. |
| tr\_alumno\_update\_NEW | alumnos | After | log\_alumnos\_evento\_NEW | Registra cada registro que se modifica. |
| tr\_alumno\_ insert \_NEW | alumnos | After | log\_alumnos\_evento\_NEW | Registra cada registro que se inserta. |
| tr\_comercio\_update\_NEW | comercios | After | log\_comercio\_evento\_NEW | Registra cada registro que se modifica. |
| tr\_comercio\_ insert \_NEW | comercios | After | log\_comercio\_evento\_NEW | Registra cada registro que se inserta. |

Creación de la base de datos y sus tablas

Repositorio de GitHub:

* ScriptTablas+Rave.sql (creación de tablas).
* ScriptIns+Rave.sql (inserción de datos).
* ScriptView+Rave.sql (creación de vistas).
* ScriptFunciones+Rave.sql (creación de funciones).
* ScriptProcedure+Rave.sql (creación de stored procedures).
* ScriptTrigger+Rave.sql (creación de triggers, tablas logs e inserciones y updates como pruebas).

<https://github.com/TomyReiv/CoderHouse_SQL.git>