KIOSKSDB

Curso: SQL Comisión: 53175

Nombre del alumno: Vazquez Velasquez, Henry José

Temática de la Base de Datos

La temática es para un Sistema de Base de Datos que usarán varios locales de mercadería (kioscos) donde se pueden ingresar los productos que se adquieren para ponerlos a la venta, crear las facturas automáticamente desde el programa y generar estadísticas que sirvan para mejorar el rendimiento de los locales.

Objetivo de la Base de Datos

El objetivo es almacenar los datos más importantes del negocio para gestionarlo y generar procedimientos automáticos que permitan agilizar la realización de compras y ventas de los locales de manera que sea escalable y eficiente.

Modelo de Negocio

La manera básica de comerciar para los kioscos es que se compren diversos productos a un precio más barato del que se venden, además de recopilar información del local, del cliente, del producto, entre otros; para crear una factura con toda la información relacionada a la venta de un producto.

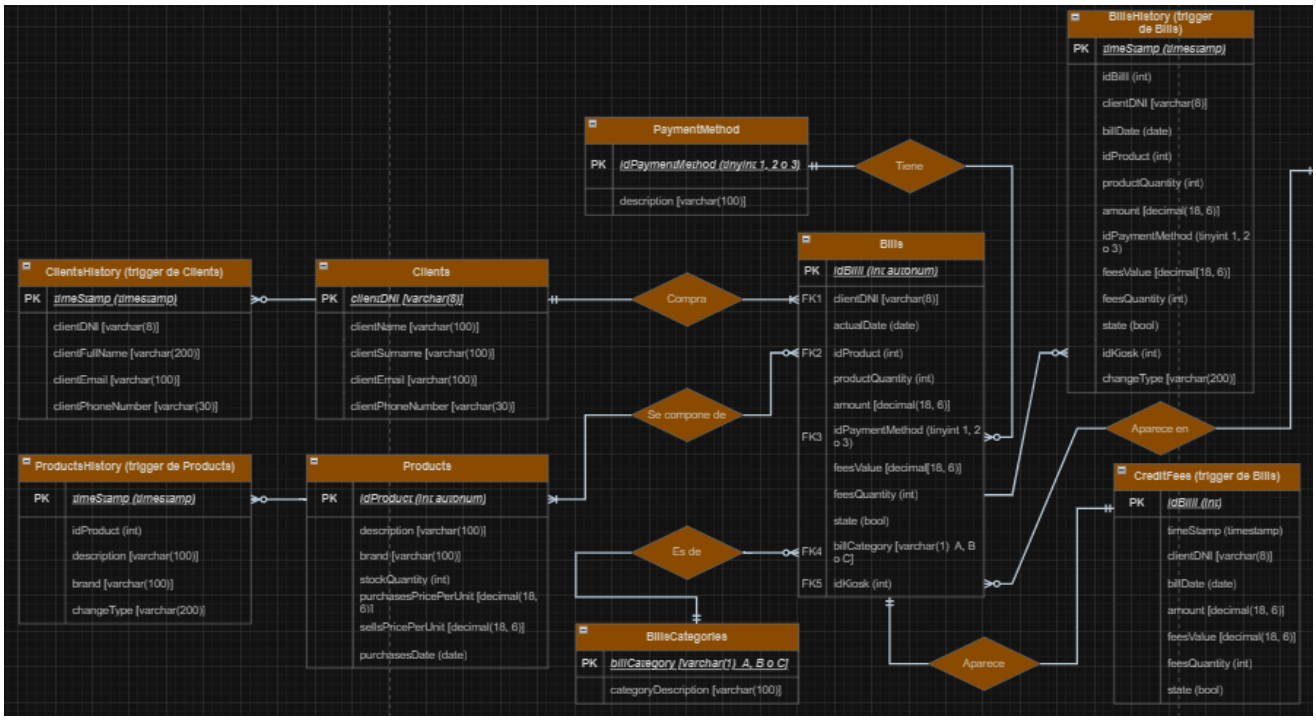
Índice de Página

(Presione ***ctrl* + *click izquierdo*** en uno de los nombres de las tablas o en la opción del DER para ir directamente a esa sección de la página)

Volver al Inicio

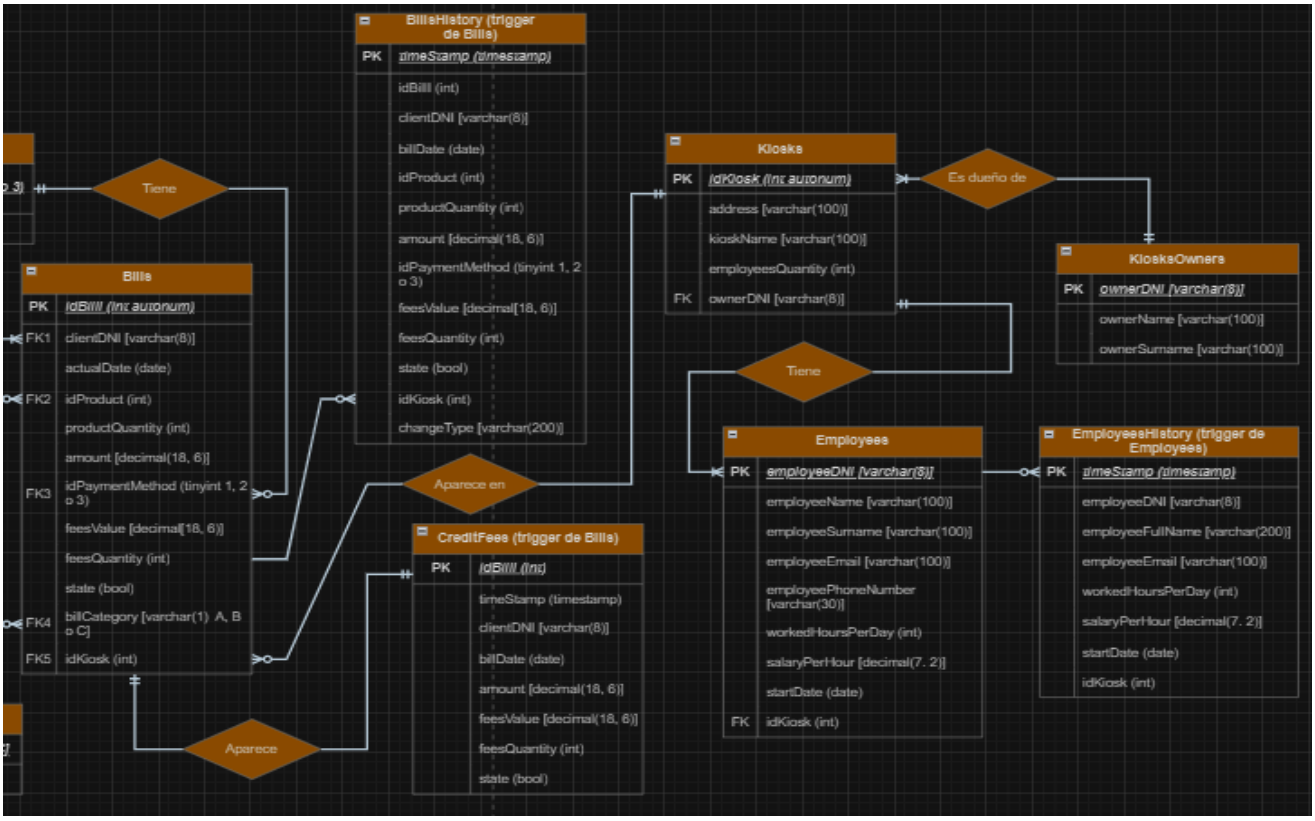
Diagrama Entidad – Relación (DER)

Parte izquierda



Volver al Inicio

Parte derecha



Nombre del archivo para editar el DER: ***DER Proyecto Final CoderHouse.drawio***

Descripción de las Tablas

Tablas Dimensionales

1. **Categorías de Factura**: Tabla que tendrá la información de los tipos de factura en que se puede realizar la compra según la clase de contribuyente que sea el cliente.
   A = Responsable inscripto a Responsable inscripto/Monotributista.
   B = Responsable inscripto a Consumidor final/exento de IVA.
   C = Monotributista/Exento en IVA a cualquier destinatario.
   Atributos:
   - Categoría de la factura [varchar(1) A, B, C] [**Primary Key**]
   - Descripción de la categoría [varchar(100)]

2. **Clientes**: Almacenará la información de los clientes que le compren un producto a los kioscos, principalmente tendrán la función de recuperar los datos ingresados en esta tabla para generar la factura con sus datos relevantes.
   Atributos:
   - DNI del cliente [varchar(8)] [**Primary Key**]
   - Nombre del cliente [varchar(100)]
   - Apellido del cliente [varchar(100)]
   - El email del cliente [varchar(100)]
   - El teléfono del cliente [varchar(30)]

Volver al Inicio

3. **Dueños de los Kioscos**: Entidad que mantendrá la información más relevante de los dueños de cada kiosco.
   Atributos:
   - DNI del dueño [varchar(8)] [**Primary Key**]
   - Nombre del dueño [varchar(100)]
   - Apellido del dueño [varchar(100)]

4. **Kioscos**: Contendrá la información relevante sobre los locales que llevan a cabo el modelo de negocio.
   Atributos:
   - ID del kiosco [int autonum] [**Primary Key**]
   - Dirección del local [varchar(100)]
   - Nombre del kiosco [varchar(100)]
   - Cantidad de empleados [int]
   - DNI del dueño [varchar(8)] [**Foreign Key**]

5. **Empleados**: Tabla dimensional que guardará los datos relevantes de los trabajadores que tenga cada local.
   Atributos:
   - DNI del empleado [varchar(8)] [**Primary Key**]
   - Nombre del empleado [varchar(100)]
   - Apellido del empleado [varchar(100)]
   - Email del empleado [varchar(100)]
   - Teléfono del empleado [varchar(30)]
   - Aproximado de horas trabajadas en el día [int]
   - Salario por hora trabajada del empleado [decimal(7, 2)]
   - Fecha en que empezó a trabajar en el kiosco [date]
   - ID del kiosco donde trabaja [int] [**Foreign Key**]

6. **Métodos de Pago**: Entidad que se especializará en guardar la información de las modalidades de pago (por ahora son 3: efectivo, débito y crédito).
   Atributos:
   - ID del método de pago [tinyint 1 a 3] [**Primary Key**]
   - Descripción del método de pago [varchar(100)]

7. **Productos**: Tendrá la información básica de los productos que se compran para el local y que pondrán venderse a los clientes posteriormente.
   Atributos:
   - ID del producto [int autonum] [**Primary Key**]
   - Descripción del producto [varchar(100)]
   - Marca del producto [varchar(100)]

Volver al Inicio

- Cantidad disponible en stock del producto [int]
- Precio al que se compró el producto [decimal(15, 6)]
- Precio en el que se venderá el producto [decimal(18, 6)]*
- La fecha en que se compró el producto [date]

*__Aclaración__: Se recomienda que el precio de venta sea siempre mayor al precio de compra en un 35% al 70% para que sea rentable la compra del producto.

El porcentaje de ganancia del producto, con respecto al precio de compra, lo podrá elegir el administrador de la Base de Datos utilizando la función *`fn_calculate_sellsPricePerUnit`* introduciendo como parámetros el precio de compra y el porcentaje de ganancia.

Tablas de Hechos

8. **Facturas**: Tabla de hechos que reunirá los datos más importantes de las 7 tablas anteriores con el objetivo de que sirva como factura válida de las ventas del local.
   Atributos:
   - ID de la factura [int autonum] [**Primary Key**]
   - DNI del cliente [varchar(8)] [**Foreign Key**]
   - Fecha en la que se realizó la factura [date]
   - ID del producto [int] [**Foreign Key**]
   - Cantidad comprada del producto [int]
   - Total de la factura [decimal(18, 6)]
   - ID del método de pago [tinyint 1, 2 o 3] [**Foreign Key**]
   - Valor de las cuotas con ese método de pago [decimal(18, 6)]
   - Cantidad de cuotas con ese método de pago [int]
   - Estado de la factura [bool]*
   - Categoría de la factura [varchar(1) A, B, C] [**Foriga Key**]**
   - ID del kiosco donde se realizó la compra [int] [**Foreign Key**]

   * **Aclaración 1 (Estado de la factura)**: False o “0” equivaldría *Cancelada* y True o “1” a *Válida*.

   ** **Aclaración 2 (Categoría de la factura)**: Responsable inscripto, Monotributo, etc.

Tablas creadas por Triggers

9. **Cuotas con Crédito**: Tabla especializada en guardar los datos de las facturas que faltan por pagar o ya fueron totalmente pagadas con el método de tarjeta de crédito. Acá pueden estar facturas con ninguna o con varias cuotas.
   Atributos:
   - Fecha y hora exacta en la que se cargó la factura [timestamp]
   - ID de la factura [int] [**Primary Key]**

Volver al Inicio

- DNI del cliente [varchar(8)]
- Fecha de la factura [date]
- Total de la factura [decimal(18, 6)]
- Valor de las cuotas de la factura [decimal(18, 6)]
- Cantidad de cuotas de la factura [int]
- Estado de la factura [bool]

10. **Historial de Clientes**: Entidad que guardará los datos anteriores de los clientes cuando sus registros originales sean actualizados en la tabla Clientes (tabla principal) a modo de auditoría o por si se cometió un error con la actualización de los datos.
    Atributos:
    - Fecha y hora exacta en la que se cargó la factura [timestamp] [**Primary Key]**
    - DNI del cliente [varchar(8)]
    - Nombre completo del cliente [varchar(200)]
    - El email del cliente [varchar(100)]
    - El teléfono del cliente [varchar(30)]

11. **Historial de Empleados**: Entidad que guardará los datos anteriores de los empleados cuando sus registros originales sean actualizados en la tabla Empleados (tabla principal) a modo de auditoría o por si se cometió un error con la actualización de los datos.
    Atributos:
    - Fecha y hora exacta en la que se cargó la factura [timestamp] [**Primary Key]**
    - DNI del empleado [varchar(8)]
    - Nombre completo del empleado [varchar(200)]
    - El email del empleado [varchar(100)]
    - Aproximado de horas trabajadas en el día [int]
    - Salario por hora trabajada del empleado [decimal(7, 2)]
    - Fecha en que empezó a trabajar en el kiosco [date]
    - ID del kiosco donde trabaja [int]

12. **Historial de Productos**: Entidad que guardará los datos anteriores de los empleados cuando sus registros originales sean actualizados en la tabla Productos (tabla principal) a modo de auditoría o por si se cometió un error con la actualización de los datos.
    Atributos:
    - Fecha y hora exacta en la que se cargó la factura [timestamp] [**Primary Key]**
    - ID del producto [int autonum] [**Primary Key**]
    - Descripción del producto [varchar(100)]
    - Marca del producto [varchar(100)]

Volver al Inicio

- Descripción del tipo de modificación realizada al registro original [varchar(200)]

13. **Historial de Facturas**: Entidad que guardará los datos anteriores de las facturas cuando sus registros originales sean actualizados en la tabla Facturas (tabla principal) a modo de auditoría o por si se cometió un error con la actualización de los datos.
    Atributos:
    - Fecha y hora exacta en la que se cargó la factura [timestamp] [**Primary Key]**
    - ID de la factura [int]
    - DNI del cliente [varchar(8)]
    - Fecha en la que se realizó la factura [date]
    - ID del producto [int]
    - Cantidad comprada del producto [int]
    - Total de la factura [decimal(18, 6)]
    - ID del método de pago [tinyint 1, 2 o 3]
    - Valor de las cuotas con ese método de pago [decimal(18, 6)]
    - Cantidad de cuotas con ese método de pago [int]
    - Estado de la factura [bool]
    - ID del kiosco donde se realizó la compra [int]
    - Descripción del tipo de modificación realizada al registro original [varchar(200)]

Objetos de la Base de Datos

Vistas (Views)

1. **vw_full_information_bill**: Trae información de casi todas las tablas dimensionales (menos la de *Empleados* y la de *Dueños de los Kioscos*) para ver el detalle completo de la factura incluso con los datos que no se almacenan directamente en la tabla *Facturas*.

2. **vw_full_information_kiosks**: Reúne toda la información de la tabla *Kioscos* y le agrega la información de la tabla *Dueños de los Kioscos* según corresponda su DNI para tener un poco más de contexto de quien maneja ese local.

3. **vw_employees_under_average_salary**: Permite visualizar los datos más relevantes de los empleados que tienen un salario por hora menor que el promedio, esto lo hace solo utilizando los datos de la tabla *Empleados*.

4. **vw_products_upon_average_purchases_price**: Se crea la vista para observar la información complete de los productos cuyo precio de compra sea mayor al promedio para revisar su rendimiento constantemente.

Funciones (Functions)

Volver al Inicio

1. **fn_calculate_sellsPricePerUnit**: Función que permite ingresar por parámetro el precio al que se compró un producto y su porcentaje de ganancia para generar automáticamente su precio de venta.

2. **fn_calculate_salaryPerDay**: Permite visualizar el salario que le corresponde a un empleado en una x cantidad de días según el salario que cobra por hora, las horas trabajadas por día y la cantidad de días que se quiere saber.

3. **fn_calculate_billAmount**: Calcula el total de una compra según el ID del producto y la cantidad de unidades compradas que son ingresadas como parámetros.

Procedimientos Almacenados (Stored Procedures)

1. **sp_calculate_most_selled_product**: Calcula cual fue el producto más comprado dentro de la tabla *Facturas* y con ese dato busca su información más importante que está guardada en la tabla *Productos* para tener una noción de lo que más se compra y la cantidad de veces que se compró el producto.

2. **sp_calculate_employee_seniority_in_years**: Calcula cuantos años tiene un empleado trabajando en un kiosco solo ingresando su DNI, esto sirve para tener una idea del tiempo que lleva el empleado en el local y si correspondería aumentarle el salario según las horas trabajadas.

Disparadores (Triggers)

Trigger	Tabla a Revisar	Acción que revisa	Tabla de Destino	Descripción
tgr_after_insert_credit_payments_bills	bills	After Insert	CreditFees	Registra la información de las facturas que fueron pagadas con crédito para tener un control de las cuotas que existen con este método de pago.
tgr_after_update_Clients	clients	After Update	ClientsHistory	Registra la información antigua de los clientes por si se actualiza con error o por si se quiere comparar los datos anteriores con los nuevos para auditar y crear un recorrido
tgr_after_update_Employees	employees	After Update	EmployeesHistory	Registra la información antigua de los empleados por si se actualiza con error o por si se quiere comparar los datos anteriores con los nuevos para auditar y crear un recorrido

Volver al Inicio

tgr_after_update_Products	products	After Update	ProductsHistory	Registra la información antigua de los productos por si se actualiza con error o por si se quiere comparar los datos anteriores con los nuevos para auditar y crear un recorrido. Además, se incorpora una columna donde resume el tipo de cambio que se le realizó al registro original.
tgr_after_update_Bills	bills	After Update	BillsHistory	Registra la información antigua de las facturas por si se actualiza con error o por si se quiere comparar los datos anteriores con los nuevos para auditar y crear un recorrido. Además, se incorpora una columna donde resume el tipo de cambio que se le realizó al registro original.

Paso a Paso para Insertar los Datos en las Tablas

Por Interfaz Gráfica

1. Con el **MySQL Workbench** abierto, ver en el panel izquierdo la tabla donde se desea ingresar los datos y darle click derecho.

2. Elegir la opción de "**Table Data Import Wizard**".

3. Se abrirá una ventana donde te pedirá ingresar la ruta del archivo que contiene los datos, presiona a la derecha donde dice "**Browse**" o "**Examinar**".

4. Elige el archivo que corresponda con la tabla (aparece con el mismo nombre de la tabla agregando la palabra "**_data.csv**"). Una vez seleccionado el archivo correctamente, presionar en "**Next>**".

5. Aparecerá otra ventana indicando si se quiere ingresar los datos en una tabla ya existente o si se quiere crear una nueva tabla para esta importación, nosotros seleccionaremos "**Usar una tabla existente**" o "**Use existing table**".

A considerar: Si es la primera vez que se insertan los datos sobre la tabla, el cuadro de "**Truncar datos antes de importar**" o "**Truncate data before import**" debe estar desmarcado; por el contrario, si algunos datos aún siguen en la tabla y quiere ingresar todos de nuevo, tendrá que marcar esa opción.

6. Seleccionar, si se requiere, las columnas de la tabla destino para que coincida con las de la tabla fuente (ambas columnas tienen los mismos nombres), presionar en "**Next>**".

7. Aparecerá otra ventana justo antes de ingresar los datos, presionar a la izquierda

Volver al Inicio

en el botón de "**Show Logs**" y presionar en "**Next>**" cuando esté listo.

8. Se abrirá la última ventana que indicará cuántos registros fueron ingresados correctamente en la tabla.

A Través del Archivo "Population.sql"

1. Se abren los archivos de la siguiente manera para crear la estructura de la Base de Datos:
    - KiosksDB.sql
    - 1. Views.sql (se encuentra en la carpeta DB_objects)
    - 2. Functions.sql (se encuentra en la carpeta DB_objects)
    - 3. Stored Procedures.sql (se encuentra en la carpeta DB_objects)
    - 4. Triggers.sql (se encuentra en la carpeta DB_objects)
    - Population.sql

Se recomienda este orden en específico ya que así se crean los triggers que permitirán hacer transacciones/traspasos automáticos de una tabla a otra según corresponda.

Herramientas Utilizadas

- **ChatGPT** (consultas sobre errores en código).
- **Documentación de MySQL** (revisión de sintaxis y de maneras diferentes para realizar determinadas sentencias).
- **Git** (sistema de control de versiones para crear un repositorio local y subir los archivos a un repositorio remoto)
- **GitHub** (creación de repositorio remoto para almacenar los archivos de la Base de Datos).
- **Mockaroo** (generación de datos aleatorios de manera masiva).

Volver al Inicio