UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

División de Ingeniería Eléctrica
Departamento de Ingeniería en Computación

Bases de Datos

Proyecto final de la asignatura de bases de datos

Ing. Fernando Arreola

Noviembre 2024

Equipo DataBenders:

Integrantes:

Luis Fernando Franco Arellano
Gustavo Isaac Soto Huerta
Eduardo Zavala Sánchez
José Eduardo Villeda Tlecuitl

Semestre 2025 - 1 **Fecha de realización: 23 de Agosto de 2024**

Capítulo 1

Introducción

En el ámbito de la gestión empresarial, el manejo adecuado y eficiente de la información es fundamental para optimizar procesos y mejorar la toma de decisiones. Este proyecto tiene como propósito desarrollar una solución de base de datos que permita a una cadena de papelerías almacenar y organizar su información de forma estructurada y accesible, contribuyendo así a su modernización operativa. La implementación de esta base de datos busca satisfacer una serie de requerimientos específicos, tales como el almacenamiento de datos detallados de proveedores, clientes, inventarios y transacciones de ventas, asegurando a su vez la integridad de la información y la eficiencia en su manejo.

Este sistema no sólo se limita a almacenar información; también está diseñado para automatizar ciertos procesos críticos, como la generación de facturas, la actualización del inventario, y el cálculo de utilidades, entre otros. A lo largo del desarrollo, se deberán aplicar conceptos de diseño de bases de datos y lenguajes de consulta, tomando en cuenta las particularidades del negocio y las reglas de integridad que garanticen la consistencia de los datos. Además, se realizarán configuraciones específicas, tales como la creación de índices y estructuras adicionales para optimizar el acceso a la información y la generación de reportes.

De este modo, el proyecto permite a los estudiantes aplicar conocimientos de diseño y administración de bases de datos en un contexto práctico, explorando cómo una infraestructura de datos bien construida puede ofrecer un soporte fundamental a las operaciones y la estrategia de una empresa.

1.1. Objetivo

El alumno analizará una serie de requerimientos y propondrá una solución que atienda a los mismos, aplicando los conceptos vistos en el curso.

1.2. Descripción del Problema

El problema se divide en dos partes:

1.2.1. Parte Uno

Consiste en el diseño de una base de datos. Una cadena de papelerías busca innovar la manera en que almacena su información, y los contratan para que desarrollen los sistemas informáticos para satisfacer los siguientes requerimientos:

- Se desea tener almacenados datos como la razón social, domicilio, nombre y teléfonos de los proveedores; RFC, nombre, domicilio y al menos un email de los clientes.
- Es necesario tener un inventario de los productos que se venden, en el que debe guardarse el código de barras, precio al que fue comprado el producto, foto, fecha de compra y cantidad de ejemplares en la bodega (stock).
- Se desea guardar la marca, descripción y precio de los regalos, artículos de papelería, impresiones y recargas, siempre y cuando se tenga su correspondiente registro en el inventario.
- Debe también guardarse el número de venta, fecha de venta, la cantidad total a pagar por la venta, empleado que concreta la venta, así como la cantidad de cada artículo y precio total a pagar por artículo.
- De los empleados interesa conocer su clave de empleado, nombre, fecha de nacimiento y fecha de ingreso.

Adicional al almacenamiento de información, se requiere que el sistema resuelva lo siguiente:

- Cada que se solicite, se genere una muestra de información que contenga la información necesaria para asemejarse a una factura de una compra.
- Dada una fecha, o una fecha de inicio y fecha de fin, regresar la cantidad total que se vendió y la ganancia correspondiente en esa fecha o periodo.
- Cada que haya la venta de un artículo, deberá decrementarse el stock por la cantidad vendida de ese artículo. Si el valor llega a cero, abortar la transacción. Si el pedido se completa pero quedan menos de 3 en stock, se deberá emitir una alerta. Debe actualizarse el total a pagar por artículo y el total a pagar por la venta.
- Permitir obtener el nombre de aquellos productos de los cuales hay menos de 3 en stock. El resultado debe almacenarse en una tabla independiente que sólo debe almacenar la información actualizada.
- Al recibir el código de barras de un producto, regrese la utilidad.
- Crear al menos un índice, del tipo que se prefiera y donde se prefiera. Justificar el porqué de la elección en ambos aspectos.

1.2.2. Consideraciones

- Puede haber distintas soluciones al problema.
- Los requerimientos enlistados anteriormente deberán ser realizados por medio de PgSQL, con los elementos que se consideren adecuados para resolverlos.
- El folio de la venta debe tener un formato similar a VENT-001, prefijo `VENT`, seguido de un guion y un número secuencial.
- Donde esté presente el atributo `domicilio`, está compuesto por estado, código postal, colonia, calle y número.
- Donde esté presente el atributo `nombre`, está compuesto por nombre, apellido paterno y materno.
- La propuesta debe contemplar todas las etapas de diseño, las consideraciones anteriores y un buen manejo de información.

1.2.3. Parte dos

Una vez diseñada e implementada la base de datos, deberá elegirse la integración de alguno de los siguientes puntos:

1. Generar un dashboard que permita visualizar, al menos:
   - Ingresos del mes (Invertido, Ingresos y Ganancias)
   - Top 3 de artículos más vendidos
   - Empleados que registran más órdenes
2. Programar en C o en Java, un objeto que simule ser una tabla de una base de datos, lo que implica que permita ingresar, borrar, actualizar y consultar información.
3. Ingresar, a partir de archivos de texto, la información registrada durante el día a otra base de datos. Se debe orquestar la inserción de cada archivo y tener validaciones del flujo en caso de errores.

1.2.4. Entregables

Es necesario entregar lo que se solicita a continuación:

1.2.5. Documento

Se deberá entregar un documento **FORMAL** elaborado en LaTeX, que contenga las siguientes secciones:

- **Introducción:** Breve descripción del análisis del problema, sus objetivos, su(s) propuesta(s) de solución. Elaborada por ustedes mismos.
- **Plan de trabajo:** Queda a criterio de cada equipo el grado de detalle que se le dará a esta sección, pero mínimo debe contener descripción general de las actividades a realizar, su correspondiente plan de actividades y un breve detalle de qué hizo cada miembro del equipo.

- **Diseño:** Descripción de lo realizado en las correspondientes fases de diseño de las bases de datos, agregando los resultados de cada una de ellas.
- **Implementación:** Descripción de funcionamiento y código de los *stored procedures*, *triggers*, funciones, etc. empleados para cumplir con los requerimientos del problema. También debe incluirse el DDL de la base de datos.
- **Presentación:** Descripción de lo que hace la modalidad seleccionada como forma de conexión hacia la base de datos.
- **Conclusiones:** Personales, detallando las dificultades, retos, aciertos, etc. que se presentaron en el proyecto.

1.2.6. Código

A través de GitHub, se deberá crear una carpeta por equipo donde se anexe lo siguiente, asignando nombres **ADECUADOS** para cada documento/archivo:

- El documento del punto anterior
- Códigos fuente de MER, MR
- Script de creación de la base de datos y tablas
- Script para el agregado de información
- Script de toda la programación a nivel BD
- Códigos de lo implementado como parte de la etapa de presentación
- La presentación a emplear a la hora de exponer

1.3. Exposición

Como parte de la evaluación del curso, se deberá realizar una presentación del proyecto. La idea es simular la presentación de una solución de software a un posible cliente, lo que implica descripción del software, muestras de funcionalidad, solución de dudas, etc. Cada equipo es libre de decidir cómo hacer su presentación, el único detalle es que debe ser **TOTALMENTE formal**.

1.4. Puntos a considerar

Se tomará en cuenta lo siguiente, partiendo del porcentaje de calificación estipulado al inicio del semestre:

- Equipos que se copien en alguna parte del proyecto, su calificación del curso será 5.
- Equipo(s) que reprueben el proyecto o la parte de exposición, deberán presentar examen final sin importar lo obtenido en los otros rubros del curso.
- La calificación del primer examen final será 30 % lo obtenido en el proyecto, 70 % lo obtenido en el examen.
- La seriedad, formalidad, presentación y calidad del proyecto, si bien no tendrán un valor numérico específico, pueden jugar a favor o en contra de la calificación de los rubros correspondientes.
- Cualquier punto no contemplado en el presente documento será determinado por el profesor.

1.5. Recomendaciones

No es obligatorio, pero se sugieren los siguientes puntos:

- Delegar correctamente responsabilidades
- No demorar en formar sus equipos y agregar su carpeta en GitHub
- Empleo de herramientas colaborativas
- Empleo de arquitecturas cliente-servidor remotas o servicios en la nube
- Redacción y presentación en inglés

Capítulo 2

Desarrollo de la Parte Uno

2.1. Entidades, Atributos y Relaciones Encontradas

Esta sección presenta el Modelo Entidad-Relación (MER) correspondiente al sistema de gestión de la cadena de tiendas de papelería. A continuación se describen las entidades, sus atributos y las relaciones que se han identificado para cumplir con los requerimientos del sistema.

2.1.1. Entidades y Atributos

Para desarrollar este modelo, identificamos 10 entidades clave. Decidimos incorporar catálogos individuales para estados y países, lo cual facilita tanto la búsqueda como la inserción de datos en el sistema. Esto permite mantener la consistencia de datos al gestionar ubicaciones.

En cuanto a las relaciones, solo se identificó una entidad débil: **StockBajo**. Esta entidad es débil porque depende exclusivamente de la existencia de un producto.

A continuación, se detallan las entidades principales junto con sus atributos:

- **Proveedores**
  - **ProveedorID** (Clave Primaria)
  - NombreComercial
  - Domicilio
    - Domicilio_Calle
    - Domicilio_Número
    - Domicilio_Colonia
    - Domicilio_CódigoPostal
    - Domicilio_Estado
  - Nombre
    - Nombre
    - ApellidoPaterno
    - ApellidoMaterno (Opcional)
  - Teléfonos (Multivalorado)
  - RFC (Clave Candidata)
- **Clientes**
  - **ClienteID** (Clave Primaria)
  - Nombre
    - Nombre
    - ApellidoPaterno
    - ApellidoMaterno (Opcional)
  - Domicilio
    - Domicilio_Calle

    - Domicilio_Número
    - Domicilio_Colonia
    - Domicilio_CódigoPostal
    - Domicilio_Estado
  - CorreosElectrónicos (Multivalorado)
  - RFC (Clave Candidata) (Opcional)

- **Productos**
  - **ProductoID** (Clave Primaria)
  - CódigoBarras (Clave Candidata)
  - Nombre
  - PrecioCompra
  - PrecioVenta
  - Foto
  - CantidadStock

- **Categorías**
  - **CategoríaID** (Clave Primaria)
  - Nombre
  - Descripción

- **Ventas**
  - **VentaID** (Clave Primaria)
  - Folio (Clave Candidata)
  - FechaVenta
  - MontoTotalPagar

- **Empleados**
  - **EmpleadoID** (Clave Primaria)
    - Nombre
    - ApellidoPaterno
    - ApellidoMaterno (Opcional)
  - FechaNacimiento
  - FechaIngreso

- **Producto_Venta** (Relación)
  - CantidadProducto
  - PrecioTotalProducto

- **StockBajo**
  - **ProductoID** (Clave Primaria)
  - EsStockBajo

- **Países**
  - **PaísID** (Clave Primaria)
  - Nombre

- **Estados**
  - **EstadoID** (Clave Primaria)
  - Nombre

2.1.2. Relaciones

- **Proveedor** a **Producto:** Un proveedor puede ofrecer múltiples productos (1:M).
- **Cliente** a **Venta:** Un cliente puede realizar múltiples ventas (1:M).
- **Venta** a **Producto:** Muchos productos pueden estar contenidos en muchas ventas (M:M). Esta relación puede tener atributos adicionales, como *Cantidad* y *PrecioTotal*.
- **Categoría** a **Producto:** Cada producto pertenece a una categoría (M:1).
- **Empleado** a **Venta:** Un empleado puede registrar múltiples ventas (1:M).
- **Estado** a **Cliente:** Un cliente reside en un estado (1:M).
- **Estado** a **Proveedor:** Un proveedor reside en un estado (1:M).
- **País** a **Estado:** Un estado pertenece a un país (1:M).
- **Cliente** a **CorreosElectrónicosCliente:** Un cliente puede tener múltiples direcciones de correo electrónico (1:M).

2.2. Modelo Entidad-Relación

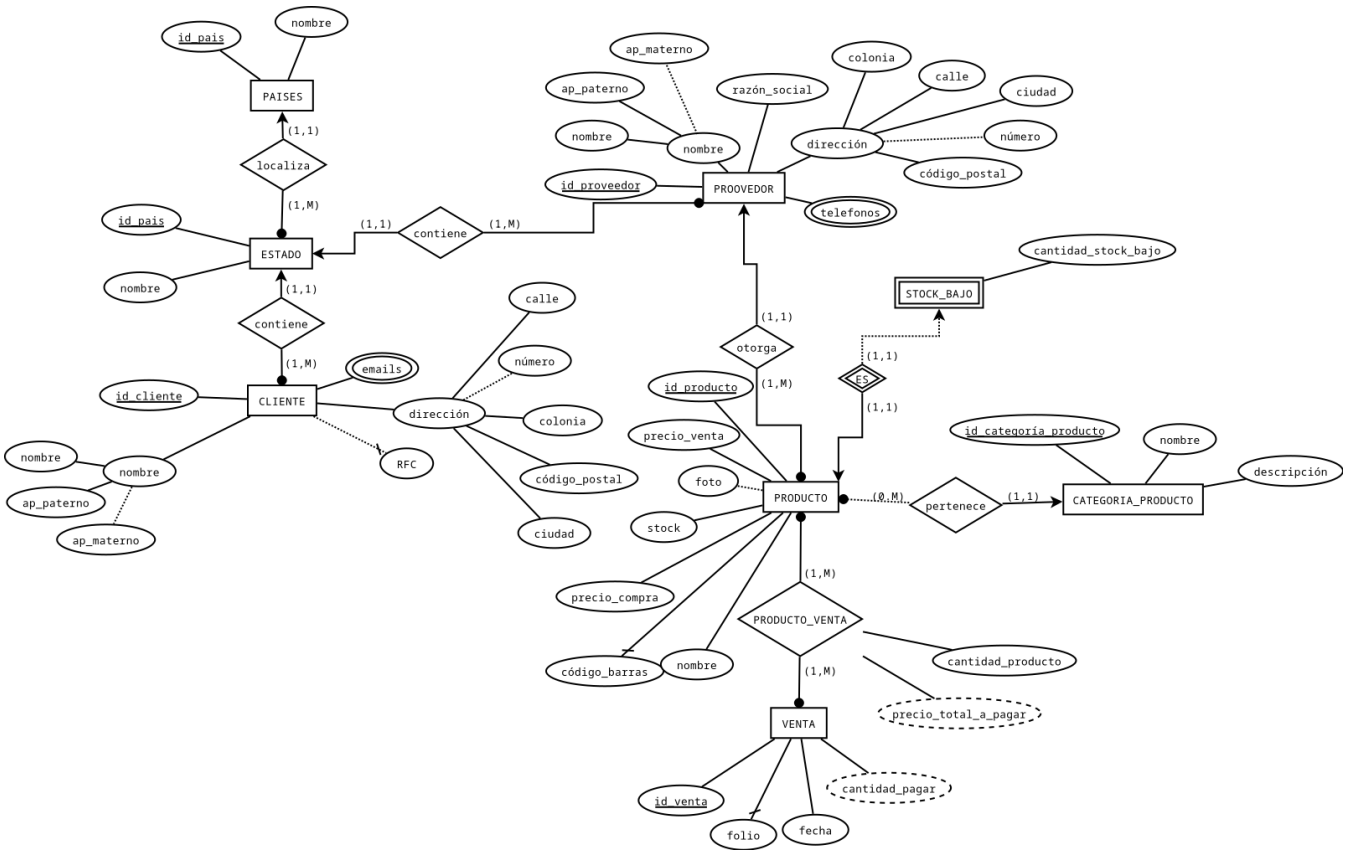

Figura 2.1: Modelo Entidad-Relación de nuestro caso de estudio

2.3. Entidades, atributos y relaciones en modelo relacional

Basamos nuestra representación en las recomendaciones de la documentación de PGModeler , donde revisamos cómo representar nuestro caso de estudio en el modelo relacional.

- **Proveedores**
  - **ProveedorID** (PK)

- RazonSocial (VARCHAR(100))
- Domicilio
  - Domicilio_Calle (VARCHAR(100))
  - Domicilio_Numero (VARCHAR(10))
  - Domicilio_Colonia (VARCHAR(100))
  - Domicilio_CodigoPostal (VARCHAR(10))
  - Domicilio_EstadoID (FK)
- Nombre (VARCHAR(100))
- RFC (VARCHAR(50), Unique)

- **ProveedorTelefonos**
  - **TelefonoID** (PK)
  - ProveedorID (FK)
  - NumeroTelefono (VARCHAR(20))
  - TipoTelefono (VARCHAR(50))
- **Clientes**
  - **ClienteID** (PK)
  - Nombre (VARCHAR(100))
  - ApellidoPaterno (VARCHAR(100))
  - ApellidoMaterno (VARCHAR(100))
  - Domicilio
    - Domicilio_Calle (VARCHAR(100))
    - Domicilio_Numero (VARCHAR(10))
    - Domicilio_Colonia (VARCHAR(100))
    - Domicilio_CodigoPostal (VARCHAR(10))
    - Domicilio_EstadoID (FK)
  - Emails (VARCHAR(100), Multivaluado)
  - RFC (VARCHAR(50), Unique) (NULL)
- **Productos**
  - **ProductoID** (PK)
  - CodigoBarras (VARCHAR(50), UNIQUE)
  - Nombre (VARCHAR(100))
  - PrecioCompra (DECIMAL(10, 2))
  - PrecioVenta (DECIMAL(10, 2))
  - Foto (BLOB)
  - CantidadStock (INT)
  - CategoriaID (FK)
  - ProveedorID (FK)
- **Categorias**
  - **CategoriaID** (PK)
  - Nombre (VARCHAR(100))
  - Descripcion (TEXT)
- **Ventas**
  - **VentaID** (PK)
  - Folio (VARCHAR(20), Unique)

- FechaVenta (DATE)
- CantidadTotalPagar (NUMERIC(10, 2))
- EmpleadoID (FK)

- **Empleados**
  - **EmpleadoID** (PK)
  - Nombre (VARCHAR(100))
  - ApellidoPaterno (VARCHAR(100))
  - ApellidoMaterno (VARCHAR(100))
  - FechaNacimiento (DATE)
  - FechaIngreso (DATE)

- **DetalleVenta**
  - **DetalleVentaID** (PK)
  - VentaID (FK)
  - ProductoID (FK)
  - CantidadProducto (INT)
  - PrecioTotalArticulo (NUMERIC(10, 2))

- **StockBajo**
  - **ProductoID** (PK, FK)
  - EnBajoStock (BOOLEAN, DEFAULT FALSE)

- **Paises**
  - **PaisID** (PK)
  - Nombre (VARCHAR(100))

- **Estados**
  - **EstadoID** (PK)
  - Nombre (VARCHAR(100))
  - PaisID (FK)

- **ClienteEmails**
  - ClienteID (FK)
  - Email (VARCHAR(100))
  - **PRIMARY KEY (ClienteID, Email)**

2.4. Modelo Relacional

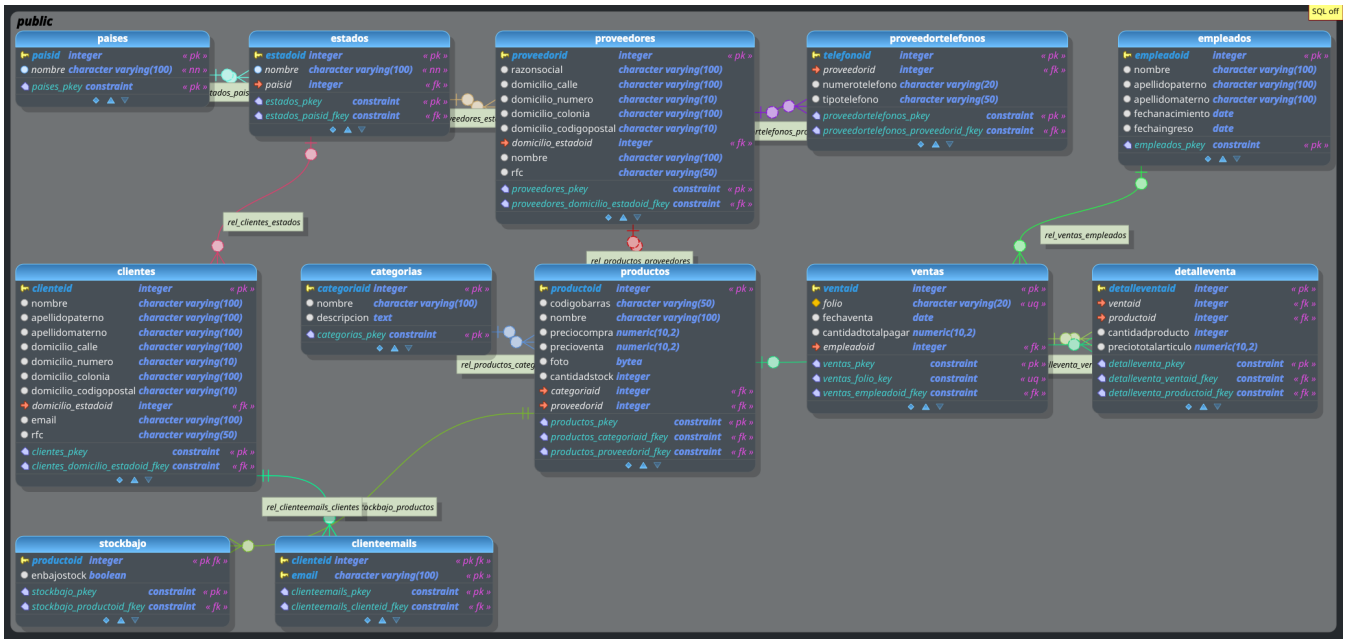

Figura 2.2: Modelo relacional de nuestro modelo

2.5. Desarrollo del modelo físico

Basandonos en nuestro modelo relacional generamos el siguiente código SQL con la herramienta PGModeler y en la documentación del desarollo de sentencias SQL para PostgreSQL obtuvimos el siguiente modelo físico.

```
CREATE TABLE Paises (
    PaisID INT PRIMARY KEY GENERATED BY DEFAULT AS IDENTITY,
    Nombre VARCHAR(100) NOT NULL
);

CREATE TABLE Estados (
    EstadoID INT PRIMARY KEY GENERATED BY DEFAULT AS IDENTITY,
    Nombre VARCHAR(100) NOT NULL,
    PaisID INT,
    FOREIGN KEY (PaisID) REFERENCES Paises(PaisID)
);

CREATE TABLE Proveedores (
    ProveedorID INT PRIMARY KEY GENERATED BY DEFAULT AS IDENTITY,
    RazonSocial VARCHAR(100),
    Domicilio_Calle VARCHAR(100),
    Domicilio_Numero VARCHAR(10),
    Domicilio_Colonia VARCHAR(100),
    Domicilio_CodigoPostal VARCHAR(10),
    Domicilio_EstadoID INT,
    Nombre VARCHAR(100),
    RFC VARCHAR(50),
    FOREIGN KEY (Domicilio_EstadoID) REFERENCES Estados(EstadoID)
);

CREATE TABLE ProveedorTelefonos (
    TelefonoID INT PRIMARY KEY GENERATED BY DEFAULT AS IDENTITY,
    ProveedorID INT,
```

```
    NumeroTelefono VARCHAR(20),
    TipoTelefono VARCHAR(50), -- Ejemplo: 'Móvil', 'Oficina', 'Fax', etc.
    FOREIGN KEY (ProveedorID) REFERENCES Proveedores(ProveedorID)
);

CREATE TABLE Empleados (
    EmpleadoID INT PRIMARY KEY GENERATED BY DEFAULT AS IDENTITY,
    Nombre VARCHAR(100),
    ApellidoPaterno VARCHAR(100),
    ApellidoMaterno VARCHAR(100),
    FechaNacimiento DATE,
    FechaIngreso DATE
);

CREATE TABLE Clientes (
    ClienteID INT PRIMARY KEY GENERATED BY DEFAULT AS IDENTITY,
    Nombre VARCHAR(100),
    ApellidoPaterno VARCHAR(100),
    ApellidoMaterno VARCHAR(100),
    Domicilio_Calle VARCHAR(100),
    Domicilio_Numero VARCHAR(10),
    Domicilio_Colonia VARCHAR(100),
    Domicilio_CodigoPostal VARCHAR(10),
    Domicilio_EstadoID INT,
    Email VARCHAR(100),
    RFC VARCHAR(50),
    FOREIGN KEY (Domicilio_EstadoID) REFERENCES Estados(EstadoID)
);

CREATE TABLE Categorias (
    CategoriaID INT PRIMARY KEY GENERATED BY DEFAULT AS IDENTITY,
    Nombre VARCHAR(100),
    Descripcion TEXT
);

CREATE TABLE Productos (
    ProductoID INT PRIMARY KEY GENERATED BY DEFAULT AS IDENTITY,
    CodigoBarras VARCHAR(50),
    Nombre VARCHAR(100),
    PrecioCompra DECIMAL(10, 2),
    PrecioVenta DECIMAL(10, 2),
    Foto BYTEA,
    CantidadStock INT,
    CategoriaID INT,
    ProveedorID INT,
    FOREIGN KEY (CategoriaID) REFERENCES Categorias(CategoriaID),
    FOREIGN KEY (ProveedorID) REFERENCES Proveedores(ProveedorID)
);

CREATE TABLE Ventas (
    VentaID INT PRIMARY KEY GENERATED BY DEFAULT AS IDENTITY,
    Folio VARCHAR(20) UNIQUE,
    FechaVenta DATE,
    CantidadTotalPagar DECIMAL(10, 2),
    EmpleadoID INT,
    FOREIGN KEY (EmpleadoID) REFERENCES Empleados(EmpleadoID)
);

CREATE TABLE DetalleVenta (
```

```
    DetalleVentaID INT PRIMARY KEY GENERATED BY DEFAULT AS IDENTITY,
    VentaID INT,
    ProductoID INT,
    CantidadProducto INT,
    PrecioTotalArticulo DECIMAL(10, 2),
    FOREIGN KEY (VentaID) REFERENCES Ventas(VentaID),
    FOREIGN KEY (ProductoID) REFERENCES Productos(ProductoID)
);

CREATE TABLE StockBajo (
    ProductoID INT PRIMARY KEY,
    EnBajoStock BOOLEAN DEFAULT FALSE,
    FOREIGN KEY (ProductoID) REFERENCES Productos(ProductoID)
);

CREATE TABLE ClienteEmails (
    ClienteID INT,
    Email VARCHAR(100),
    PRIMARY KEY (ClienteID, Email),
    FOREIGN KEY (ClienteID) REFERENCES Clientes(ClienteID)
);
```

Bibliografía

[pgModeler, 2024] pgModeler (2024). pgmodeler documentation: Relationships. Accessed: 2024-11-03.

[PostgreSQL Global Development Group, 2024] PostgreSQL Global Development Group (2024). Postgresql documentation: Table basics. Accessed: 2024-11-03.