Diseño de Base de Datos - TaskManager

Autor: Carlos Alberto Bolaños Gamarra
Rodrigo Holgado Quispe
Fabricio Huillca Perez
Raul Ppacsi Chillihuani
Versión: 1.2

1. Introducción

Este documento describe el modelo de base de datos relacional diseñado para el sistema Agencia de Viaje**s**, cuyo propósito es gestionar de forma integral las operaciones de ventas, clientes, paquetes turísticos, pagos, reportes contables, logística, operaciones y marketing. El diseño garantiza integridad de datos, escalabilidad y soporte a los diferentes procesos administrativos, comerciales y operativos de la empresa.

El diseño de la Base de Datos de TaskManager es esencial para asegurar que el sistema funcione de manera eficiente y escalable. En este documento, se presenta la estructura que soporta todas las funciones del sistema, como la gestión de tareas y usuarios. Utilizando PostgreSQL como sistema de gestión de bases de datos, se garantiza la integridad de los datos y la capacidad de manejar grandes volúmenes a medida que el sistema crece

La estructura está pensada para ser sencilla pero flexible, permitiendo su expansión conforme se agreguen más funcionalidades. Además, se ha integrado una capa de seguridad que protege la información y garantiza el cumplimiento de buenas prácticas para el manejo de datos sensibles.

En resumen, este diseño de base de datos proporciona una solución robusta y segura que facilita la administración de las tareas y recursos dentro del sistema TaskManager, mientras que mantiene la eficiencia y la facilidad de uso para los desarrolladores y usuarios finales.

2. Diagrama Entidad-Relación (ERD)

2.1. Entidades y sus Atributos

2.1. Entidades y sus atributos

Usuarios

- id_usuario (PK)
- nombre
- email (único)
- rol *(gerente, ventas, contabilidad, operaciones, logistica, marketing)*
- fecha_registro

Clientes

- **id_cliente (PK)**
- **nombres**
- **apellidos**
- **email**
- **teléfono**
- **documento**

Paquetes

- **id_paquete (PK)**
- **nombre**
- **descripcion**
- **precio**
- **disponibilidad**
- **condiciones**

Ventas

- **id_venta (PK)**
- **id_cliente (FK → Clientes)**
- **id_asesor (FK → Usuarios)**
- **es_grupal**
- **num_personas**
- **estado *(en_proceso, pendiente_pago, finalizado, anulado)***
- **total**
- **fecha_creacion**

Venta_Detalles *(resuelve N:M Ventas–Paquetes)*

- **id_detalle (PK)**
  id_venta (FK → Ventas)
- **id_paquete (FK → Paquetes)**
- **cantidad**
- **precio_unit**
- **subtotal**

Pagos

- **id_pago (PK)**
- **id_venta (FK → Ventas)**
- **monto**
- **estado *(pendiente, parcial, pagado, fallido, devuelto)***
- **medio**
- **referencia**
- **fecha_pago**

Reportes_Contables

- **id_reporte (PK)**
- **fecha**
- **total_ventas**
- **total_pagos**
- **total_deudas**
- **generado_por (FK → Usuarios)**

Activos

- **id_activo (PK)**
- **tipo *(vehiculo, mobiliario, equipo)***
- **codigo (único)**
- **descripcion**
- **ubicacion**
- **estado**
- **responsable_id (FK → Usuarios)**

Mantenimientos

- **id_mantenimiento (PK)**
- **id_activo (FK → Activos)**
- **tipo *(preventivo, correctivo)***
- **estado *(programado, en_ejecucion, culminado, cancelado)***
- **programado_para**
- **iniciado_en**
- **finalizado_en**
- **responsable_id (FK → Usuarios)**
- **notas**

Inventario

- **id_item (PK)**
- **sku (único)**
- **nombre**
- **descripcion**
- **unidad**
- **stock**
- **stock_min**
- **ubicacion**
- **responsable_id (FK → Usuarios)**

Movimientos_Inventario

- **id_mov (PK)**
- **id_item (FK → Inventario)**
- **tipo *(IN, OUT, AJUSTE)***

- **cantidad**
- **motivo**
- **referencia**
- **realizado_por (FK → Usuarios)**
- **fecha_mov**

Campañas

- **id_campaña (PK)**
- **nombre**
- **canal *(social, email, afiliados, interno, otro)***
- **mensaje**
- **fecha_inicio**
- **fecha_fin**
- **presupuesto**
- **creado_por (FK → Usuarios)**

Campaña_Assets

- **id_asset (PK)**
- **id_campaña (FK → Campañas)**
- **tipo *(imagen, video, copy, landing, promo)***
- **url**
- **descripcion**

2.2. Relaciones

A continuación se detallan las relaciones entre las entidades descritas en la base de datos, incluyendo las claves primarias (PK) y las claves foráneas (FK) que aseguran la integridad referencial:

1. Usuarios - Ventas
   - Un usuario (asesor de ventas) puede tener varias ventas asociadas.
   - Relación: Usuarios (id_usuario) → Ventas (id_asesor) (1:N)
   - La clave foránea id_asesor en la tabla Ventas hace referencia a id_usuario en la tabla Usuarios.
2. Clientes - Ventas
   - Un cliente puede realizar múltiples ventas a lo largo del tiempo.
   - Relación: Clientes (id_cliente) → Ventas (id_cliente) (1:N)
   - La clave foránea id_cliente en la tabla Ventas hace referencia a id_cliente en la tabla Clientes.
3. Ventas - Paquetes (a través de Venta_Detalles)
   - Una venta puede involucrar múltiples paquetes turísticos y un paquete puede estar en muchas ventas.
   - Relación: Ventas (id_venta) ↔ Paquetes (id_paquete) (N:M)

- La tabla Venta_Detalles resuelve esta relación muchos a muchos, con las claves foráneas id_venta y id_paquete.

4. Ventas - Pagos
    - Una venta puede tener varios pagos asociados.
    - Relación: Ventas (id_venta) → Pagos (id_venta) (1:N)
    - La clave foránea id_venta en la tabla Pagos hace referencia a id_venta en la tabla Ventas.
5. Usuarios - Reportes_Contables
    - Un usuario (generalmente el encargado de contabilidad) puede generar múltiples reportes contables.
    - Relación: Usuarios (id_usuario) → Reportes_Contables (generado_por) (1:N)
    - La clave foránea generado_por en la tabla Reportes_Contables hace referencia a id_usuario en la tabla Usuarios.
6. Usuarios - Activos
    - Un usuario (encargado de operaciones) puede ser responsable de varios activos (vehículos, equipos, etc.).
    - Relación: Usuarios (id_usuario) → Activos (responsable_id) (1:N)
    - La clave foránea responsable_id en la tabla Activos hace referencia a id_usuario en la tabla Usuarios.
7. Activos - Mantenimientos
    - Un activo (por ejemplo, un vehículo o equipo) puede tener varios mantenimientos.
    - Relación: Activos (id_activo) → Mantenimientos (id_activo) (1:N)
    - La clave foránea id_activo en la tabla Mantenimientos hace referencia a id_activo en la tabla Activos.
8. Usuarios - Mantenimientos
    - Un usuario (responsable de mantenimiento) puede estar asociado a varios mantenimientos.
    - Relación: Usuarios (id_usuario) → Mantenimientos (responsable_id) (1:N)
    - La clave foránea responsable_id en la tabla Mantenimientos hace referencia a id_usuario en la tabla Usuarios.
9. Usuarios - Inventario
    - Un usuario (responsable de logística) puede ser responsable de varios ítems en el inventario.
    - Relación: Usuarios (id_usuario) → Inventario (responsable_id) (1:N)
    - La clave foránea responsable_id en la tabla Inventario hace referencia a id_usuario en la tabla Usuarios.
10. Inventario - Movimientos_Inventario
    - Un ítem de inventario puede tener múltiples movimientos (entradas, salidas, ajustes).
    - Relación: Inventario (id_item) → Movimientos_Inventario (id_item) (1:N)
    - La clave foránea id_item en la tabla Movimientos_Inventario hace referencia a id_item en la tabla Inventario.
11. Usuarios - Movimientos_Inventario
    - Un usuario (responsable de inventario) puede registrar varios movimientos de inventario.

- Relación: Usuarios (id_usuario) → Movimientos_Inventario (realizado_por) (1:N)
- La clave foránea realizado_por en la tabla Movimientos_Inventario hace referencia a id_usuario en la tabla Usuarios.

12. Campañas - Usuarios
    - Un usuario (generalmente del área de marketing) puede crear y gestionar varias campañas publicitarias.
    - Relación: Usuarios (id_usuario) → Campañas (creado_por) (1:N)
    - La clave foránea creado_por en la tabla Campañas hace referencia a id_usuario en la tabla Usuarios.
13. Campañas - Campaña_Assets
    - Una campaña puede tener varios activos asociados (imágenes, videos, copys, etc.).
    - Relación: Campañas (id_campaña) → Campaña_Assets (id_campaña) (1:N)
    - La clave foránea id_campaña en la tabla Campaña_Assets hace referencia a id_campaña en la tabla Campañas.

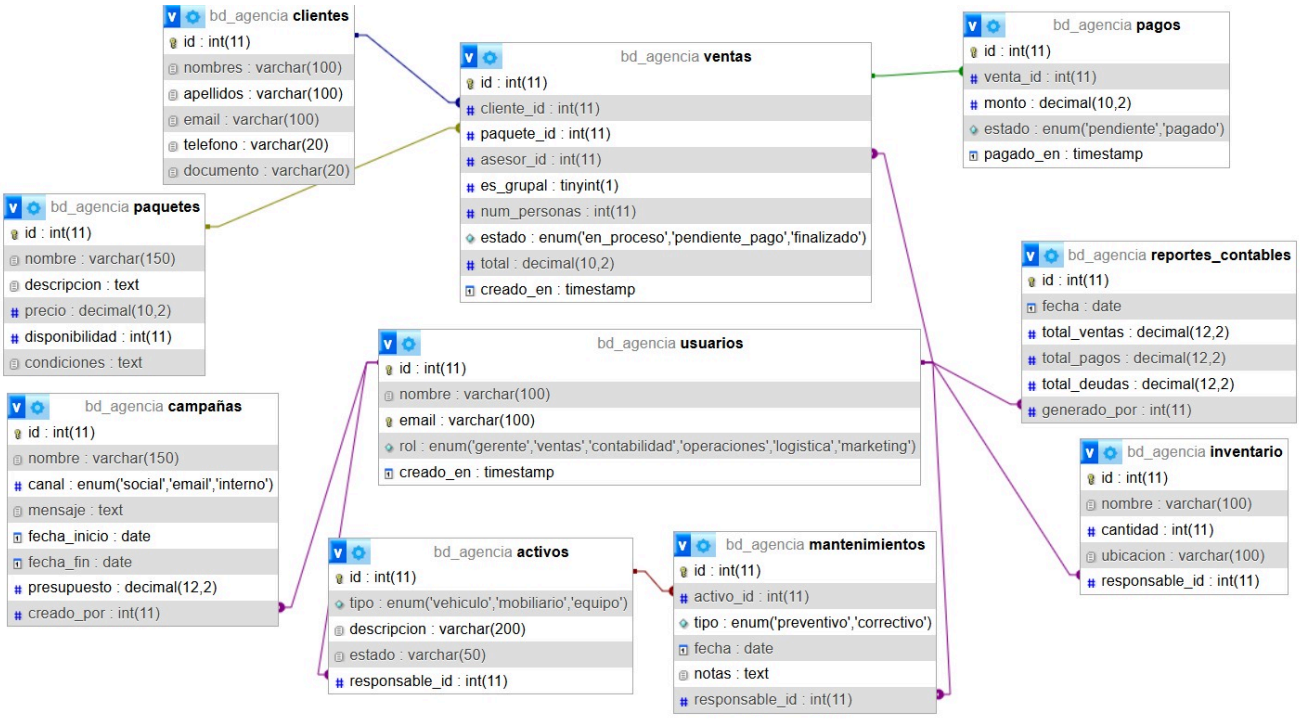
2.3. Reglas y Restricciones

- El **email de usuarios** debe ser único.
- Cada **venta** debe tener un cliente, un paquete y un asesor asignado.
- Los **pagos** dependen de ventas registradas.
- Los **reportes contables** sólo pueden ser generados por usuarios con rol de contabilidad.
- Un **activo o inventario** siempre debe estar bajo responsabilidad de un usuario.
  Se evita duplicidad en asignación de roles críticos (ej. un mismo pago no puede repetirse en la misma venta).

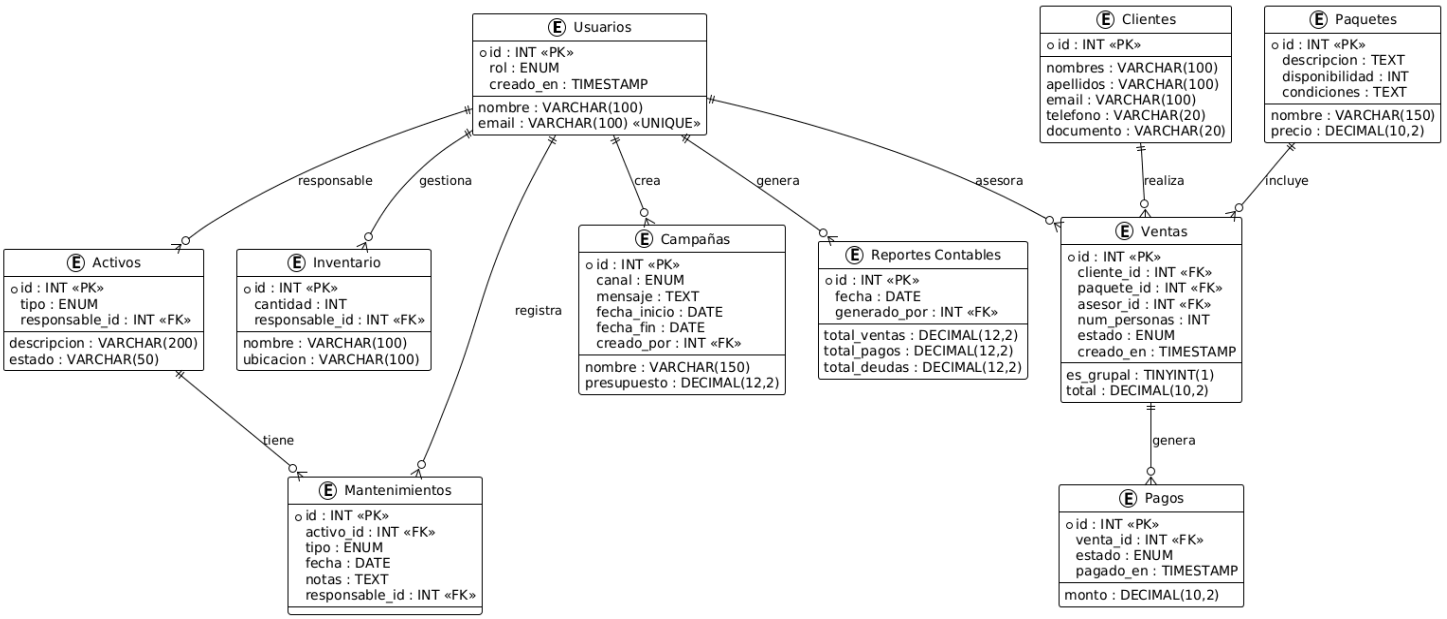
2.4. Diagrama de Base de Datos (E/R)

- **Usuarios (1) → (N) Ventas (como asesores).**
- **Clientes (1) → (N) Ventas.**
- **Paquetes (1) → (N) Ventas.**
- **Ventas (1) → (N) Pagos.**

- **Usuarios (1) → (N) Reportes Contables.**
- **Usuarios (1) → (N) Activos.**
- **Activos (1) → (N) Mantenimientos.**
- **Usuarios (1) → (N) Inventario.**
- **Usuarios (1) → (N) Campañas.**



3. Diagrama Relacional



4. Diccionario de Datos

Tabla Usuarios

Campo	Tipo de Dato	Descripción	Restricciones
id	INT	Identificador único del usuario del sistema.	PRIMARY KEY, AUTO_INCREMENT
nombre	VARCHAR(100)	Nombre completo del usuario.	NOT NULL
email	VARCHAR(100)	Dirección de correo electrónico del usuario.	UNIQUE, NOT NULL
rol	ENUM(...)	Rol del usuario en la agencia (gerente, ventas, contabilidad, operaciones, logística, marketing).	NOT NULL
creado_en	TIMESTAMP	Fecha y hora de creación del registro.	DEFAULT CURRENT_TIMESTAMP

Tabla Clientes

Campo	Tipo de Dato	Descripción	Restricciones
id	INT	Identificador único del cliente.	PRIMARY KEY, AUTO_INCREMENT
nombres	VARCHAR(100)	Nombres del cliente.	NOT NULL
apellidos	VARCHAR(100)	Apellidos del cliente.	NULLABLE
email	VARCHAR(100)	Correo electrónico del cliente.	NULLABLE
telefono	VARCHAR(20)	Número de teléfono del cliente.	NULLABLE
documento	VARCHAR(20)	Documento de identidad del cliente.	NULLABLE

Tabla Paquetes

Campo	Tipo de Dato	Descripción	Restricciones
id	INT	Identificador único del paquete turístico.	PRIMARY KEY, AUTO_INCREMENT
nombre	VARCHAR(150)	Nombre del paquete turístico.	NOT NULL
descripcion	TEXT	Descripción detallada del paquete.	NULLABLE
precio	DECIMAL(10,2)	Precio del paquete turístico.	NOT NULL
disponibilidad	INT	Número de plazas o cupos disponibles.	NOT NULL
condiciones	TEXT	Condiciones o restricciones del paquete.	NULLABLE

Tabla Ventas

Campo	Tipo de Dato	Descripción	Restricciones
id	INT	Identificador único de la venta.	PRIMARY KEY, AUTO_INCREMENT
cliente_id	INT	Cliente que realiza la compra.	FOREIGN KEY (Clientes.id), NOT NULL

paquete_id	INT	Paquete adquirido en la venta.	FOREIGN KEY (Paquetes.id), NOT NULL
asesor_id	INT	Asesor de ventas responsable.	FOREIGN KEY (Usuarios.id), NOT NULL
es_grupal	TINYINT(1)	Indica si la venta es grupal (1) o individual (0).	DEFAULT 0
num_personas	INT	Número de personas incluidas en la venta grupal.	NULLABLE
estado	ENUM(...)	Estado de la venta (en_proceso, pendiente_pago, finalizado).	DEFAULT 'en_proceso'
total	DECIMAL(10,2)	Monto total de la venta.	NULLABLE
creado_en	TIMESTAMP	Fecha y hora de registro de la venta.	DEFAULT CURRENT_TIMESTAMP

Tabla Pagos

Campo	Tipo de Dato	Descripción	Restricciones
id	INT	Identificador único del pago.	PRIMARY KEY, AUTO_INCREMENT
venta_id	INT	Venta a la que pertenece el pago.	FOREIGN KEY (Ventas.id), NOT NULL
monto	DECIMAL(10,2)	Monto pagado.	NOT NULL
estado	ENUM(...)	Estado del pago (pendiente, pagado).	DEFAULT 'pendiente'
pagado_en	TIMESTAMP	Fecha y hora en que se realizó el pago.	NULLABLE

Tabla Reportes Contables

Campo	Tipo de Dato	Descripción	Restricciones
id	INT	Identificador único del reporte contable.	PRIMARY KEY, AUTO_INCREMENT

fecha	DATE	Fecha del reporte.	NOT NULL
total_ventas	DECIMAL(12,2)	Monto total de ventas registradas.	NULLABLE
total_pagos	DECIMAL(12,2)	Monto total de pagos recibidos.	NULLABLE
total_deudas	DECIMAL(12,2)	Monto total de deudas pendientes.	NULLABLE
generado_por	INT	Usuario del área contable que genera el reporte.	FOREIGN KEY (Usuarios.id)

Tabla Campaña

Campo	Tipo de Dato	Descripción	Restricciones
id	INT	Identificador único de la campaña.	PRIMARY KEY, AUTO_INCREMENT
nombre	VARCHAR(150)	Nombre de la campaña de marketing.	NOT NULL
canal	ENUM(...)	Canal de difusión (social, email, interno).	NOT NULL
mensaje	TEXT	Mensaje de la campaña.	NULLABLE
fecha_inicio	DATE	Fecha de inicio de la campaña.	NULLABLE
fecha_fin	DATE	Fecha de finalización de la campaña.	NULLABLE
presupuesto	DECIMAL(12,2)	Presupuesto asignado a la campaña.	NULLABLE
creado_por	INT	Usuario creador de la campaña.	FOREIGN KEY (Usuarios.id)

4.1. Relaciones y Restricciones:

La estructura de la base de datos de TaskManager está diseñada para garantizar la integridad de los datos mediante relaciones claras entre las entidades, así como restricciones que aseguran la consistencia y seguridad de la información almacenada. A continuación, se detallan las relaciones y restricciones implementadas en el sistema:

Relaciones

1. Usuarios ↔ Ventas
   - Un usuario (asesor de ventas) puede estar relacionado con varias ventas, pero una venta solo puede ser realizada por un único usuario.

- Restricción: La clave foránea id_asesor en la tabla Ventas debe hacer referencia a un id_usuario válido en la tabla Usuarios.

2. Clientes ↔ Ventas
   - Un cliente puede realizar múltiples ventas, pero una venta está asociada a un solo cliente.
   - Restricción: La clave foránea id_cliente en la tabla Ventas debe hacer referencia a un id_cliente válido en la tabla Clientes.
3. Ventas ↔ Paquetes (a través de Venta_Detalles)
   - Una venta puede involucrar varios paquetes, y un paquete puede estar asociado a múltiples ventas. La relación muchos a muchos se resuelve a través de la tabla Venta_Detalles.
   - Restricción: Las claves foráneas id_venta en Venta_Detalles deben hacer referencia a un id_venta válido en la tabla Ventas, y las claves foráneas id_paquete en Venta_Detalles deben hacer referencia a un id_paquete válido en la tabla Paquetes.
4. Ventas ↔ Pagos
   - Una venta puede tener varios pagos asociados (por ejemplo, pagos parciales o completos), pero cada pago se refiere a una sola venta.
   - Restricción: La clave foránea id_venta en la tabla Pagos debe hacer referencia a un id_venta válido en la tabla Ventas.
5. Usuarios ↔ Reportes_Contables
   - Un usuario (generalmente el encargado de contabilidad) puede generar varios reportes contables. Cada reporte contable está asociado a un único usuario.
   - Restricción: La clave foránea generado_por en la tabla Reportes_Contables debe hacer referencia a un id_usuario válido en la tabla Usuarios.
6. Usuarios ↔ Activos
   - Un usuario (responsable de operaciones) puede ser responsable de varios activos. Cada activo tiene asignado un único usuario responsable.
   - Restricción: La clave foránea responsable_id en la tabla Activos debe hacer referencia a un id_usuario válido en la tabla Usuarios.
7. Activos ↔ Mantenimientos
   - Un activo puede tener múltiples mantenimientos (por ejemplo, mantenimiento preventivo o correctivo), pero cada mantenimiento está asociado a un solo activo.
   - Restricción: La clave foránea id_activo en la tabla Mantenimientos debe hacer referencia a un id_activo válido en la tabla Activos.
8. Usuarios ↔ Mantenimientos
   - Un usuario (encargado de mantenimiento) puede estar relacionado con múltiples mantenimientos. Cada mantenimiento es realizado por un único usuario responsable.
   - Restricción: La clave foránea responsable_id en la tabla Mantenimientos debe hacer referencia a un id_usuario válido en la tabla Usuarios.
9. Usuarios ↔ Inventario
   - Un usuario (responsable de logística) puede estar encargado de varios ítems en el inventario. Cada ítem tiene asignado un usuario responsable.
   - Restricción: La clave foránea responsable_id en la tabla Inventario debe hacer referencia a un id_usuario válido en la tabla Usuarios.
10. Inventario ↔ Movimientos_Inventario
    - Un ítem de inventario puede tener múltiples movimientos (entradas, salidas o ajustes), pero cada movimiento está asociado a un solo ítem de inventario.

- Restricción: La clave foránea id_item en la tabla Movimientos_Inventario debe hacer referencia a un id_item válido en la tabla Inventario.

11. Usuarios ↔ Movimientos_Inventario
    - Un usuario (responsable de inventario) puede registrar varios movimientos en el inventario. Cada movimiento está realizado por un único usuario.
    - Restricción: La clave foránea realizado_por en la tabla Movimientos_Inventario debe hacer referencia a un id_usuario válido en la tabla Usuarios.
12. Usuarios ↔ Campañas
    - Un usuario (del área de marketing) puede crear y gestionar varias campañas publicitarias. Cada campaña tiene un único usuario creador.
    - Restricción: La clave foránea creado_por en la tabla Campañas debe hacer referencia a un id_usuario válido en la tabla Usuarios.
13. Campañas ↔ Campaña_Assets
    - Una campaña puede tener varios activos asociados (como imágenes, videos, copys, etc.), pero cada activo está relacionado con una sola campaña.
    - Restricción: La clave foránea id_campaña en la tabla Campaña_Assets debe hacer referencia a un id_campaña válido en la tabla Campañas.

5. Consideraciones de Escalabilidad

Uso de **índices** en campos de búsqueda frecuente (email, cliente_id, paquete_id).

ON DELETE CASCADE en claves foráneas críticas (ej. pagos al eliminar una venta).

La base de datos está diseñada para **escalar horizontalmente** en caso de alto volumen de ventas o clientes.

División lógica entre áreas: ventas, contabilidad, logística, marketing, lo que facilita microservicios futuros.

6. Conclusiones

El modelo relacional de la **Agencia de Viajes** permite cubrir las áreas críticas de gestión:

- Comercial (clientes, ventas, pagos).
- Contable (reportes financieros).
- Operativa (activos, mantenimientos, inventario).
- Estratégica (campañas de marketing).

Con este diseño, la base de datos soporta la **integridad referencial**, garantiza seguridad en la trazabilidad de operaciones y está preparada para crecer en complejidad sin perder consistencia.

ANEXO. BASE DE DATOS

```
-- ========================================================
-- BASE DE DATOS AGENCIA DE VIAJES - VERSIÓN PROFESIONAL
-- ========================================================

USE agencia_viajes;

-- Configuración de motor de almacenamiento y codificación
SET foreign_key_checks = 0;

-- ========================================================
-- TABLAS DE CONFIGURACIÓN Y CATÁLOGOS
-- ========================================================

-- Catálogo de países
CREATE TABLE IF NOT EXISTS paises (
    id INT AUTO_INCREMENT PRIMARY KEY,
    codigo_iso CHAR(2) NOT NULL UNIQUE,
    nombre VARCHAR(100) NOT NULL,
    activo BOOLEAN DEFAULT TRUE,
    INDEX idx_codigo_iso (codigo_iso)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

-- Catálogo de ciudades
CREATE TABLE IF NOT EXISTS ciudades (
```

```
    id INT AUTO_INCREMENT PRIMARY KEY,
    pais_id INT NOT NULL,
    nombre VARCHAR(100) NOT NULL,
    activo BOOLEAN DEFAULT TRUE,
    INDEX idx_pais_id (pais_id),
    FOREIGN KEY (pais_id) REFERENCES paises(id) ON DELETE RESTRICT
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

-- ========================================================
-- GESTIÓN DE USUARIOS Y EMPLEADOS
-- ========================================================

CREATE TABLE IF NOT EXISTS departamentos (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nombre VARCHAR(50) NOT NULL UNIQUE,
    descripcion TEXT,
    activo BOOLEAN DEFAULT TRUE,
    creado_en TIMESTAMP DEFAULT CURRENT_TIMESTAMP
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

CREATE TABLE IF NOT EXISTS usuarios (
    id INT AUTO_INCREMENT PRIMARY KEY,
    codigo_empleado VARCHAR(20) NOT NULL UNIQUE,
    nombres VARCHAR(100) NOT NULL,
    apellidos VARCHAR(100) NOT NULL,
```

```
    email VARCHAR(100) NOT NULL UNIQUE,

    telefono VARCHAR(20),

    departamento_id INT NOT NULL,

                                                                                rol
ENUM('gerente','supervisor','ventas','contabilidad','operaciones','logistica','marketing')
NOT NULL,

    password_hash VARCHAR(255) NOT NULL,

    activo BOOLEAN DEFAULT TRUE,

    ultimo_acceso TIMESTAMP NULL,

    creado_en TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

     actualizado_en TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,

    INDEX idx_email (email),

    INDEX idx_departamento (departamento_id),

    INDEX idx_rol (rol),

    FOREIGN KEY (departamento_id) REFERENCES departamentos(id) ON DELETE
RESTRICT

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;



-- ======================================================

-- GESTIÓN DE CLIENTES

-- ======================================================



CREATE TABLE IF NOT EXISTS tipos_documento (

    id INT AUTO_INCREMENT PRIMARY KEY,

    codigo VARCHAR(10) NOT NULL UNIQUE,

    descripcion VARCHAR(50) NOT NULL,
```

```
    activo BOOLEAN DEFAULT TRUE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;


CREATE TABLE IF NOT EXISTS clientes (
    id INT AUTO_INCREMENT PRIMARY KEY,
    codigo_cliente VARCHAR(20) NOT NULL UNIQUE,
    nombres VARCHAR(100) NOT NULL,
    apellidos VARCHAR(100) NOT NULL,
    email VARCHAR(100),
    telefono VARCHAR(20),
    telefono_alternativo VARCHAR(20),
    tipo_documento_id INT,
    numero_documento VARCHAR(20),
    fecha_nacimiento DATE,
    direccion TEXT,
    ciudad_id INT,
    preferencias_viaje TEXT,
    activo BOOLEAN DEFAULT TRUE,
    creado_en TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    actualizado_en TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,
    INDEX idx_email (email),
    INDEX idx_documento (tipo_documento_id, numero_documento),
    INDEX idx_codigo_cliente (codigo_cliente),
    FOREIGN KEY (tipo_documento_id) REFERENCES tipos_documento(id) ON
DELETE RESTRICT,
```

```
    FOREIGN KEY (ciudad_id) REFERENCES ciudades(id) ON DELETE RESTRICT
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

-- ========================================================
-- GESTIÓN DE PRODUCTOS TURÍSTICOS
-- ========================================================

CREATE TABLE IF NOT EXISTS categorias_paquetes (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL UNIQUE,
    descripcion TEXT,
    activo BOOLEAN DEFAULT TRUE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

CREATE TABLE IF NOT EXISTS proveedores (
    id INT AUTO_INCREMENT PRIMARY KEY,
    codigo_proveedor VARCHAR(20) NOT NULL UNIQUE,
    razon_social VARCHAR(200) NOT NULL,
    ruc VARCHAR(20),
    telefono VARCHAR(20),
    email VARCHAR(100),
    direccion TEXT,
    ciudad_id INT,
    activo BOOLEAN DEFAULT TRUE,
    creado_en TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
```

```
    INDEX idx_codigo_proveedor (codigo_proveedor),
    FOREIGN KEY (ciudad_id) REFERENCES ciudades(id) ON DELETE RESTRICT
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

CREATE TABLE IF NOT EXISTS paquetes (
    id INT AUTO_INCREMENT PRIMARY KEY,
    codigo_paquete VARCHAR(30) NOT NULL UNIQUE,
    nombre VARCHAR(200) NOT NULL,
    descripcion TEXT,
    categoria_id INT NOT NULL,
    duracion_dias INT NOT NULL DEFAULT 1,
    precio_base DECIMAL(12,2) NOT NULL,
    precio_venta DECIMAL(12,2) NOT NULL,
    cupo_minimo INT DEFAULT 1,
    cupo_maximo INT DEFAULT 50,
    disponibilidad_actual INT NOT NULL DEFAULT 0,
    fecha_inicio_vigencia DATE,
    fecha_fin_vigencia DATE,
    incluye TEXT,
    no_incluye TEXT,
    condiciones_cancelacion TEXT,
    proveedor_id INT,
    activo BOOLEAN DEFAULT TRUE,
    creado_por INT NOT NULL,
    creado_en TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
```

```
    actualizado_en TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
    INDEX idx_codigo_paquete (codigo_paquete),
    INDEX idx_categoria (categoria_id),
    INDEX idx_vigencia (fecha_inicio_vigencia, fecha_fin_vigencia),
    INDEX idx_precios (precio_base, precio_venta),
    FOREIGN KEY (categoria_id) REFERENCES categorias_paquetes(id) ON DELETE RESTRICT,
    FOREIGN KEY (proveedor_id) REFERENCES proveedores(id) ON DELETE RESTRICT,
    FOREIGN KEY (creado_por) REFERENCES usuarios(id) ON DELETE RESTRICT
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

-- Destinos incluidos en cada paquete
CREATE TABLE IF NOT EXISTS paquete_destinos (
    id INT AUTO_INCREMENT PRIMARY KEY,
    paquete_id INT NOT NULL,
    ciudad_id INT NOT NULL,
    orden_visita INT DEFAULT 1,
    dias_estancia INT DEFAULT 1,
    notas TEXT,
    FOREIGN KEY (paquete_id) REFERENCES paquetes(id) ON DELETE CASCADE,
    FOREIGN KEY (ciudad_id) REFERENCES ciudades(id) ON DELETE RESTRICT,
    UNIQUE KEY unique_paquete_ciudad (paquete_id, ciudad_id)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

```
-- ================================================

-- GESTIÓN DE VENTAS Y COTIZACIONES

-- ================================================


CREATE TABLE IF NOT EXISTS cotizaciones (

    id INT AUTO_INCREMENT PRIMARY KEY,

    numero_cotizacion VARCHAR(30) NOT NULL UNIQUE,

    cliente_id INT NOT NULL,

    asesor_id INT NOT NULL,

    fecha_cotizacion DATE NOT NULL,

    fecha_vencimiento DATE NOT NULL,

    subtotal DECIMAL(12,2) NOT NULL DEFAULT 0,

    descuento DECIMAL(12,2) DEFAULT 0,

    impuestos DECIMAL(12,2) DEFAULT 0,

    total DECIMAL(12,2) NOT NULL DEFAULT 0,

        estado ENUM('borrador','enviada','aprobada','rechazada','vencida') DEFAULT
'borrador',

    observaciones TEXT,

    creado_en TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

     actualizado_en TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,

    INDEX idx_numero_cotizacion (numero_cotizacion),

    INDEX idx_cliente (cliente_id),

    INDEX idx_asesor (asesor_id),

    INDEX idx_estado (estado),

    INDEX idx_fecha_vencimiento (fecha_vencimiento),
```

```
    FOREIGN KEY (cliente_id) REFERENCES clientes(id) ON DELETE RESTRICT,
    FOREIGN KEY (asesor_id) REFERENCES usuarios(id) ON DELETE RESTRICT
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

CREATE TABLE IF NOT EXISTS cotizacion_detalles (
    id INT AUTO_INCREMENT PRIMARY KEY,
    cotizacion_id INT NOT NULL,
    paquete_id INT NOT NULL,
    cantidad_personas INT NOT NULL DEFAULT 1,
    precio_unitario DECIMAL(12,2) NOT NULL,
    descuento_unitario DECIMAL(12,2) DEFAULT 0,
    subtotal DECIMAL(12,2) NOT NULL,
    observaciones TEXT,
    FOREIGN KEY (cotizacion_id) REFERENCES cotizaciones(id) ON DELETE CASCADE,
    FOREIGN KEY (paquete_id) REFERENCES paquetes(id) ON DELETE RESTRICT
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

CREATE TABLE IF NOT EXISTS ventas (
    id INT AUTO_INCREMENT PRIMARY KEY,
    numero_venta VARCHAR(30) NOT NULL UNIQUE,
    cotizacion_id INT,
    cliente_id INT NOT NULL,
    asesor_id INT NOT NULL,
    fecha_venta DATE NOT NULL,
    fecha_viaje DATE,
```

```
    numero_personas INT NOT NULL DEFAULT 1,
    subtotal DECIMAL(12,2) NOT NULL DEFAULT 0,
    descuento DECIMAL(12,2) DEFAULT 0,
    impuestos DECIMAL(12,2) DEFAULT 0,
    total DECIMAL(12,2) NOT NULL DEFAULT 0,
      estado ENUM('confirmada','en_proceso','pagada','cancelada','finalizada') DEFAULT 'confirmada',
    tipo_venta ENUM('individual','grupal','corporativa') DEFAULT 'individual',
    observaciones TEXT,
    creado_en TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
      actualizado_en TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
    INDEX idx_numero_venta (numero_venta),
    INDEX idx_cliente (cliente_id),
    INDEX idx_asesor (asesor_id),
    INDEX idx_estado (estado),
    INDEX idx_fecha_venta (fecha_venta),
    INDEX idx_fecha_viaje (fecha_viaje),
      FOREIGN KEY (cotizacion_id) REFERENCES cotizaciones(id) ON DELETE RESTRICT,
    FOREIGN KEY (cliente_id) REFERENCES clientes(id) ON DELETE RESTRICT,
    FOREIGN KEY (asesor_id) REFERENCES usuarios(id) ON DELETE RESTRICT
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

CREATE TABLE IF NOT EXISTS venta_detalles (
    id INT AUTO_INCREMENT PRIMARY KEY,
    venta_id INT NOT NULL,
```

```
    paquete_id INT NOT NULL,
    cantidad_personas INT NOT NULL DEFAULT 1,
    precio_unitario DECIMAL(12,2) NOT NULL,
    descuento_unitario DECIMAL(12,2) DEFAULT 0,
    subtotal DECIMAL(12,2) NOT NULL,
    observaciones TEXT,
    FOREIGN KEY (venta_id) REFERENCES ventas(id) ON DELETE CASCADE,
    FOREIGN KEY (paquete_id) REFERENCES paquetes(id) ON DELETE RESTRICT
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

-- ========================================================
-- GESTIÓN DE PAGOS Y FINANZAS
-- ========================================================

CREATE TABLE IF NOT EXISTS metodos_pago (
    id INT AUTO_INCREMENT PRIMARY KEY,
    codigo VARCHAR(10) NOT NULL UNIQUE,
    descripcion VARCHAR(50) NOT NULL,
    requiere_referencia BOOLEAN DEFAULT FALSE,
    activo BOOLEAN DEFAULT TRUE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

CREATE TABLE IF NOT EXISTS pagos (
    id INT AUTO_INCREMENT PRIMARY KEY,
    numero_pago VARCHAR(30) NOT NULL UNIQUE,
```

```
    venta_id INT NOT NULL,

    metodo_pago_id INT NOT NULL,

    monto DECIMAL(12,2) NOT NULL,

    fecha_programada DATE,

    fecha_pago DATE,

    numero_referencia VARCHAR(100),

     estado ENUM('programado','recibido','confirmado','rechazado','anulado') DEFAULT
'programado',

    observaciones TEXT,

    procesado_por INT,

    creado_en TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

     actualizado_en TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,

    INDEX idx_numero_pago (numero_pago),

    INDEX idx_venta (venta_id),

    INDEX idx_estado (estado),

    INDEX idx_fecha_programada (fecha_programada),

    INDEX idx_fecha_pago (fecha_pago),

    FOREIGN KEY (venta_id) REFERENCES ventas(id) ON DELETE RESTRICT,

     FOREIGN KEY (metodo_pago_id) REFERENCES metodos_pago(id) ON DELETE
RESTRICT,

        FOREIGN KEY (procesado_por) REFERENCES usuarios(id) ON DELETE
RESTRICT

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;



-- ====================================================

-- GESTIÓN DE REPORTES Y CONTABILIDAD
```

```
-- ================================================

CREATE TABLE IF NOT EXISTS tipos_reporte (
    id INT AUTO_INCREMENT PRIMARY KEY,
    codigo VARCHAR(20) NOT NULL UNIQUE,
    nombre VARCHAR(100) NOT NULL,
    descripcion TEXT,
    activo BOOLEAN DEFAULT TRUE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

CREATE TABLE IF NOT EXISTS reportes_contables (
    id INT AUTO_INCREMENT PRIMARY KEY,
    numero_reporte VARCHAR(30) NOT NULL UNIQUE,
    tipo_reporte_id INT NOT NULL,
    periodo_inicio DATE NOT NULL,
    periodo_fin DATE NOT NULL,
    total_ventas DECIMAL(15,2) DEFAULT 0,
    total_pagos_recibidos DECIMAL(15,2) DEFAULT 0,
    total_cuentas_por_cobrar DECIMAL(15,2) DEFAULT 0,
    total_gastos DECIMAL(15,2) DEFAULT 0,
    utilidad_bruta DECIMAL(15,2) DEFAULT 0,
    observaciones TEXT,
    generado_por INT NOT NULL,
    aprobado_por INT,
    fecha_generacion TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
```

```
    fecha_aprobacion TIMESTAMP NULL,

    INDEX idx_numero_reporte (numero_reporte),

    INDEX idx_periodo (periodo_inicio, periodo_fin),

    INDEX idx_tipo_reporte (tipo_reporte_id),

      FOREIGN KEY (tipo_reporte_id) REFERENCES tipos_reporte(id) ON DELETE RESTRICT,

        FOREIGN KEY (generado_por) REFERENCES usuarios(id) ON DELETE RESTRICT,

        FOREIGN KEY (aprobado_por) REFERENCES usuarios(id) ON DELETE RESTRICT

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;


-- ====================================================

-- GESTIÓN DE ACTIVOS Y OPERACIONES

-- ====================================================


CREATE TABLE IF NOT EXISTS tipos_activo (

    id INT AUTO_INCREMENT PRIMARY KEY,

    codigo VARCHAR(10) NOT NULL UNIQUE,

    descripcion VARCHAR(50) NOT NULL,

    vida_util_años INT DEFAULT 5,

    activo BOOLEAN DEFAULT TRUE

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;


CREATE TABLE IF NOT EXISTS activos (

    id INT AUTO_INCREMENT PRIMARY KEY,

    codigo_activo VARCHAR(30) NOT NULL UNIQUE,
```

```
    tipo_activo_id INT NOT NULL,

    descripcion VARCHAR(200) NOT NULL,

    marca VARCHAR(100),

    modelo VARCHAR(100),

    numero_serie VARCHAR(100),

    fecha_adquisicion DATE,

    valor_adquisicion DECIMAL(12,2),

        estado ENUM('operativo','mantenimiento','dañado','dado_de_baja') DEFAULT
'operativo',

    ubicacion_actual VARCHAR(100),

    responsable_id INT,

    observaciones TEXT,

    creado_en TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

     actualizado_en TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,

    INDEX idx_codigo_activo (codigo_activo),

    INDEX idx_tipo_activo (tipo_activo_id),

    INDEX idx_estado (estado),

    INDEX idx_responsable (responsable_id),

      FOREIGN KEY (tipo_activo_id) REFERENCES tipos_activo(id) ON DELETE
RESTRICT,

       FOREIGN KEY (responsable_id) REFERENCES usuarios(id) ON DELETE
RESTRICT

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;



CREATE TABLE IF NOT EXISTS mantenimientos (

    id INT AUTO_INCREMENT PRIMARY KEY,
```

```
    numero_mantenimiento VARCHAR(30) NOT NULL UNIQUE,
    activo_id INT NOT NULL,
    tipo ENUM('preventivo','correctivo','predictivo') NOT NULL,
    fecha_programada DATE NOT NULL,
    fecha_ejecutada DATE,
    descripcion TEXT NOT NULL,
    costo DECIMAL(10,2) DEFAULT 0,
    proveedor_servicio VARCHAR(200),
    estado ENUM('programado','en_progreso','completado','cancelado') DEFAULT 'programado',
    responsable_id INT NOT NULL,
    observaciones TEXT,
    creado_en TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    INDEX idx_numero_mantenimiento (numero_mantenimiento),
    INDEX idx_activo (activo_id),
    INDEX idx_tipo (tipo),
    INDEX idx_estado (estado),
    INDEX idx_fecha_programada (fecha_programada),
    FOREIGN KEY (activo_id) REFERENCES activos(id) ON DELETE RESTRICT,
    FOREIGN KEY (responsable_id) REFERENCES usuarios(id) ON DELETE RESTRICT
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

-- ================================================
-- GESTIÓN DE INVENTARIO Y LOGÍSTICA
-- ================================================
```

```
CREATE TABLE IF NOT EXISTS categorias_inventario (
    id INT AUTO_INCREMENT PRIMARY KEY,
    codigo VARCHAR(10) NOT NULL UNIQUE,
    nombre VARCHAR(100) NOT NULL,
    descripcion TEXT,
    activo BOOLEAN DEFAULT TRUE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

CREATE TABLE IF NOT EXISTS almacenes (
    id INT AUTO_INCREMENT PRIMARY KEY,
    codigo_almacen VARCHAR(20) NOT NULL UNIQUE,
    nombre VARCHAR(100) NOT NULL,
    direccion TEXT,
    ciudad_id INT,
    responsable_id INT,
    activo BOOLEAN DEFAULT TRUE,
    INDEX idx_codigo_almacen (codigo_almacen),
    FOREIGN KEY (ciudad_id) REFERENCES ciudades(id) ON DELETE RESTRICT,
        FOREIGN KEY (responsable_id) REFERENCES usuarios(id) ON DELETE RESTRICT
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

CREATE TABLE IF NOT EXISTS productos_inventario (
    id INT AUTO_INCREMENT PRIMARY KEY,
    codigo_producto VARCHAR(30) NOT NULL UNIQUE,
```

```
    nombre VARCHAR(150) NOT NULL,
    descripcion TEXT,
    categoria_id INT NOT NULL,
    unidad_medida VARCHAR(20) DEFAULT 'UNIDAD',
    stock_minimo INT DEFAULT 0,
    stock_maximo INT DEFAULT 1000,
    precio_unitario DECIMAL(10,2) DEFAULT 0,
    activo BOOLEAN DEFAULT TRUE,
    creado_en TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    INDEX idx_codigo_producto (codigo_producto),
    INDEX idx_categoria (categoria_id),
        FOREIGN KEY (categoria_id) REFERENCES categorias_inventario(id) ON DELETE RESTRICT
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

CREATE TABLE IF NOT EXISTS inventario (
    id INT AUTO_INCREMENT PRIMARY KEY,
    producto_id INT NOT NULL,
    almacen_id INT NOT NULL,
    cantidad_actual INT NOT NULL DEFAULT 0,
     fecha_ultimo_movimiento TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
    responsable_id INT,
        FOREIGN KEY (producto_id) REFERENCES productos_inventario(id) ON DELETE RESTRICT,
        FOREIGN KEY (almacen_id) REFERENCES almacenes(id) ON DELETE RESTRICT,
```

```
    FOREIGN KEY (responsable_id) REFERENCES usuarios(id) ON DELETE RESTRICT,
    UNIQUE KEY unique_producto_almacen (producto_id, almacen_id)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

CREATE TABLE IF NOT EXISTS movimientos_inventario (
    id INT AUTO_INCREMENT PRIMARY KEY,
    numero_movimiento VARCHAR(30) NOT NULL UNIQUE,
    producto_id INT NOT NULL,
    almacen_id INT NOT NULL,
    tipo_movimiento ENUM('entrada','salida','ajuste','transferencia') NOT NULL,
    cantidad INT NOT NULL,
    motivo VARCHAR(200),
    documento_referencia VARCHAR(100),
    responsable_id INT NOT NULL,
    fecha_movimiento TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    INDEX idx_numero_movimiento (numero_movimiento),
    INDEX idx_producto (producto_id),
    INDEX idx_almacen (almacen_id),
    INDEX idx_tipo_movimiento (tipo_movimiento),
    INDEX idx_fecha_movimiento (fecha_movimiento),
    FOREIGN KEY (producto_id) REFERENCES productos_inventario(id) ON DELETE RESTRICT,
    FOREIGN KEY (almacen_id) REFERENCES almacenes(id) ON DELETE RESTRICT,
    FOREIGN KEY (responsable_id) REFERENCES usuarios(id) ON DELETE RESTRICT
```

```
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

-- ========================================================
-- GESTIÓN DE MARKETING Y CAMPAÑAS
-- ========================================================

CREATE TABLE IF NOT EXISTS canales_marketing (
    id INT AUTO_INCREMENT PRIMARY KEY,
    codigo VARCHAR(10) NOT NULL UNIQUE,
    nombre VARCHAR(50) NOT NULL,
    descripcion TEXT,
    activo BOOLEAN DEFAULT TRUE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

CREATE TABLE IF NOT EXISTS campañas (
    id INT AUTO_INCREMENT PRIMARY KEY,
    codigo_campaña VARCHAR(30) NOT NULL UNIQUE,
    nombre VARCHAR(200) NOT NULL,
    objetivo TEXT,
    canal_id INT NOT NULL,
    mensaje TEXT,
    fecha_inicio DATE NOT NULL,
    fecha_fin DATE NOT NULL,
    presupuesto_asignado DECIMAL(12,2) DEFAULT 0,
    presupuesto_ejecutado DECIMAL(12,2) DEFAULT 0,
```

```
        estado ENUM('planificada','activa','pausada','finalizada','cancelada') DEFAULT 'planificada',

    publico_objetivo TEXT,

    metricas_objetivo JSON,

    resultados_obtenidos JSON,

    creado_por INT NOT NULL,

    aprobado_por INT,

    creado_en TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

     actualizado_en TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,

    INDEX idx_codigo_campaña (codigo_campaña),

    INDEX idx_canal (canal_id),

    INDEX idx_estado (estado),

    INDEX idx_fechas (fecha_inicio, fecha_fin),

      FOREIGN KEY (canal_id) REFERENCES canales_marketing(id) ON DELETE RESTRICT,

        FOREIGN KEY (creado_por) REFERENCES usuarios(id) ON DELETE RESTRICT,

        FOREIGN KEY (aprobado_por) REFERENCES usuarios(id) ON DELETE RESTRICT

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;



-- ==========================================================

-- TRIGGERS PARA MANTENER CONSISTENCIA DE DATOS

-- ==========================================================



DELIMITER //
```

```
-- Trigger para actualizar el stock cuando se registra un movimiento de inventario
CREATE TRIGGER tr_actualizar_stock_inventario
    AFTER INSERT ON movimientos_inventario
    FOR EACH ROW
BEGIN
    IF NEW.tipo_movimiento IN ('entrada', 'ajuste') THEN
        UPDATE inventario
        SET cantidad_actual = cantidad_actual + NEW.cantidad,
            fecha_ultimo_movimiento = NEW.fecha_movimiento
        WHERE producto_id = NEW.producto_id AND almacen_id = NEW.almacen_id;
    ELSEIF NEW.tipo_movimiento = 'salida' THEN
        UPDATE inventario
        SET cantidad_actual = cantidad_actual - NEW.cantidad,
            fecha_ultimo_movimiento = NEW.fecha_movimiento
        WHERE producto_id = NEW.producto_id AND almacen_id = NEW.almacen_id;
    END IF;
END//

-- Trigger para actualizar totales de cotización cuando se modifican los detalles
CREATE TRIGGER tr_actualizar_total_cotizacion
    AFTER INSERT ON cotizacion_detalles
    FOR EACH ROW
BEGIN
    UPDATE cotizaciones
    SET subtotal = (
```

```
        SELECT COALESCE(SUM(subtotal), 0)
        FROM cotizacion_detalles
        WHERE cotizacion_id = NEW.cotizacion_id
    ),
    total = subtotal - descuento + impuestos,
    actualizado_en = CURRENT_TIMESTAMP
    WHERE id = NEW.cotizacion_id;
END//

-- Trigger para actualizar totales de venta cuando se modifican los detalles
CREATE TRIGGER tr_actualizar_total_venta
    AFTER INSERT ON venta_detalles
    FOR EACH ROW
BEGIN
    UPDATE ventas
    SET subtotal = (
        SELECT COALESCE(SUM(subtotal), 0)
        FROM venta_detalles
        WHERE venta_id = NEW.venta_id
    ),
    total = subtotal - descuento + impuestos,
    actualizado_en = CURRENT_TIMESTAMP
    WHERE id = NEW.venta_id;
END//
```

```
DELIMITER ;

-- ========================================================

-- INSERCIÓN DE DATOS INICIALES

-- ========================================================

-- Datos iniciales para departamentos

INSERT IGNORE INTO departamentos (nombre, descripcion) VALUES

('Gerencia', 'Dirección general de la empresa'),

('Ventas', 'Departamento de ventas y atención al cliente'),

('Contabilidad', 'Departamento de finanzas y contabilidad'),

('Operaciones', 'Departamento de operaciones y logística'),

('Marketing', 'Departamento de marketing y publicidad'),

('Sistemas', 'Departamento de tecnología e informática');

-- Datos iniciales para tipos de documento

INSERT IGNORE INTO tipos_documento (codigo, descripcion) VALUES

('DNI', 'Documento Nacional de Identidad'),

('CE', 'Cédula de Extranjería'),

('PAS', 'Pasaporte'),

('RUC', 'Registro Único de Contribuyentes');

-- Datos iniciales para métodos de pago

INSERT IGNORE INTO metodos_pago (codigo, descripcion, requiere_referencia)
VALUES

('EFE', 'Efectivo', FALSE),
```

('TJT', 'Tarjeta de Crédito/Débito', TRUE),

('TRA', 'Transferencia Bancaria', TRUE),

('DEP', 'Depósito Bancario', TRUE),

('CHE', 'Cheque', TRUE);

-- Datos iniciales para tipos de activo

INSERT IGNORE INTO tipos_activo (codigo, descripcion, vida_util_años) VALUES

('VEH', 'Vehículos', 10),

('MOB', 'Mobiliario', 8),

('EQU', 'Equipos', 5),

('TEC', 'Tecnología', 3);

-- Datos iniciales para canales de marketing

INSERT IGNORE INTO canales_marketing (codigo, nombre) VALUES

('SOC', 'Redes Sociales'),

('EMA', 'Email Marketing'),

('WEB', 'Sitio Web'),

('RAD', 'Radio'),

('TV', 'Televisión'),

('IMP', 'Medios Impresos');

-- Datos iniciales para tipos de reporte

INSERT IGNORE INTO tipos_reporte (codigo, nombre) VALUES

('VEN', 'Reporte de Ventas'),

('FIN', 'Reporte Financiero'),

```
('INV', 'Reporte de Inventario'),

('CLI', 'Reporte de Clientes'),

('CAM', 'Reporte de Campañas');


SET foreign_key_checks = 1;


-- ========================================================

-- FIN DE SCRIPT

-- ========================================================
```