Fecha: 17/12/2025

Título: “Gestión de Comercio – Marrero Ignacio”

Introducción

En este proyecto se propone el diseño de una base de datos relacional para la gestión de un comercio que compra productos a distintos proveedores y los vende a clientes minoristas y mayoristas.

La base de datos busca unificar en un mismo esquema la información de productos, categorías, clientes, tipos de clientes, proveedores, pedidos de venta, pedidos de compra, detalle de cada operación y stock de inventario. De esta forma se obtiene una estructura consistente, normalizada y preparada para responder a necesidades contables, logísticas y analíticas del negocio.

Esta entrega corresponde a la primera etapa del proyecto final e incluye: la descripción de la temática, el objetivo del proyecto, la situación problemática, el modelo de negocio y el modelo entidad–relación, el listado detallado de tablas y el script SQL de creación del esquema.

Objetivo del proyecto

El objetivo principal es diseñar e implementar un esquema de base de datos que permita:

Registrar de forma integrada las ventas a clientes y las compras a proveedores.

Mantener el inventario actualizado por producto.

Disponer de información confiable para consultas contables (facturación, totales por cliente/proveedor), logísticas (faltantes y reposiciones de stock) y analíticas (productos más vendidos, comportamiento por tipo de cliente).

Situación problemática

En muchas pymes comerciales o pequeñas fábricas la información del negocio se gestiona mediante planillas aisladas, documentos en papel o sistemas parciales que no se integran. Esto genera problemas como:

Duplicación o inconsistencia de datos de clientes, proveedores y productos.

Dificultad para reconstruir qué se vendió a quién y en qué fecha.

Falta de control de inventario, con riesgos de faltantes o sobre-stock.

Mucho esfuerzo manual para obtener indicadores básicos de gestión (ventas por producto, compras por proveedor, etc.).

La implementación de una base de datos relacional bien diseñada apunta a solucionar estas brechas, asegurando integridad referencial, evitando redundancias y facilitando la generación de reportes para la toma de decisiones.

Modelo de negocio

El modelo de negocio que se representa en esta base de datos es el de un comercio que:

Compra productos o materias primas a diversos proveedores.

Almacena estos productos en inventario, organizados en categorías.

Vende dichos productos a clientes de distinto tipo (minoristas y mayoristas).

El flujo principal es:

El área de compras genera pedidos de compra a proveedores, con el detalle de productos, cantidades y precios.

Al recibir la mercadería, se actualiza el stock en la tabla de inventario.

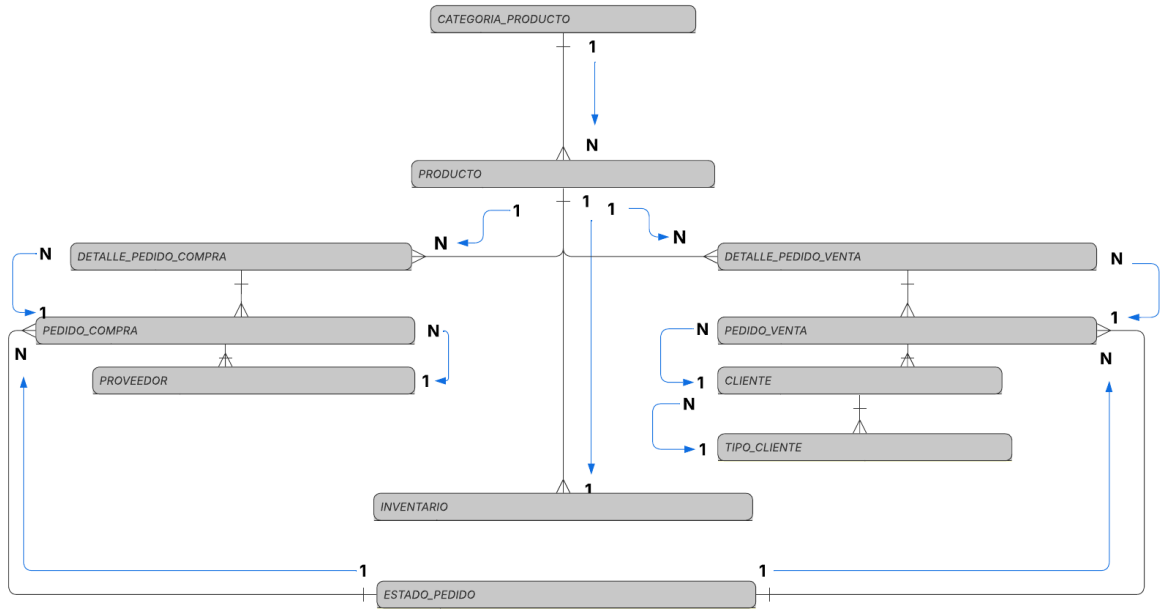
El área comercial genera pedidos de venta a clientes, registrando los productos vendidos, cantidades y precios.

Al confirmar la venta, se descuenta el stock correspondiente.

Los pedidos de compra y venta tienen asociado un estado (pendiente, facturado, cancelado), lo que facilita el seguimiento operativo.

De esta forma, la base de datos soporta la operación diaria y, al mismo tiempo, permite luego construir reportes y análisis sobre la misma.

Modelo Entidad–Relación (MER)

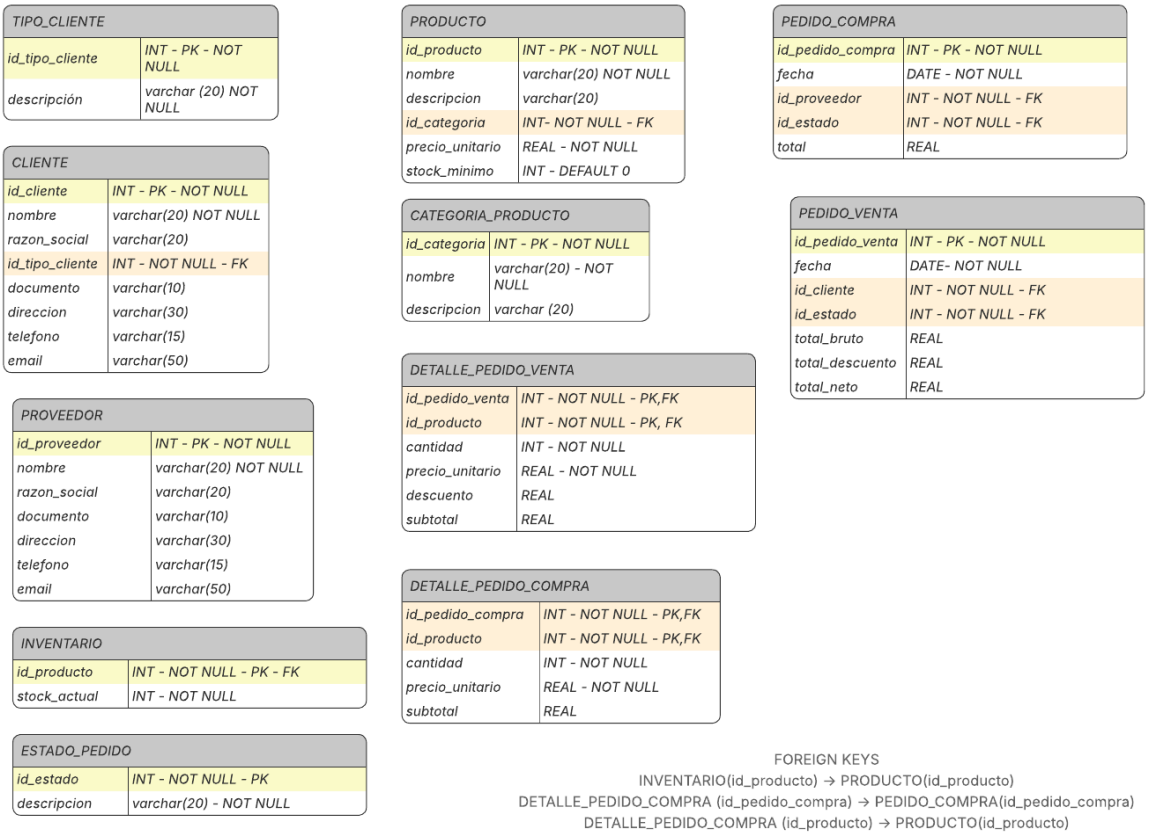


Entidades principales
- CATEGORIA_PRODUCTO: clasifica los productos en categorías.
- PRODUCTO: artículos que se compran a proveedores y se venden a clientes.
- CLIENTE: clientes minoristas o mayoristas del comercio.
- TIPO_CLIENTE: catálogo de tipos de cliente (por ejemplo, minorista/mayorista).
- PROVEEDOR: proveedores de productos o materias primas.
- PEDIDO_VENTA: cabecera de los pedidos de venta realizados a clientes.
- DETALLE_PEDIDO_VENTA: detalle de los productos incluidos en cada pedido de venta.
- PEDIDO_COMPRA: cabecera de los pedidos de compra realizados a proveedores.
- DETALLE_PEDIDO_COMPRA: detalle de los productos incluidos en cada pedido de compra.
- INVENTARIO: stock actual disponible para cada producto.
- ESTADO_PEDIDO: estados posibles de un pedido (pendiente, facturado, cancelado, etc.).

Relaciones entre entidades
CATEGORIA_PRODUCTO (1) — (N) PRODUCTO: una categoría puede tener muchos productos, cada producto pertenece a una sola categoría.
TIPO_CLIENTE (1) — (N) CLIENTE: cada tipo de cliente agrupa a muchos clientes, cada cliente tiene un único tipo.
CLIENTE (1) — (N) PEDIDO_VENTA: un cliente puede realizar muchos pedidos de venta, cada pedido de venta pertenece a un solo cliente.
PROVEEDOR (1) — (N) PEDIDO_COMPRA: un proveedor puede recibir muchos pedidos de compra, cada pedido de compra pertenece a un solo proveedor.
ESTADO_PEDIDO (1) — (N) PEDIDO_VENTA: un estado puede aplicarse a muchos pedidos de venta, cada pedido de venta tiene un único estado.
ESTADO_PEDIDO (1) — (N) PEDIDO_COMPRA: un estado puede aplicarse a muchos pedidos de compra, cada pedido de compra tiene un único estado.

PEDIDO_VENTA (1) — (N) DETALLE_PEDIDO_VENTA: un pedido de venta tiene muchas líneas de detalle, cada detalle pertenece a un solo pedido de venta.
PEDIDO_COMPRA (1) — (N) DETALLE_PEDIDO_COMPRA: un pedido de compra tiene muchas líneas de detalle, cada detalle pertenece a un solo pedido de compra.
PRODUCTO (1) — (N) DETALLE_PEDIDO_VENTA: un producto puede aparecer en muchos detalles de venta, cada detalle de venta refiere a un solo producto.
PRODUCTO (1) — (N) DETALLE_PEDIDO_COMPRA: un producto puede aparecer en muchos detalles de compra, cada detalle de compra refiere a un solo producto.
PRODUCTO (1) — (1) INVENTARIO: cada producto tiene asociado un único registro de inventario, y cada registro de inventario corresponde

Listado de tablas



FOREIGN KEYS

INVENTARIO(id_producto) → PRODUCTO(id_producto)
DETALLE_PEDIDO_COMPRA (id_pedido_compra) → PEDIDO_COMPRA(id_pedido_compra)
DETALLE_PEDIDO_COMPRA (id_producto) → PRODUCTO(id_producto)
PEDIDO_COMPRA(id_proveedor) →PROVEEDOR(id_proveedor)
DETALLE_PEDIDO_VENTA (id_pedido_venta) → PEDIDO_VENTA (id_pedido_venta)
DETALLE_PEDIDO_VENTA (id_producto) → PRODUCTO (id_producto)
PEDIDO_VENTA (id_cliente) → CLIENTE(id_cliente)
PRODUCTO (id_categoria) → CATEGORIA_PRODUCTO (id_categoria)
CLIENTE (id_tipo_cliente) → TIPO_CLIENTE (id_tipo_cliente)
PEDIDO_VENTA(id_estado) → ESTADO_PEDIDO(id_estado)
PEDIDO_COMPRA(id_estado) → ESTADO_PEDIDO(id_estado)

VISTAS

1. *vw_ventas_detalladas*

Objetivo: ver cada venta con cliente, tipo de cliente, estado, producto y subtotal por línea.
Tablas: PEDIDO_VENTA, DETALLE_PEDIDO_VENTA, CLIENTE, TIPO_CLIENTE, PRODUCTO, ESTADO_PEDIDO.

2. *vw_compras_detalladas*

Objetivo: ver cada compra con proveedor, estado, producto y subtotal por línea.
Tablas: PEDIDO_COMPRA, DETALLE_PEDIDO_COMPRA, PROVEEDOR, PRODUCTO, ESTADO_PEDIDO.

3. *vw_stock_bajo_minimo*

Objetivo: detectar productos con stock_actual < stock_minimo (reposición).
Tablas: INVENTARIO, PRODUCTO, CATEGORIA_PRODUCTO.

4. *vw_totales_por_cliente*

Objetivo: total neto vendido por cliente (reportes/contabilidad).
Tablas: PEDIDO_VENTA, CLIENTE.

5. *vw_totales_por_proveedor*

Objetivo: total comprado por proveedor (control de compras).
Tablas: PEDIDO_COMPRA, PROVEEDOR.

6. *vw_productos_mas_vendidos*

Objetivo: ranking por cantidad vendida e importe (analítica).
Tablas: DETALLE_PEDIDO_VENTA, PRODUCTO.

FUNCIONES

1. *fn_stock_actual(p_id_producto)*

Objetivo: devolver el stock actual de un producto.
Tablas: INVENTARIO.

2. *fn_total_neto_venta(p_id_pedido_venta)*

Objetivo: calcular el total neto (sumatoria subtotales − descuentos) del pedido.
Tablas: DETALLE_PEDIDO_VENTA.

3. *fn_total_compra(p_id_pedido_compra)*

Objetivo: calcular el total de una compra desde su detalle.
Tablas: DETALLE_PEDIDO_COMPRA.

STORED PROCEDURES

1. *sp_recalcular_totales_venta(p_id_pedido_venta)*

Objetivo: recalcular los totales de un pedido de venta a partir de su detalle, actualizando los campos total_bruto, total_descuento y total_neto.
Este procedimiento permite mantener la consistencia de los importes de venta cuando se agregan o modifican líneas de detalle.
Tablas involucradas: PEDIDO_VENTA, DETALLE_PEDIDO_VENTA.

2. *sp_recalcular_total_compra(p_id_pedido_compra)*

Objetivo: recalcular el total de un pedido de compra en función del detalle de productos asociados.
Se utiliza para asegurar que el importe total del pedido refleje correctamente la suma de los subtotales de cada línea de compra.
Tablas involucradas: PEDIDO_COMPRA, DETALLE_PEDIDO_COMPRA.

3. *sp_cambiar_estado_pedido_venta(p_id_pedido_venta, p_id_estado)*

Objetivo: modificar el estado de un pedido de venta (pendiente, facturado o cancelado) de forma controlada.
Este procedimiento facilita la gestión del ciclo de vida de los pedidos de venta sin necesidad de actualizar manualmente la tabla.
Tablas involucradas: PEDIDO_VENTA, ESTADO_PEDIDO.

4. *sp_cambiar_estado_pedido_compra(p_id_pedido_compra, p_id_estado)*

Objetivo: modificar el estado de un pedido de compra, permitiendo su seguimiento operativo desde la generación hasta la facturación o cancelación.
Aporta control y trazabilidad al proceso de compras del comercio.
Tablas involucradas: PEDIDO_COMPRA, ESTADO_PEDIDO.

TRIGGERS

1. *trg_compra_suma_stock*

Cuándo: AFTER INSERT en DETALLE_PEDIDO_COMPRA
Qué hace: suma cantidad al INVENTARIO.stock_actual.

2. *trg_venta_resta_stock*

Cuándo: AFTER INSERT en DETALLE_PEDIDO_VENTA
Qué hace: resta cantidad al INVENTARIO.stock_actual.

3. *trg_venta_valida_stock*

Cuándo: BEFORE INSERT en DETALLE_PEDIDO_VENTA
Qué hace: impide vender si no hay stock suficiente.