Proyecto Final – Base de Datos Tienda de Guitarras

Autor: Diego Sánchez
Curso: SQL – Septiembre 2025

Índice

---

1. Introducción

Este proyecto consiste en el diseño e implementación de una base de datos relacional para una tienda de guitarras. La solución permite administrar clientes, proveedores, empleados, productos, ventas y compras, así como generar reportes analíticos a partir de la información registrada.
El proyecto busca centralizar la información, optimizar la gestión de inventarios y facilitar el análisis de ventas para la toma de decisiones estratégicas.

---

2. Objetivos

Objetivo General

Crear una base de datos relacional que soporte la operación de la tienda de guitarras, incluyendo clientes, empleados, proveedores, productos, ventas y compras, y permita generar informes confiables para la gestión.

Objetivos Específicos

- Centralizar la gestión de clientes, ventas y proveedores.
- Permitir análisis de ventas y control de stock.
- Automatizar el ajuste de inventario mediante triggers.
- Facilitar la toma de decisiones mediante consultas e informes analíticos.
- Generar un modelo de datos profesional con integridad referencial y relaciones claras.

---

3. Situación Problemática

La tienda de guitarras llevaba su control en planillas y registros dispersos, lo que generaba:

- Falta de trazabilidad en ventas y stock.
- Duplicación de información de clientes y proveedores.
- Dificultad para generar informes de ventas y compras confiables.

La implementación de la base de datos busca solucionar estas problemáticas mediante la unificación de la información y la automatización de procesos de gestión.

---

4. Modelo de Negocio

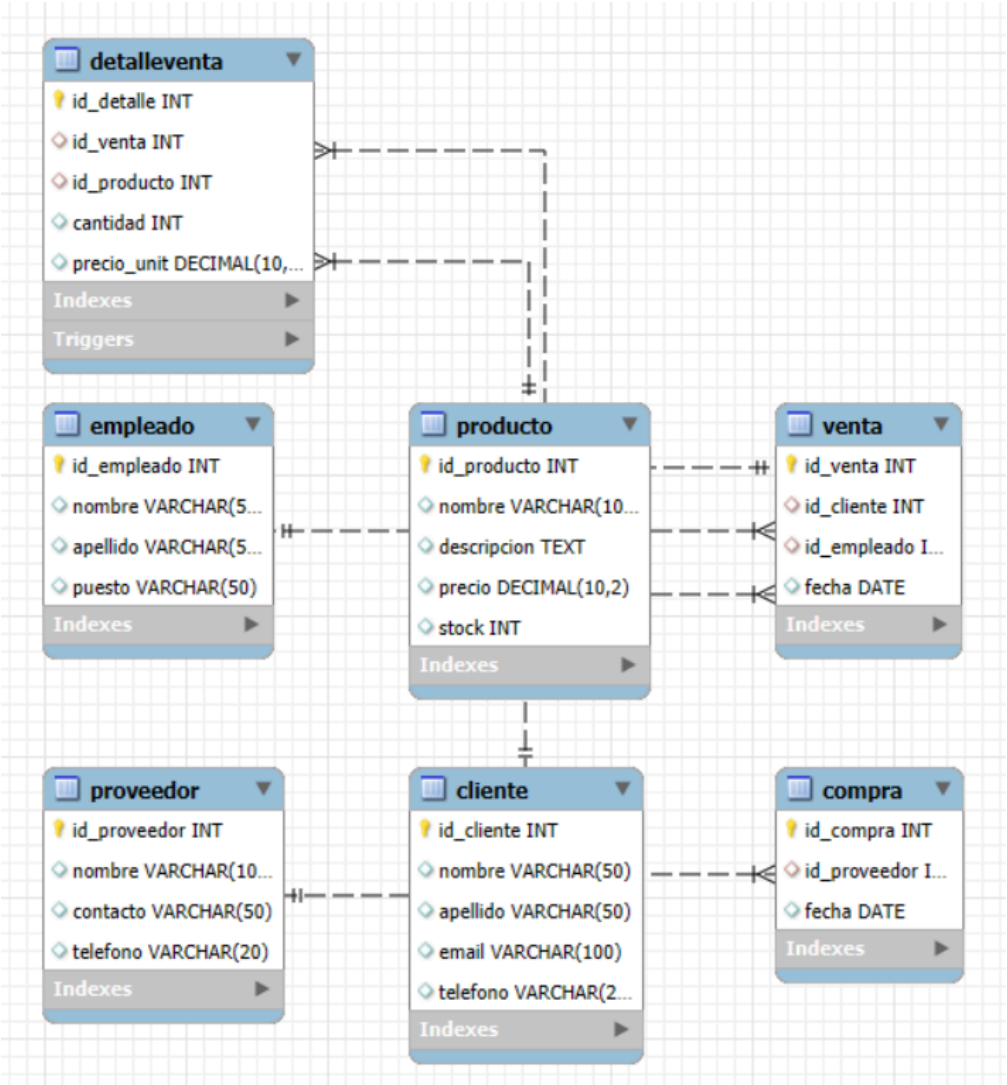
La tienda comercializa instrumentos musicales y accesorios relacionados.

- **Clientes**: realizan compras de productos atendidos por empleados.
- **Empleados**: registran ventas y mantienen contacto con los clientes.
- **Productos**: instrumentos, accesorios y consumibles en stock.
- **Proveedores**: abastecen a la tienda para mantener disponibilidad de productos.
- **Ventas y Compras**: registran las transacciones, asociadas a clientes, empleados y proveedores.

Cada venta se desglosa en detalle de productos, y cada compra mantiene el stock actualizado.

---

5. Diagrama Entidad–Relación



Entidades principales: Cliente, Empleado, Proveedor, Producto, Venta, DetalleVenta, Compra.
Relaciones principales:

- Cliente 1:N Venta
- Empleado 1:N Venta
- Venta 1:N DetalleVenta
- Producto 1:N DetalleVenta
- Proveedor 1:N Compra

---

6. Diseño de Tablas

Tabla Cliente

- `id_cliente` (INT, PK, AUTO_INCREMENT) → Identificador único.
- `nombre` (VARCHAR(50), NOT NULL) → Nombre del cliente.
- `apellido` (VARCHAR(50), NOT NULL) → Apellido del cliente.
- `email` (VARCHAR(100), UNIQUE) → Correo electrónico.
- `telefono` (VARCHAR(20), NOT NULL) → Teléfono de contacto.

Tabla Proveedor

- `id_proveedor` (INT, PK, AUTO_INCREMENT) → Identificador único.
- `nombre` (VARCHAR(100), NOT NULL) → Nombre de la empresa proveedora.
- `contacto` (VARCHAR(50), NOT NULL) → Persona de contacto.
- `telefono` (VARCHAR(20), NOT NULL) → Teléfono de contacto.

Tabla Empleado

- `id_empleado` (INT, PK, AUTO_INCREMENT) → Identificador único.
- `nombre` (VARCHAR(50), NOT NULL) → Nombre del empleado.
- `apellido` (VARCHAR(50), NOT NULL) → Apellido del empleado.
- `puesto` (VARCHAR(50), NOT NULL) → Cargo dentro de la tienda.

Tabla Producto

- `id_producto` (INT, PK, AUTO_INCREMENT) → Identificador único del producto.
- `nombre` (VARCHAR(100), NOT NULL) → Nombre del producto.
- `descripcion` (TEXT, NOT NULL) → Descripción del producto.
- `precio` (DECIMAL(10,2), NOT NULL) → Precio unitario.
- `stock` (INT, NOT NULL) → Cantidad disponible.

Tabla Venta

- `id_venta` (INT, PK, AUTO_INCREMENT) → Identificador de la venta.
- `id_cliente` (INT, FK → Cliente.id_cliente) → Cliente asociado a la venta.
- `id_empleado` (INT, FK → Empleado.id_empleado) → Empleado que registra la venta.
- `fecha` (DATE) → Fecha de la venta.

Tabla DetalleVenta

- `id_detalle` (INT, PK, AUTO_INCREMENT) → Identificador único del detalle.
- `id_venta` (INT, FK → Venta.id_venta) → Venta asociada.
- `id_producto` (INT, FK → Producto.id_producto) → Producto vendido.
- `cantidad` (INT) → Cantidad vendida.

- `precio_unit` (DECIMAL(10,2)) → Precio unitario al momento de la venta.

Tabla Compra

- `id_compra` (INT, PK, AUTO_INCREMENT) → Identificador de la compra.
- `id_proveedor` (INT, FK → Proveedor.id_proveedor) → Proveedor asociado.
- `fecha` (DATE) → Fecha de la compra.

---

7. Vistas, Funciones, Procedimientos y Triggers

Vistas

1. **VistaClientesVentas** → Relaciona clientes con sus ventas.
2. **VistaStockProductos** → Lista productos con stock y precio.
3. **VistaDetalleVentas** → Muestra detalle de cada venta con cliente y productos.
4. **VistaVentasPorEmpleado** → Total de ventas registradas por cada empleado.
5. **VistaComprasPorProveedor** → Total de compras realizadas por cada proveedor.

Funciones

- **fn_TotalVenta** → Calcula el total de una venta sumando cantidad × precio unitario.
- **fn_StockDisponible** → Devuelve la cantidad disponible de un producto.

Procedimientos Almacenados

- **sp_RegistrarVenta** → Inserta una nueva venta asociada a cliente y empleado.
- **sp_ActualizarStock** → Actualiza el stock de un producto después de la venta.

Triggers

- **trg_AfterVentaDetalle** → Disminuye automáticamente el stock de un producto después de registrar un detalle de venta.

Todas las vistas, funciones, procedimientos y triggers permiten automatizar y simplificar consultas e informes sin repetir lógica en cada consulta.

---

8. Ejemplos de Consultas

- Obtener el total de venta de un ID:
  `SELECT fn_TotalVenta(1);`

- Consultar stock disponible de un producto:
  `SELECT fn_StockDisponible(1);`

- Listar clientes y sus ventas:
  `SELECT * FROM VistaClientesVentas;`

- Consultar ventas por empleado:
  `SELECT * FROM VistaVentasPorEmpleado;`

---

9. Informes Generados

- **Ventas por cliente y producto** → vista `VistaDetalleVentas`.

- **Productos más vendidos** → consulta agrupada sobre `DetalleVenta`.

- **Stock de productos** → función `fn_StockDisponible`.

- **Total de ventas** → función `fn_TotalVenta`.

- **Compras por proveedor** → vista `VistaComprasPorProveedor`.

Estos informes pueden exportarse a Excel o Power BI para análisis gráfico y reportes ejecutivos.

---

10. Herramientas Utilizadas

- **MySQL Workbench** → Creación y gestión de la base de datos. Armado de Diagrama E-R.
- **GitHub :** https://github.com/DiegoSanchez78.
- **Visual Studio Code** → Para subir proyecto a Github.
- **Google Drive Word** → Armado de este informe.

---

11. Conclusiones

- Se centralizó la información de clientes, empleados, proveedores, productos, ventas y compras.
- Se mejoró el control de stock mediante triggers y funciones.
- Se automatizó la generación de informes mediante vistas y procedimientos.
- La base de datos permite un análisis confiable de la operación, facilitando la toma de decisiones.
- La estructura implementada es escalable y puede integrarse con aplicaciones web o sistemas adicionales.