Maria Victoria D'Ercole

BASE DE DATOS: LIBRERÍA

Introducción

El presente proyecto consiste en el desarrollo de una base de datos para una librería. La finalidad es organizar y gestionar de manera eficiente la información relacionada con los libros, autores, clientes y ventas. Este sistema permitirá almacenar datos estructurados y generar consultas rápidas que faciliten la gestión y administración de la librería, mejorando la toma de decisiones.

Objetivo

El objetivo principal de este proyecto es diseñar una base de datos relacional que permita gestionar la información clave de una librería. Esto incluye la administración de autores y sus obras, el registro de clientes y el control de ventas. Además, se busca proporcionar información contable para el seguimiento de ingresos, una mejor logística para el control de stock y datos analíticos que faciliten la toma de decisiones comerciales.

Situación Problemática

Las librerías manejan grandes volúmenes de información sobre libros, clientes y transacciones. Sin un sistema estructurado, el almacenamiento y recuperación de datos pueden volverse ineficientes, lo que dificulta la gestión de inventario y las ventas. La implementación de esta base de datos permite solucionar problemas como:

- Falta de organización en la gestión de autores y libros.
- Dificultades en la administración de clientes y ventas.
- Falta de registros detallados sobre el stock disponible.
- Ausencia de informes de ventas y análisis de rendimiento.

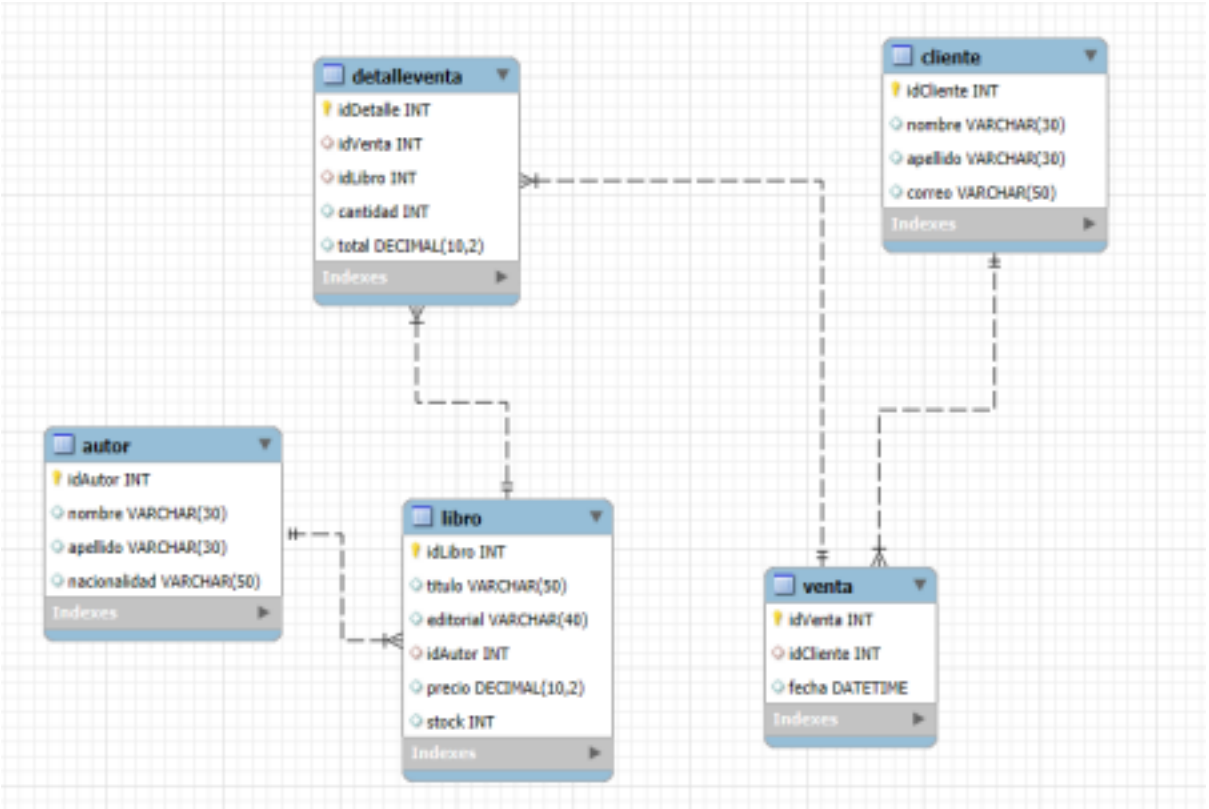
Modelo de Negocio

El modelo de negocio de la librería está basado en la venta de libros físicos a clientes particulares. Se busca ofrecer una amplia variedad de títulos de diferentes autores y nacionalidades. La base de datos propuesta permitirá a la librería mejorar la gestión de su inventario, mantener un historial de compras de sus clientes y analizar el rendimiento de ventas.

Diagrama Entidad-Relación

El diagrama entidad-relación de la base de datos refleja la estructura establecida en el esquema SQL. Se incluyen las siguientes entidades y sus relaciones:

- **Autor** (1 a muchos con Libro)
- **Libro** (1 a muchos con DetalleVenta)
- **Cliente** (1 a muchos con Venta)
- **Venta** (1 a muchos con DetalleVenta)



Listado de Tablas

A continuación, se describen las tablas de la base de datos junto con sus campos, tipos de datos y su propósito en el sistema:

1. **Autor**
    - idAutor (INT, PRIMARY KEY, AUTO_INCREMENT): Identificador único de cada autor.
    - nombre (VARCHAR(30)): Nombre del autor.
    - apellido (VARCHAR(30)): Apellido del autor.
    - nacionalidad (VARCHAR(50)): Nacionalidad del autor.

   Esta tabla almacena la información básica de los autores de los libros.

2. **Libro**
    - idLibro (INT, PRIMARY KEY, AUTO_INCREMENT): Identificador único del libro.
    - titulo (VARCHAR(100)): Título del libro.
    - idAutor (INT, FOREIGN KEY): Identificador del autor del libro.
    - precio (DECIMAL(10,2)): Precio del libro.
    - stock (INT): Cantidad de unidades disponibles.
    - editorial (VARCHAR(100)): Nombre de la editorial que publica el libro.

   En esta tabla se registran los libros disponibles en la librería, incluyendo su autor y editorial.

3. **Cliente**
    - idCliente (INT, PRIMARY KEY, AUTO_INCREMENT): Identificador único de cada cliente.
    - nombre (VARCHAR(30)): Nombre del cliente.
    - apellido (VARCHAR(30)): Apellido del cliente.
    - correo (VARCHAR(50)): Correo electrónico del cliente.

Esta tabla almacena información sobre los clientes de la librería.

4. **Venta**
    - idVenta (INT, PRIMARY KEY, AUTO_INCREMENT): Identificador único de cada venta.
    - idCliente (INT, FOREIGN KEY): Cliente que realizó la compra.
    - fecha (DATETIME): Fecha y hora de la venta.

Esta tabla registra cada venta realizada en la librería.

5. **DetalleVenta**
    - idDetalle (INT, PRIMARY KEY, AUTO_INCREMENT): Identificador único del detalle de venta.
    - idVenta (INT, FOREIGN KEY): Venta a la que pertenece el detalle.
    - idLibro (INT, FOREIGN KEY): Libro vendido en la transacción.
    - cantidad (INT): Cantidad de ejemplares vendidos.
    - total (DECIMAL(10,2)): Precio total de la compra.

Esta tabla desglosa cada venta, especificando qué libros fueron comprados y en qué cantidad.

Listado de Vistas

- **VistaLibrosDisponibles**

Descripción:
Esta vista muestra los libros disponibles en la librería, junto con su información relevante.

Objetivo:
Facilitar la consulta rápida de los libros en stock, incluyendo el nombre del autor y el precio. **Tablas involucradas:**

`Libro` → Proporciona el título, editorial, precio y stock.

`Autor` → Permite obtener el nombre y apellido del autor del libro.

Consulta SQL:

```sql
CREATE VIEW VistaLibrosDisponibles AS
SELECT
    l.idLibro, l.titulo, l.editorial,
    CONCAT(a.nombre, ' ', a.apellido) AS autor,
    l.precio, l.stock
FROM Libro l
LEFT JOIN Autor a ON l.idAutor = a.idAutor
WHERE l.stock > 0;
```

- **VistaVentasClientes**

Descripción:
Esta vista muestra el historial de ventas realizadas, con información sobre el cliente que realizó la compra y la fecha de la venta.

Objetivo:
Facilitar el acceso a los registros de ventas, permitiendo consultar qué clientes han realizado compras y en qué fecha.

Tablas involucradas:

`Venta` → Proporciona el ID de la venta y la fecha en que se realizó. `Cliente` →

Permite obtener el nombre y apellido del cliente que realizó la compra. **Consulta SQL:**

```sql
CREATE VIEW VistaVentasClientes AS
SELECT
    v.idVenta,
    CONCAT(c.nombre, ' ', c.apellido) AS cliente,
    v.fecha
FROM Venta v
JOIN Cliente c ON v.idCliente = c.idCliente;
```

Listado de Funciones

- **CalcularTotalVenta(id_venta INT)**

Descripción:
Esta función calcula el monto total de una venta específica sumando los valores totales de los libros comprados en el detalle de la venta.

Objetivo:
Facilitar la consulta del monto total de una venta sin necesidad de hacer cálculos

manualmente en las consultas SQL.

Tablas involucradas:

DetalleVenta → Se usa para obtener los valores individuales de cada producto en la venta y sumarlos.

Código SQL:

```
DELIMITER $$
CREATE FUNCTION CalcularTotalVenta(id_venta INT) RETURNS DECIMAL(10,2)
DETERMINISTIC
BEGIN
    DECLARE total DECIMAL(10,2);
    SELECT SUM(total) INTO total FROM DetalleVenta WHERE idVenta = id_venta;
    RETURN IFNULL(total, 0);
END$$
DELIMITER ;
```

- **StockDisponible(id_libro INT)**

Descripción:
Esta función obtiene la cantidad de unidades disponibles en stock de un libro específico.

Objetivo:
Permitir la consulta rápida del stock de un libro sin necesidad de realizar una consulta manual en la tabla Libro.

Tablas involucradas:

Libro → Se usa para extraer el stock disponible del libro

solicitado. Código SQL:

```
DELIMITER $$
CREATE FUNCTION StockDisponible(id_libro INT) RETURNS INT
DETERMINISTIC
BEGIN
    DECLARE stock_actual INT;
    SELECT stock INTO stock_actual FROM Libro WHERE idLibro = id_libro;
    RETURN IFNULL(stock_actual, 0);
END$$
DELIMITER ;
```

Listado de Stored Procedures

- **RegistrarVenta(id_cliente INT, OUT nuevo_id INT)**

Descripción:
Este procedimiento almacenado registra una nueva venta en la base de datos y devuelve el ID de la venta recién creada.

Objetivo:
Automatizar la creación de ventas sin que el usuario tenga que insertar manualmente los datos y obtener de inmediato el ID de la nueva venta para agregar los detalles correspondientes.

Tablas involucradas:

- Venta → Se inserta un nuevo registro en esta tabla con la fecha actual y el ID del cliente que realiza la compra.

Código SQL:

```
DELIMITER $$
CREATE PROCEDURE RegistrarVenta(IN id_cliente INT, OUT nuevo_id INT)
BEGIN
    INSERT INTO Venta (idCliente) VALUES (id_cliente);
    SET nuevo_id = LAST_INSERT_ID();
END$$
DELIMITER ;
```

• AgregarDetalleVenta(id_venta INT, id_libro INT, cantidad INT)

Descripción:
Este procedimiento almacenado añade un detalle de venta a una venta existente, asegurando que haya suficiente stock del libro antes de realizar la operación. Si el stock es insuficiente, se genera un error.

Objetivo:
Facilitar la gestión de ventas asegurando que los libros vendidos tengan stock suficiente y actualizando automáticamente la cantidad en inventario.

Tablas involucradas:

DetalleVenta → Se inserta un nuevo registro con el libro, cantidad y total de la compra.

Libro → Se consulta el precio y el stock del libro antes de la venta y se actualiza el stock tras la transacción.

Código SQL:

```
DELIMITER $$
CREATE PROCEDURE AgregarDetalleVenta(
    IN id_venta INT,
    IN id_libro INT,
    IN cantidad INT
)
BEGIN
    DECLARE precio_libro DECIMAL(10,2);
    DECLARE stock_actual INT;

    -- Obtener el stock y el precio del libro
    SELECT precio, stock INTO precio_libro, stock_actual FROM Libro WHERE idLibro = id_libro;

    -- Verificar que haya stock suficiente
    IF stock_actual >= cantidad THEN
        INSERT INTO DetalleVenta (idVenta, idLibro, cantidad, total)
        VALUES (id_venta, id_libro, cantidad, precio_libro * cantidad);

        -- Actualizar stock
        UPDATE Libro SET stock = stock - cantidad WHERE idLibro = id_libro;
    ELSE
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Stock insuficiente';
    END IF;
END$$
DELIMITER ;
```

Listado de Triggers

- **Trigger para calcular automáticamente el total en DetalleVenta**

Descripción:

Este trigger calcula automáticamente el total de una venta en `DetalleVenta` antes de insertar un nuevo registro. Obtiene el precio del libro desde la tabla `Libro` y lo multiplica por la cantidad comprada.

Objetivo:

Automatizar el cálculo del total en `DetalleVenta`, evitando errores manuales y garantizando que el valor registrado sea correcto.

Tablas involucradas:

- `DetalleVenta` → Se inserta un nuevo registro con el libro, cantidad y total calculado.
- `Libro` → Se consulta el precio del libro antes de la inserción en `DetalleVenta`.

```sql
-- Trigger para calcular automáticamente el total en DetalleVenta
DELIMITER $$
CREATE TRIGGER before_insert_detalleventa
BEFORE INSERT ON DetalleVenta
FOR EACH ROW
BEGIN
    DECLARE precio_libro DECIMAL(10,2);
    -- Obtener el precio del libro
    SELECT precio INTO precio_libro FROM Libro WHERE idLibro = NEW.idLibro;
    -- Calcular el total
    SET NEW.total = precio_libro * NEW.cantidad;
END$$
DELIMITER ;
```

- **Trigger para evitar ventas si el stock es insuficiente**

Descripción:

Este procedimiento almacenado permite añadir un detalle de venta a una venta existente, garantizando que haya suficiente stock disponible antes de completar la transacción. Si el stock es insuficiente, se genera un error y la operación no se ejecuta. Además, el procedimiento actualiza automáticamente la cantidad en inventario tras la venta.

Objetivo:

Facilitar la gestión de ventas asegurando que los libros vendidos tengan stock suficiente antes de registrar la compra, evitando inconsistencias en el inventario y proporcionando un flujo de trabajo eficiente.

Tablas involucradas:

- DetalleVenta → Se inserta un nuevo registro con la venta, el libro vendido, la cantidad y el total de la compra.

- Libro → Se consulta el stock y el precio del libro antes de la venta y, tras una venta exitosa, se actualiza la cantidad disponible en el inventario.

```sql
-- Trigger para evitar ventas si el stock es insuficiente
DELIMITER $$
CREATE TRIGGER before_insert_detalleventa_check_stock
BEFORE INSERT ON DetalleVenta
FOR EACH ROW
BEGIN
    DECLARE stock_actual INT;

    -- Obtener el stock actual del libro
    SELECT stock INTO stock_actual FROM Libro WHERE idLibro = NEW.idLibro;

    -- Verificar si hay suficiente stock
    IF stock_actual < NEW.cantidad THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Stock insuficiente para completar la venta';
    END IF;
END$$
DELIMITER ;
```

- **Trigger para actualizar el stock al eliminar un detalle de venta**

Descripción:

Este procedimiento almacenado permite eliminar un detalle de venta y, al hacerlo, reponer automáticamente el stock del libro afectado. De esta manera, se garantiza que los productos eliminados de una venta vuelvan a estar disponibles en el inventario.

Objetivo:

Mantener la coherencia en el inventario asegurando que, si una venta es cancelada o un detalle de venta es eliminado, el stock de los libros involucrados se restaure correctamente.

Tablas involucradas:

- DetalleVenta → Se elimina un registro correspondiente a una venta específica, reduciendo la cantidad de libros vendidos.

- Libro → Se actualiza el stock del libro, sumando la cantidad previamente vendida cuando se elimina el detalle de la venta.

```sql
-- Trigger para actualizar el stock al eliminar un detalle de venta
DELIMITER $$
CREATE TRIGGER after_delete_detalleventa_update_stock
AFTER DELETE ON DetalleVenta
FOR EACH ROW
BEGIN
    -- Reponer el stock del libro cuando se elimina un detalle de venta
    UPDATE Libro
    SET stock = stock + OLD.cantidad
    WHERE idLibro = OLD.idLibro;
END$$
DELIMITER ;
```