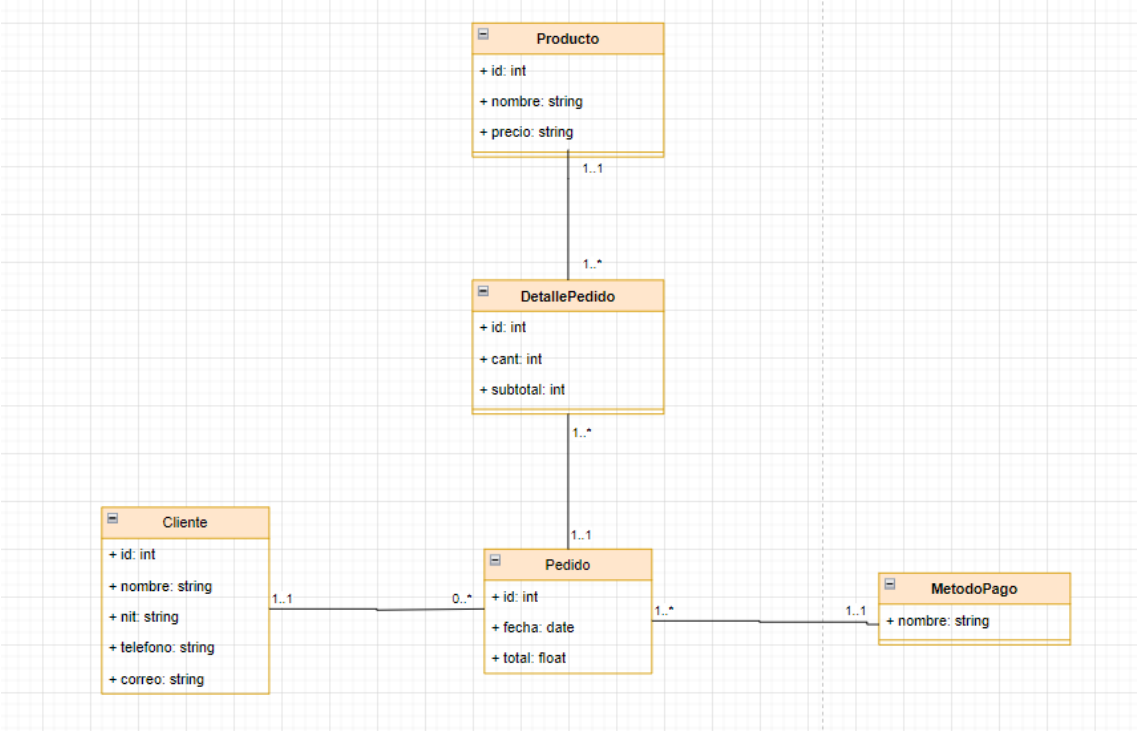
TEST 3

Diseño de base de datos relacional

Diseño conceptual:



Diseño Logico:

CLIENTES				
ID	NOMBRE	CORREO	TELEFONO	NIT
PK				

METODOPAGOS			
ID	NOMBRE		
PK			

PEDIDOS				
ID	FECHA	TOTAL	CLIENTE_ID	METODO_PAGO_ID
PK			FK	FK

PRODUCTOS			
ID	NOMBRE	PRECIO	COLOR
PK			

DETALLE_PEDIDOS				
ID	CANT	SUBTOTAL	PEDIDO_ID	PRODUCTO_ID
PK			FK	FK

Diseño Físico

-- Script sql

```
-- Crear la base de datos

CREATE DATABASE VentasDB;

GO


-- Usar la base de datos

USE VentasDB;

GO

-- Crear tabla clientes

CREATE TABLE clientes (

   id INT PRIMARY KEY IDENTITY(1,1),

   nombre NVARCHAR(100) NOT NULL,

   correo NVARCHAR(100),

   telefono NVARCHAR(50),

            nit NVARCHAR(50)

);

GO


-- Insertar registros en la tabla clientes

/*

INSERT INTO clientes (nombre, correo, telefono, nit)

VALUES

('Juan Pérez', 'juan.perez@gmail.com', '555-1234', '123456789'),

('María López', 'maria.lopez@gmail.com', '555-5678', '987654321'),

('Carlos Gómez', 'carlos.gomez@gmail.com', '555-8765', '112233445'),

('Ana Morales', 'ana.morales@gmail.com', '555-4321', '223344556'),

('Luis Rodríguez', 'luis.rodriguez@gmail.com', '555-2345', '334455667'),

('Sofía Ramírez', 'sofia.ramirez@gmail.com', '555-6543', '445566778'),

('Diego Fernández', 'diego.fernandez@gmail.com', '555-9876', '556677889'),

('Laura García', 'laura.garcia@gmail.com', '555-7654', '667788990'),

('Pedro Torres', 'pedro.torres@gmail.com', '555-3456', '778899001'),

('Julia Vega', 'julia.vega@gmail.com', '555-5432', '889900112');

*/


-- Crear tabla MetodoPago
```

```
CREATE TABLE metodo_pagos (
   id INT PRIMARY KEY IDENTITY(1,1),
   nombre NVARCHAR(100) NOT NULL
);
GO

-- Insertar registros en la tabla metodo_pagos
/*
INSERT INTO metodo_pagos (nombre)
VALUES
('Tarjeta de crédito'),
('Tarjeta de débito'),
('Efectivo'),
('Transferencia bancaria'),
('Paypal'),
('Google Pay'),
('Apple Pay'),
('Cheque'),
('Criptomoneda'),
('Pago móvil');
*/

-- Crear tabla pedidos
CREATE TABLE pedidos(
		id INT PRIMARY KEY IDENTITY(1,1),
		fecha NVARCHAR(10) NOT NULL,
		total DECIMAL(10,2) NOT NULL,
		cliente_id INT NOT NULL,
		metodo_pago_id INT,
		FOREIGN KEY (cliente_id) REFERENCES clientes(id)
		ON UPDATE CASCADE ON DELETE CASCADE,
   FOREIGN KEY (metodo_pago_id) REFERENCES metodo_pagos(id)
		ON UPDATE CASCADE ON DELETE CASCADE
);
GO
-- Insertar registros en la tabla pedidos
/*
INSERT INTO pedidos (fecha, total, cliente_id, metodo_pago_id)
```

```
VALUES
('2024-01-01', 100.50, 1, 1),
('2024-01-02', 150.75, 2, 2),
('2024-01-03', 200.00, 3, 3),
('2024-01-04', 250.25, 4, 4),
('2024-01-05', 300.50, 5, 5),
('2024-01-06', 350.75, 6, 6),
('2024-01-07', 400.00, 7, 7),
('2024-01-08', 450.25, 8, 8),
('2024-01-09', 500.50, 9, 9),
('2024-01-10', 550.75, 10, 10);
*/

-- Crear tabla Producto
CREATE TABLE productos (
   id INT PRIMARY KEY IDENTITY(1,1),
   nombre NVARCHAR(100) NOT NULL,
   precio DECIMAL(10,2) NOT NULL,
            color NVARCHAR(40) NOT NULL
);
GO
/*
-- Insertar registros en la tabla productos
INSERT INTO productos (nombre, precio, color)
VALUES
('Camiseta', 15.99, 'Rojo'),
('Pantalón', 29.99, 'Azul'),
('Zapatos', 49.99, 'Negro'),
('Chaqueta', 79.99, 'Gris'),
('Sombrero', 19.99, 'Marrón'),
('Bufanda', 12.99, 'Verde'),
('Guantes', 9.99, 'Negro'),
('Gafas de sol', 25.99, 'Negro'),
('Reloj', 150.99, 'Plata'),
('Bolso', 55.99, 'Negro');
*/

-- Crear tabla detalle_pedidos
```

```
CREATE TABLE detalle_pedidos (
   id INT PRIMARY KEY IDENTITY(1,1),
   cant INT NOT NULL,
   subtotal DECIMAL(10,2) NOT NULL,
            pedido_id INT NOT NULL,
            producto_id INT NOT NULL,
            FOREIGN KEY (pedido_id) REFERENCES pedidos(id)
            ON UPDATE CASCADE ON DELETE CASCADE,
            FOREIGN KEY (producto_id) REFERENCES productos(id)
            ON UPDATE CASCADE ON DELETE CASCADE,
);
GO
/*
-- Insertar registros en la tabla detalle_pedidos
INSERT INTO detalle_pedidos (cant, subtotal, pedido_id, producto_id)
VALUES
(1, 15.99, 1, 1),
(2, 29.99, 2, 2),
(1, 49.99, 3, 3),
(1, 79.99, 4, 4),
(1, 19.99, 5, 5),
(2, 12.99, 6, 6),
(1, 9.99, 7, 7),
(1, 25.99, 8, 8),
(1, 150.99, 9, 9),
(1, 55.99, 10, 10);
*/
```

Optimización de consultas

```
-- Parte 2: Optimización de consultas
-- Listar los clientes que han realizado el mayor número de pedidos en los últimos 6 meses.

SELECT c.id, c.nombre, COUNT(p.id) AS cantPedidos
FROM clientes c
JOIN Pedidos p ON c.id = p.cliente_id
WHERE CAST(p.fecha AS DATE) >= DATEADD(MONTH, -6, GETDATE())
GROUP BY c.id, c.nombre
ORDER BY cantPedidos DESC;
```

```
-- Obtener el producto más vendido en el último mes y la cantidad vendida.

SELECT p.id, p.nombre, SUM(dp.cant) AS cant

FROM Productos p

JOIN detalle_pedidos dp ON p.id = dp.id

JOIN pedidos ped ON dp.id = ped.id

WHERE ped.fecha >= DATEADD(MONTH, -1, GETDATE())

GROUP BY p.id, p.Nombre

ORDER BY cant DESC


-- Mostrar los clientes que no han realizado ningún pedido en el último año.

SELECT c.id, c.nombre

FROM clientes c

LEFT JOIN pedidos p ON c.id = p.id AND p.fecha >= DATEADD(YEAR, -1, GETDATE())

WHERE p.id IS NULL;


-- Listar los pedidos cuyo monto total supera los 5000 Bs.

SELECT id, Fecha, Total, cliente_id

FROM Pedidos

WHERE Total > 5000;
```

Análisis de rendimiento

Para optimizar el rendimiento de las consultas en la base de datos, es recomendable establecer índices en las columnas que se utilizan con frecuencia en las cláusulas **WHERE**, **JOIN** y **ORDER BY**. En particular, un índice en la columna **fecha** de la tabla **pedidos** es fundamental, ya que muchas consultas requieren filtrar pedidos según rangos de fechas, lo que permitirá una recuperación más eficiente de los registros. También sería beneficioso crear un índice en la columna **cliente_id** de la tabla **pedidos**, lo que optimizará las búsquedas relacionadas con los clientes y mejorará el rendimiento de las uniones con la tabla **clientes**. Además, un índice en la columna **producto_id** de la tabla **detalle_pedidos** sería útil para facilitar las consultas que buscan identificar los productos más vendidos o aquellos que forman parte de pedidos específicos. Es importante tener en cuenta que, aunque los índices son ventajosos para las consultas de lectura, pueden impactar negativamente el rendimiento de las operaciones de escritura (como **INSERT**, **UPDATE** y **DELETE**), ya que cada modificación en los datos requiere que se actualicen los índices. Por lo tanto, es esencial llevar a cabo un análisis detallado de las consultas más comunes y el patrón de acceso a los datos para definir una estrategia de indexación eficaz, logrando así un balance entre la velocidad de lectura y la efectividad en las operaciones de escritura.