Projeto de Banco de dados lógico E-commerce
Autor: Otoniele Santos
Data: 03.04.25

```
-- Criação do banco de dados e utilização
CREATE DATABASE EcommerceDB;
USE EcommerceDB;

-------------------------------------
-- Criação das Tabelas do Esquema
-------------------------------------

-- Tabela de Clientes (PF ou PJ, mas não ambos)
CREATE TABLE Cliente (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nome VARCHAR(100) NOT NULL,
    email VARCHAR(100) NOT NULL UNIQUE,
    telefone VARCHAR(20),
    tipo ENUM('PF', 'PJ') NOT NULL,
    cpf VARCHAR(14),
    cnpj VARCHAR(18),
    CHECK (
      (tipo = 'PF' AND cpf IS NOT NULL AND cnpj IS NULL) OR
      (tipo = 'PJ' AND cnpj IS NOT NULL AND cpf IS NULL)
    )
);

-- Tabela de Formas de Pagamento
CREATE TABLE FormaPagamento (
    id INT AUTO_INCREMENT PRIMARY KEY,
    descricao VARCHAR(50) NOT NULL
);

-- Associação entre Cliente e suas Formas de Pagamento (muitos para muitos)
CREATE TABLE ClientePagamento (
    cliente_id INT,
    forma_pagamento_id INT,
    PRIMARY KEY (cliente_id, forma_pagamento_id),
    FOREIGN KEY (cliente_id) REFERENCES Cliente(id),
    FOREIGN KEY (forma_pagamento_id) REFERENCES FormaPagamento(id)
);

-- Tabela de Fornecedores
CREATE TABLE Fornecedor (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nome VARCHAR(100) NOT NULL,
    contato VARCHAR(100),
```

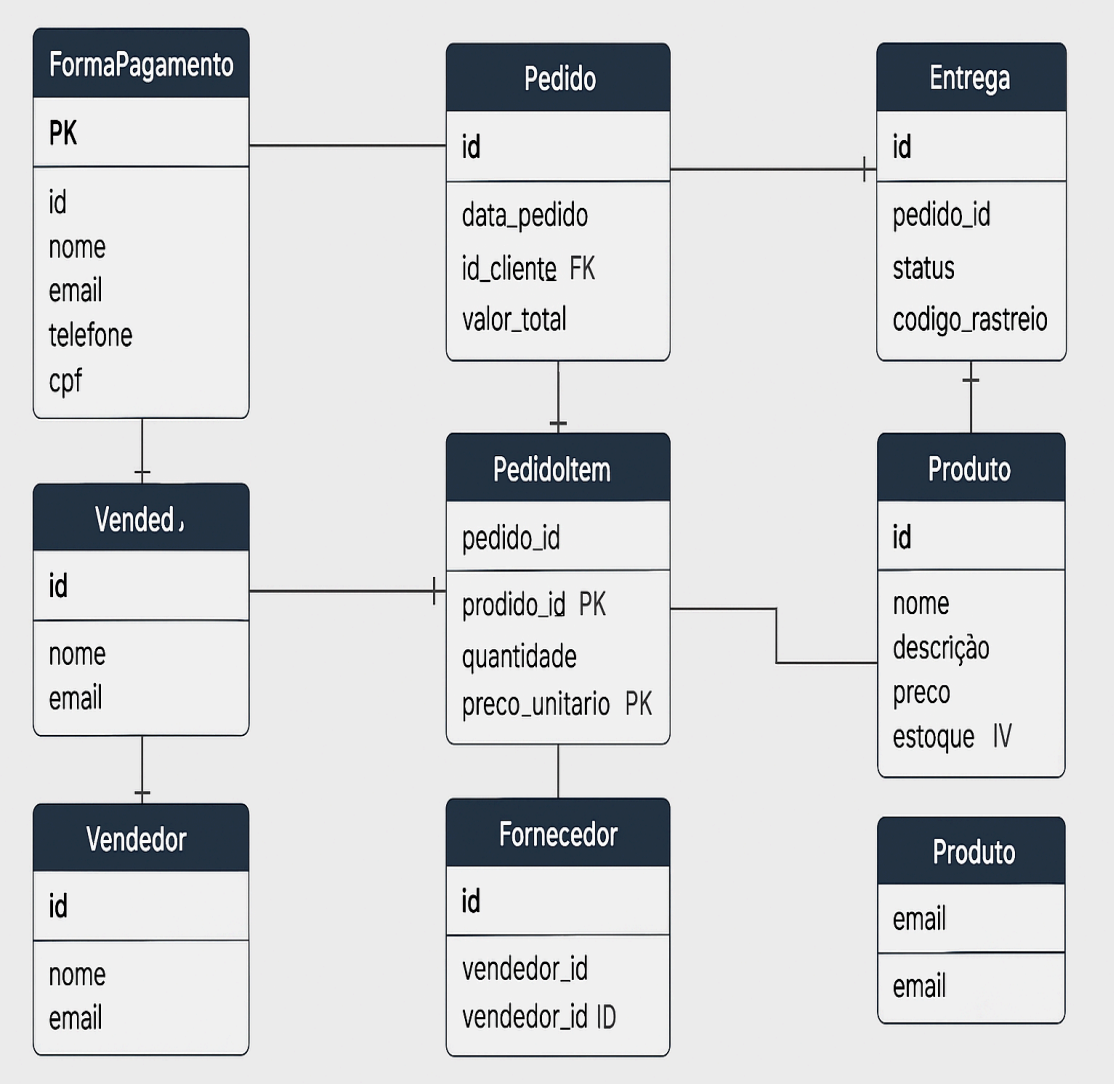
```
    -- Se o fornecedor também for vendedor, este campo receberá o id do vendedor
    vendedor_id INT DEFAULT NULL
);

-- Tabela de Vendedores
CREATE TABLE Vendedor (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nome VARCHAR(100) NOT NULL,
    email VARCHAR(100) NOT NULL UNIQUE
);

-- Se desejar relacionar Fornecedor e Vendedor, a constraint abaixo já foi definida na coluna
vendedor_id de Fornecedor:
ALTER TABLE Fornecedor
  ADD CONSTRAINT fk_fornecedor_vendedor FOREIGN KEY (vendedor_id)
REFERENCES Vendedor(id);

-- Tabela de Produtos (associados a um fornecedor)
CREATE TABLE Produto (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nome VARCHAR(100) NOT NULL,
    descricao TEXT,
    preco DECIMAL(10,2) NOT NULL,
    estoque INT DEFAULT 0,
    fornecedor_id INT,
    FOREIGN KEY (fornecedor_id) REFERENCES Fornecedor(id)
);

-- Tabela de Pedidos
CREATE TABLE Pedido (
    id INT AUTO_INCREMENT PRIMARY KEY,
    data_pedido DATE NOT NULL,
    id_cliente INT NOT NULL,
    valor_total DECIMAL(10,2) DEFAULT 0,
    FOREIGN KEY (id_cliente) REFERENCES Cliente(id)
);

-- Tabela de Itens do Pedido
CREATE TABLE PedidoItem (
    pedido_id INT NOT NULL,
    produto_id INT NOT NULL,
    quantidade INT NOT NULL,
    preco_unitario DECIMAL(10,2) NOT NULL,
    PRIMARY KEY (pedido_id, produto_id),
    FOREIGN KEY (pedido_id) REFERENCES Pedido(id),
    FOREIGN KEY (produto_id) REFERENCES Produto(id)
);
```

```
-- Tabela de Entregas (para cada pedido)
CREATE TABLE Entrega (
    id INT AUTO_INCREMENT PRIMARY KEY,
    pedido_id INT NOT NULL,
    status ENUM('Pendente', 'Em trânsito', 'Entregue', 'Cancelado') DEFAULT 'Pendente',
    codigo_rastreio VARCHAR(50),
    FOREIGN KEY (pedido_id) REFERENCES Pedido(id)
);

-------------------------------------
-- Inserção de Dados Fictícios
-------------------------------------

-- Inserindo Clientes (PF e PJ)
INSERT INTO Cliente (nome, email, telefone, tipo, cpf, cnpj) VALUES
('Ana Martins', 'ana.martins@example.com', '11912345678', 'PF', '123.456.789-00', NULL),
('Empresa Tech', 'contato@empresatech.com', '1132123456', 'PJ', NULL,
'12.345.678/0001-99'),
('Bruno Silva', 'bruno.silva@example.com', '11987654321', 'PF', '987.654.321-00', NULL);

-- Inserindo Formas de Pagamento
INSERT INTO FormaPagamento (descricao) VALUES
('Cartão de Crédito'),
('Boleto Bancário'),
('PayPal');

-- Associando Clientes às Formas de Pagamento
INSERT INTO ClientePagamento (cliente_id, forma_pagamento_id) VALUES
(1, 1),
(1, 2),
(2, 1),
(2, 3),
(3, 2);

-- Inserindo Vendedores
INSERT INTO Vendedor (nome, email) VALUES
('Carlos Vendas', 'carlos.vendas@example.com'),
('Mariana Comercial', 'mariana.comercial@example.com');

-- Inserindo Fornecedores (um dos fornecedores também atua como vendedor)
INSERT INTO Fornecedor (nome, contato, vendedor_id) VALUES
('Fornecedor A', 'contato@fornecedora.com', 1),
('Fornecedor B', 'contato@fornecedorb.com', NULL);

-- Inserindo Produtos
INSERT INTO Produto (nome, descricao, preco, estoque, fornecedor_id) VALUES
('Notebook X', 'Notebook com 16GB RAM, 512GB SSD', 4500.00, 10, 1),
('Smartphone Y', 'Smartphone com 8GB RAM, 128GB de armazenamento', 2500.00, 20, 2),
```

```
('Mouse Óptico', 'Mouse com sensor óptico', 50.00, 100, 1),
('Teclado Mecânico', 'Teclado com retroiluminação', 200.00, 50, 2);

-- Inserindo Pedidos
INSERT INTO Pedido (data_pedido, id_cliente, valor_total) VALUES
('2025-04-01', 1, 0),
('2025-04-02', 2, 0),
('2025-04-03', 3, 0);

-- Inserindo Itens dos Pedidos
INSERT INTO PedidoItem (pedido_id, produto_id, quantidade, preco_unitario) VALUES
(1, 1, 1, 4500.00), -- Ana Martins compra 1 Notebook X
(1, 3, 2, 50.00),   -- e 2 Mouse Óptico
(2, 2, 1, 2500.00), -- Empresa Tech compra 1 Smartphone Y
(2, 4, 1, 200.00),  -- e 1 Teclado Mecânico
(3, 3, 1, 50.00);   -- Bruno Silva compra 1 Mouse Óptico

-- Atualizando o valor_total do pedido com base nos itens inseridos (valor_total = soma(qtde
* preco_unitario))
UPDATE Pedido SET valor_total = (
    SELECT SUM(quantidade * preco_unitario)
    FROM PedidoItem
    WHERE PedidoItem.pedido_id = Pedido.id
);

-- Inserindo Entregas
INSERT INTO Entrega (pedido_id, status, codigo_rastreio) VALUES
(1, 'Em trânsito', 'TRACK123ABC'),
(2, 'Pendente', 'TRACK456DEF'),
(3, 'Entregue', 'TRACK789GHI');

--------------------------------------
-- Exemplos de Queries SQL Complexas
--------------------------------------

-- 1. Recuperação simples: Lista de todos os clientes
SELECT id, nome, email, telefone, tipo
FROM Cliente;

-- 2. Filtro com WHERE: Clientes do tipo 'PF'
SELECT id, nome, cpf
FROM Cliente
WHERE tipo = 'PF';

-- 3. Expressão derivada: Exibir pedido com valor_total acrescido de 10% de taxa de serviço
SELECT id, data_pedido, valor_total,
       (valor_total * 1.10) AS valor_com_taxa
FROM Pedido;
```

```
-- 4. Ordenação dos dados: Lista de produtos ordenados por estoque (do maior para o menor)
SELECT id, nome, estoque
FROM Produto
ORDER BY estoque DESC;

-- 5. Condição de filtros aos grupos – HAVING: Quantos pedidos foram feitos por cada cliente (somente para clientes com mais de 1 pedido)
SELECT c.nome, COUNT(p.id) AS total_pedidos
FROM Cliente c
JOIN Pedido p ON c.id = p.id_cliente
GROUP BY c.nome
HAVING COUNT(p.id) > 1;

-- 6. Junção entre tabelas: Relação de produtos com seus fornecedores, mostrando nome do produto, fornecedor e estoque
SELECT pr.nome AS produto, f.nome AS fornecedor, pr.estoque
FROM Produto pr
JOIN Fornecedor f ON pr.fornecedor_id = f.id;

-- 7. Consulta com JOIN e filtro: Relação de pedidos com entrega, exibindo cliente, data do pedido, status da entrega e código de rastreio
SELECT c.nome AS cliente, p.data_pedido, e.status, e.codigo_rastreio
FROM Pedido p
JOIN Cliente c ON p.id_cliente = c.id
JOIN Entrega e ON p.id = e.pedido_id
WHERE e.status <> 'Cancelado'
ORDER BY p.data_pedido;

-- 8. Consulta com expressão derivada e agrupamento:
-- Exibir o total faturado (soma dos pedidos) por cada cliente, mostrando somente clientes com faturamento acima de R$ 4000
SELECT c.nome, SUM(p.valor_total) AS faturamento_total,
       (SUM(p.valor_total) * 0.05) AS comissao_5_porcento
FROM Cliente c
JOIN Pedido p ON c.id = p.id_cliente
GROUP BY c.nome
HAVING SUM(p.valor_total) > 4000
```

Diagrama do modelo lógico


FormaPagamento
PK
id
nome
email
telefone
cpf
Pedido
id
data_pedido
id_cliente FK
valor_total
Entrega
id
pedido_id
status
codigo_rastreio
Vended,
id
nome
email
PedidoItem
pedido_id
prodido_id PK
quantidade
preco_unitario PK
Produto
id
nome
descrição
preco
estoque IV
Vendedor
id
nome
email
Fornecedor
id
vendedor_id
vendedor_id ID
Produto
email
email
;