Projeto Final Disciplina de BD

22/08/2023

Seu nome

Steffany Rodrigues;

Rodrigo Pinheiro;

Nimai Marchiori;

Livya Coelho;

Jhullian Turque;

Jefferson Moraes.

Visão geral

Criação de Tabelas:

São criadas tabelas para armazenar informações sobre categorias, produtos, clientes, funcionários, pedidos, itens de pedido e notas fiscais.

Chaves primárias e chaves estrangeiras são definidas para estabelecer relações entre as tabelas.

Inserção de Dados:

Diversos dados são inseridos nas tabelas, como categorias de produtos, produtos, clientes, funcionários e pedidos, juntamente com detalhes associados.

Atualizações e Exclusões:

O preço de um produto específico é atualizado.

A data de um pedido específico é atualizada.

Uma nota fiscal específica é excluída.

Consultas (SELECT):

São realizadas diversas consultas para obter informações do banco de dados.

São listados os pedidos com seus números, datas e nomes dos clientes correspondentes.

Produtos são listados juntamente com suas categorias e quantidades totais em estoque.

A quantidade de pedidos por cliente é listada.

Os detalhes de um pedido específico são listados, incluindo os produtos, quantidades, preços unitários e totais para construção de uma nota fiscal.

Produtos com estoque abaixo de 50 unidades são listados.

Objetivos

O objetivo do conjunto de códigos apresentado é criar e administrar um sistema de gerenciamento de vendas. Esse sistema visa controlar informações relacionadas a produtos, clientes, pedidos, funcionários e notas fiscais. Por meio da criação de tabelas, são definidas estruturas para armazenar dados relevantes, como detalhes de produtos, informações de clientes e registros de transações.

As consultas SQL incluídas possibilitam a obtenção de insights valiosos, como a listagem de pedidos por cliente, a quantidade de produtos em estoque e até mesmo a elaboração de notas fiscais a partir dos detalhes dos pedidos. Em essência, esse sistema busca centralizar e organizar informações cruciais para auxiliar na tomada de decisões, rastrear operações comerciais e oferecer um panorama abrangente das atividades de vendas.

Marcos

I. Diagrama ER (primeira versão)

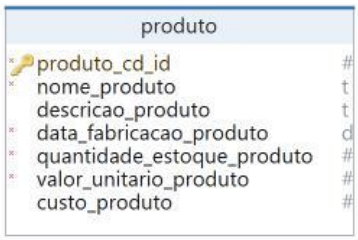
A. 1ª Forma Normal (1ª FN)

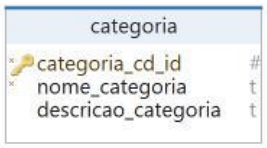
Para atender aos requisitos da 1ª FN em um sistema de gerenciamento de banco de dados como o PostgreSQL, uma tabela deve satisfazer às seguintes condições:

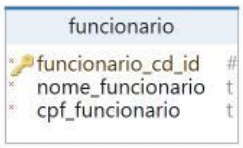
Valores Atômicos: Cada campo (coluna) em uma tabela deve conter apenas valores atômicos e não fracionáveis. Isso significa que cada campo deve conter apenas um único valor, em vez de listas ou conjuntos de valores.

Valores Não-Repetidos: Não deve haver repetição de dados. Isso significa que cada valor em um campo deve ser único e não pode ser replicado em outras partes da mesma tabela.

Identificação Única: Cada linha na tabela deve ter uma identificação única, normalmente representada por uma chave primária, que permite a distinção única de cada registro na tabela.







II. Diagrama ER (depois da normalização)

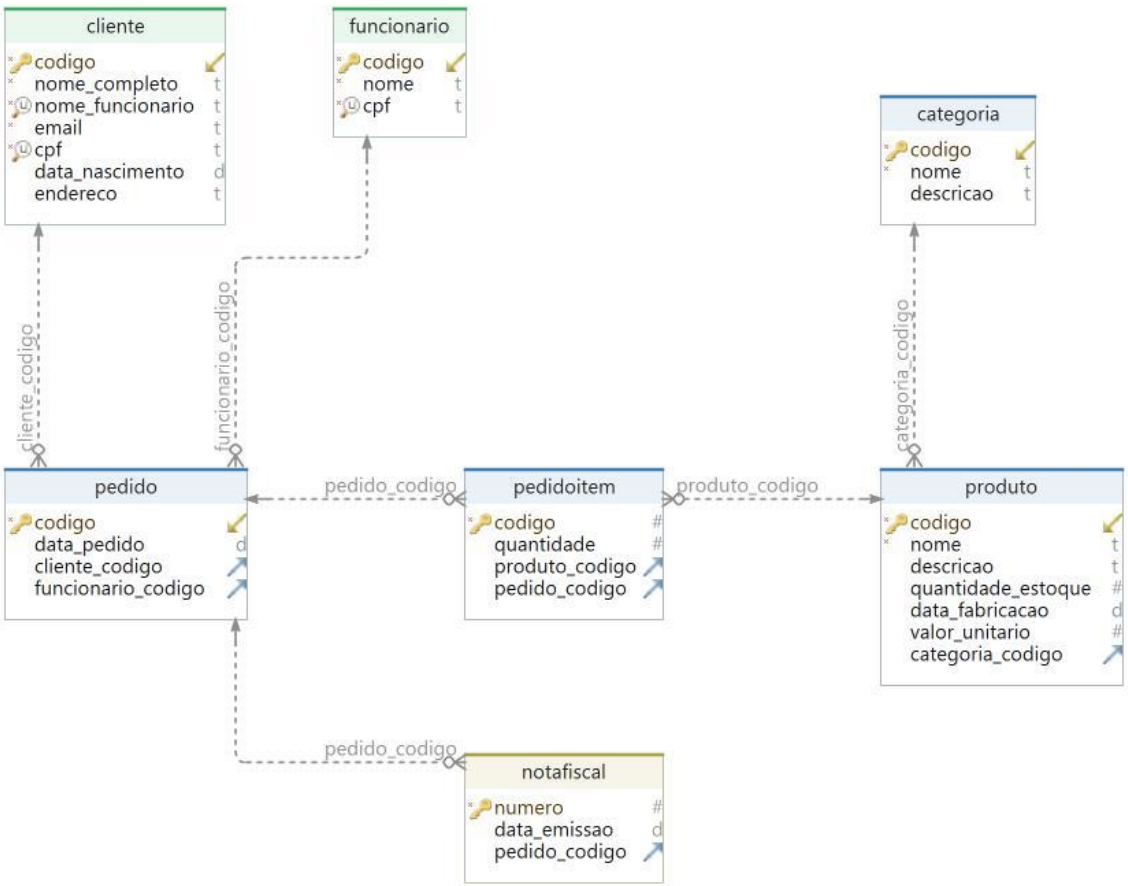
A. 2ª Forma Normal (2ª FN):

Para atender aos requisitos da 2ª FN em um sistema de gerenciamento de banco de dados como o PostgreSQL, uma tabela já deve estar na 1ª FN e deve satisfazer às seguintes condições:

Atender à 1ª Forma Normal: A tabela deve estar na 1ª Forma Normal, ou seja, deve conter apenas valores atômicos, sem repetições e com identificação única.

Eliminar Dependências Parciais: Cada atributo não chave (não faz parte da chave primária) deve ser totalmente dependente da chave primária. Isso significa que não deve haver atributos que dependem apenas de uma parte da chave primária.

Neste ponto analisamos que o Atributo “Endereço” é uma dependência parcial, com isso será aplicado na próxima etapa a 3ª Forma Normal (3ª fn).



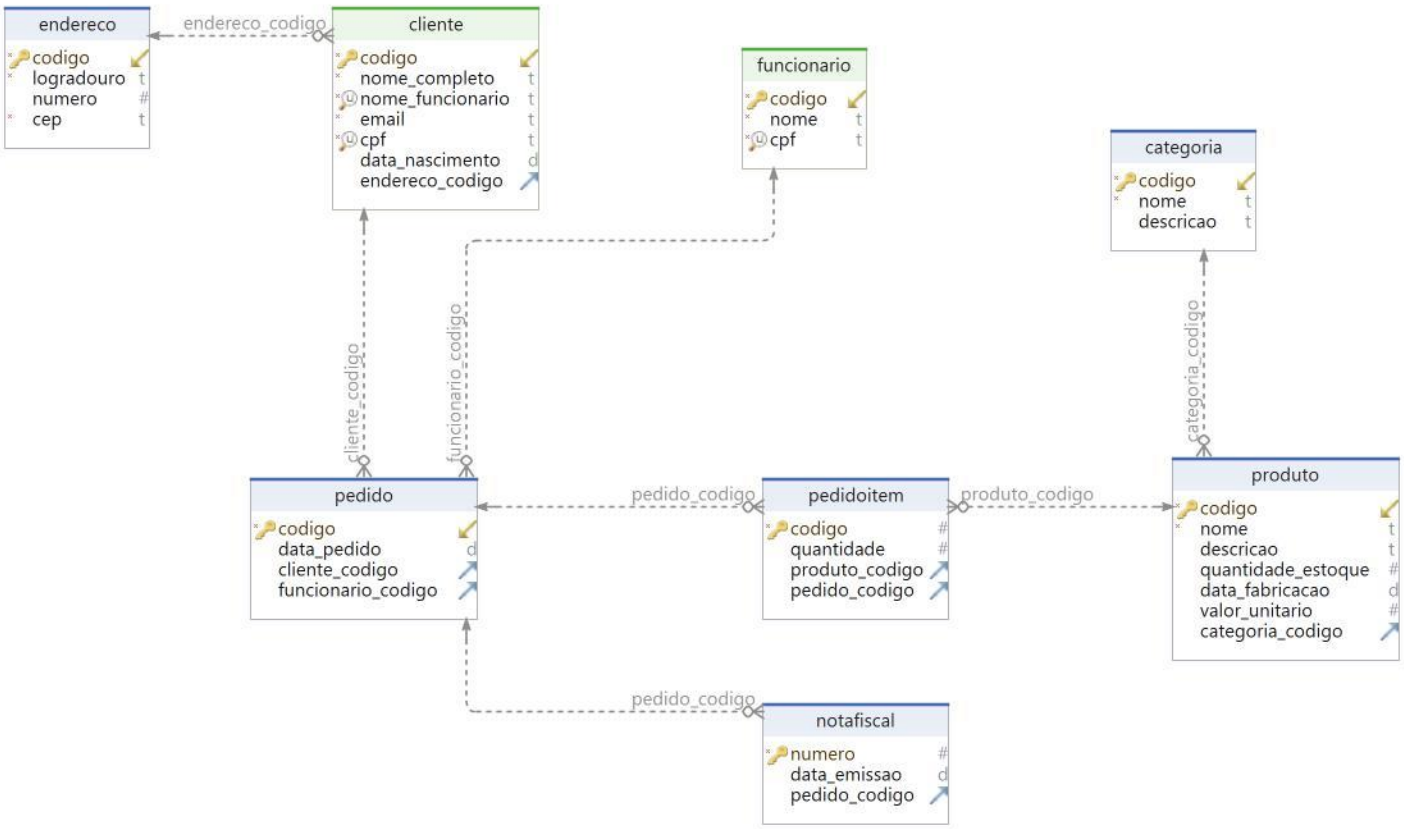
B. 3ª Forma Normal (3ª FN):

Para atender aos requisitos da 3ª FN em um sistema de gerenciamento de banco de dados como o PostgreSQL, uma tabela deve atender aos requisitos da 2ª FN e satisfazer as seguintes condições:

Atender à 2ª Forma Normal: A tabela deve estar na 2ª Forma Normal, o que significa que ela deve estar livre de dependências parciais.

Eliminar Dependências Transitivas: Cada atributo não chave deve ser funcionalmente dependente apenas da chave primária e não deve depender de outros atributos não chave.

Onde o Atributo “Endereço” tornou-se uma Entidade para as devidas aplicações de Forma Normal, sendo assim tendo o seu relacionamento com a respectiva tabela.



III. SQL de criação das tabelas

```sql
-- Criação da tabela Categoria
CREATE TABLE Categoria (
    codigo SERIAL PRIMARY KEY,
    nome VARCHAR(100) NOT NULL,
    descricao TEXT
);

-- Criação da tabela Endereço
CREATE TABLE Endereco (
    codigo SERIAL PRIMARY KEY,
    logradouro VARCHAR(100) NOT NULL,
    numero INT,
    cep VARCHAR(8) NOT NULL
);

-- Criação da tabela Produto
CREATE TABLE Produto (
    codigo SERIAL PRIMARY KEY,
    nome VARCHAR(100) NOT NULL,
    descricao TEXT,
    quantidade_estoque INT,
    data_fabricacao DATE,
    valor_unitario DECIMAL(10, 2),
    categoria_codigo INT,
    FOREIGN KEY (categoria_codigo) REFERENCES Categoria(codigo)
);
```

```sql
-- Criação da tabela Cliente
CREATE TABLE Cliente (
    codigo SERIAL PRIMARY KEY,
    nome_completo VARCHAR(100) NOT NULL,
    nome_funcionario VARCHAR(50) NOT NULL UNIQUE,
    email VARCHAR(100) NOT NULL,
    cpf VARCHAR(11) NOT NULL UNIQUE,
    data_nascimento DATE,
    endereco_codigo INT,
    FOREIGN KEY (endereco_codigo) REFERENCES Endereco (codigo)
);

-- Criação da tabela Funcionario
CREATE TABLE Funcionario (
    codigo SERIAL PRIMARY KEY,
    nome VARCHAR(100) NOT NULL,
    cpf VARCHAR(11) NOT NULL UNIQUE
);

-- Criação da tabela Pedido
CREATE TABLE Pedido (
    codigo SERIAL PRIMARY KEY,
    data_pedido DATE,
    cliente_codigo INT,
    Funcionario_codigo INT,
    FOREIGN KEY (cliente_codigo) REFERENCES Cliente(codigo),
    FOREIGN KEY (Funcionario_codigo) REFERENCES Funcionario(codigo)
);
```

```sql
-- Criação da tabela PedidoItem
CREATE TABLE PedidoItem (
    codigo SERIAL PRIMARY KEY,
    quantidade INT,
    produto_codigo INT,
    pedido_codigo INT,
    FOREIGN KEY (produto_codigo) REFERENCES Produto(codigo),
    FOREIGN KEY (pedido_codigo) REFERENCES Pedido(codigo)
);

-- Criação da tabela NotaFiscal
CREATE TABLE NotaFiscal (
    numero SERIAL PRIMARY KEY,
    data_emissao DATE,
    pedido_codigo INT,
    FOREIGN KEY (pedido_codigo) REFERENCES Pedido(codigo)
);
```

IV. SQL de inserção de dados nas tabelas (pelo menos 5 registros em cada uma)

```sql
-- Inserção de dados na tabela Categoria
INSERT INTO Categoria (nome, descricao) VALUES
    ('Eletrônicos', 'Produtos eletrônicos em geral'),
    ('Roupas', 'Roupas e acessórios'),
    ('Alimentos', 'Produtos alimentícios'),
    ('Livros', 'Livros de diversos gêneros'),
    ('Decoração', 'Itens de decoração para casa');

INSERT INTO Endereco (logradouro, numero, cep) VALUES
    ('Rua A', 123, '12345678'),
    ('Avenida B', 456, '23456789'),
    ('Praça C', 789, '34567890'),
    ('Alameda D', 1010, '45678901'),
    ('Estrada E', 2020, '56789012');

-- Inserção de dados na tabela Produto
INSERT INTO Produto (nome, descricao, quantidade_estoque,
data_fabricacao, valor_unitario, categoria_codigo) VALUES
    ('Samsung Galaxy S20', 'Um smartphone Samsung melhor que
iPhone', 100, '2023-01-01', 800.00, 1),
    ('Camiseta Nerd', 'Camiseta de algodão logo Caverna do Dragão',
200, '2023-02-01', 20.00, 2),
    ('Chocolate Lindt', 'Deliciosa barra de chocolate Lindt', 50,
'2023-03-01', 5.00, 3),
    ('O Código Da Vinci', 'Um best-seller de Dan Brown', 30,
'2023-02-15', 35.00, 4),
    ('Vaso Decorativo Japonês', 'Vaso de cerâmica da dinastia Ming',
15, '2023-03-10', 45.00, 5);
```

```sql
-- Inserção de dados na tabela Cliente
INSERT INTO Cliente (nome_completo, nome_Funcionario, email, cpf, data_nascimento, endereco_codigo) VALUES
    ('Ana Silva', 'ana123', 'ana@serratec.com', '51889965022', '1990-05-15', 1),
    ('João Pereira', 'joao456', 'joao@serratec.com', '42427849022', '1985-08-10', 2),
    ('Maria Santos', 'maria789', 'maria@serratec.com', '14012515012', '1995-03-20', 3),
    ('Pedro Souza', 'pedro101', 'pedro@serratec.com', '98611028090', '1988-11-05', 4),
    ('Luisa Lima', 'luisa2023', 'luisa@serratec.com', '91529952085', '2000-01-30', 5);

-- Inserção de dados na tabela Funcionario
INSERT INTO Funcionario (nome, cpf) VALUES
    ('Funcionário Jefferson', '52654737004'),
    ('Funcionário Jhullian', '20576483036'),
    ('Funcionário Nimai', '89927707070'),
    ('Funcionário Rodrigo', '83013042001'),
    ('Funcionário Steffany', '65827118060');

-- Inserção de dados na tabela Pedido
INSERT INTO Pedido (data_pedido, cliente_codigo, Funcionario_codigo)
VALUES
    ('2023-04-01', 1, 1),
    ('2023-04-02', 2, 2),
    ('2023-04-03', 3, 3),
    ('2023-04-04', 4, 4),
```

```
    ('2023-04-05', 5, 5),
    ('2023-04-15', 2, 1);


-- Inserção de dados na tabela PedidoItem
INSERT INTO PedidoItem (quantidade, produto_codigo, pedido_codigo)
VALUES
    (2, 1, 1),
    (3, 2, 2),
    (1, 3, 3),
    (4, 4, 4),
    (2, 5, 5);


-- Inserção de dados na tabela NotaFiscal
INSERT INTO NotaFiscal (data_emissao, pedido_codigo)
VALUES
    ('2023-04-01', 1),
    ('2023-04-02', 2),
    ('2023-04-03', 3),
    ('2023-04-04', 4),
    ('2023-04-05', 5);
```

V. Um comando SQL de atualização em algum registro em uma tabela

```sql
-- Atualização do preço do produto com código 1 (Smartphone)

UPDATE Produto

   SET valor_unitario = 850.00

WHERE codigo = 1;



-- Atualização da data de um pedido com código 1

UPDATE Pedido

   SET data_pedido = '2023-04-10'

WHERE codigo = 1;
```

VI. Um comando SQL de exclusão de algum registro em uma tabela

```sql
-- Exclusão de uma nota fiscal com número 2

DELETE FROM NotaFiscal

WHERE numero = 2;
```

VII. 5 SQLs de consulta

A. Pelo menos 2 com algum tipo de junção

```
 -- Lista os pedidos e os nomes dos clientes que fizeram esses
pedidos
 SELECT Pedido.codigo AS NumeroPedido, Pedido.data_pedido AS
DataPedido, Cliente.nome_completo AS NomeCliente
    FROM Pedido
    INNER JOIN Cliente ON Pedido.cliente_codigo = Cliente.codigo;


 -- Lista os produtos, suas categorias e a quantidade total de cada
produto em estoque
 SELECT Categoria.nome AS Categoria, Produto.nome AS Produto,
SUM(Produto.quantidade_estoque) AS QuantidadeTotal
    FROM Produto
    INNER JOIN Categoria ON Produto.categoria_codigo =
Categoria.codigo
 GROUP BY Categoria.nome, Produto.nome;
```

B. Pelo menos 1 com usando count() e group by()

```
 -- Lista a quantidade de pedidos realizados por cada cliente
 SELECT Cliente.nome_completo AS NomeCliente, COUNT(Pedido.codigo)
AS QuantidadePedidos
    FROM Cliente
    LEFT JOIN Pedido ON Cliente.codigo = Pedido.cliente_codigo
 GROUP BY Cliente.nome_completo;
```

C. 1 SQL para construção de nota fiscal

```
 -- Lista os detalhes de um pedido específico, incluindo os produtos e suas quantidades para construir uma nota fiscal
 SELECT Pedido.codigo AS NumeroPedido, Pedido.data_pedido AS DataPedido, Cliente.nome_completo AS NomeCliente,
       Produto.nome AS NomeProduto, PedidoItem.quantidade AS Quantidade, Produto.valor_unitario AS PrecoUnitario,
       (PedidoItem.quantidade * Produto.valor_unitario) AS ValorTotal
    FROM Pedido
    INNER JOIN Cliente ON Pedido.cliente_codigo = Cliente.codigo
    INNER JOIN PedidoItem ON Pedido.codigo = PedidoItem.pedido_codigo
    INNER JOIN Produto ON PedidoItem.produto_codigo = Produto.codigo
 WHERE Pedido.codigo = 1;
```

D. Consulta Adicional (5ª consulta)

```
 -- Lista todos os produtos com estoque abaixo de 50 unidades
 SELECT Produto.nome AS NomeProduto, Produto.quantidade_estoque AS QuantidadeEstoque
    FROM Produto
 WHERE Produto.quantidade_estoque < 50;
```