FACULDADE SENAI DE TECNOLOGIA GASPAR RICARDO JÚNIOR

GRADUAÇÃO EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

Giovani Miamoto
Luciano Júnior
Victor Geroto
Rafael Pecorari

Trabalho de Banco de Dados

SOROCABA - SP
2025

Giovani Miamoto
Luciano Júnior
Victor Geroto
Rafael Pecorari

Trabalho de Banco de Dados

Documentação apresentanda à Faculdade SENAI de Tecnologia Gaspar Ricardo Júnior, para a entrega no dia 20/05, como requisito da matéria de ciência de dados na graduação em Análise e Desenvolvimento de Sistemas sob a orientação do professor:

Prof. André Cassulino Araujo Souza

SOROCABA - SP
2025

FACULDADE SENAI DE TECNOLOGIA GASPAR RICARDO JÚNIOR

Trabalho de Banco de Dados

SUMÁRIO

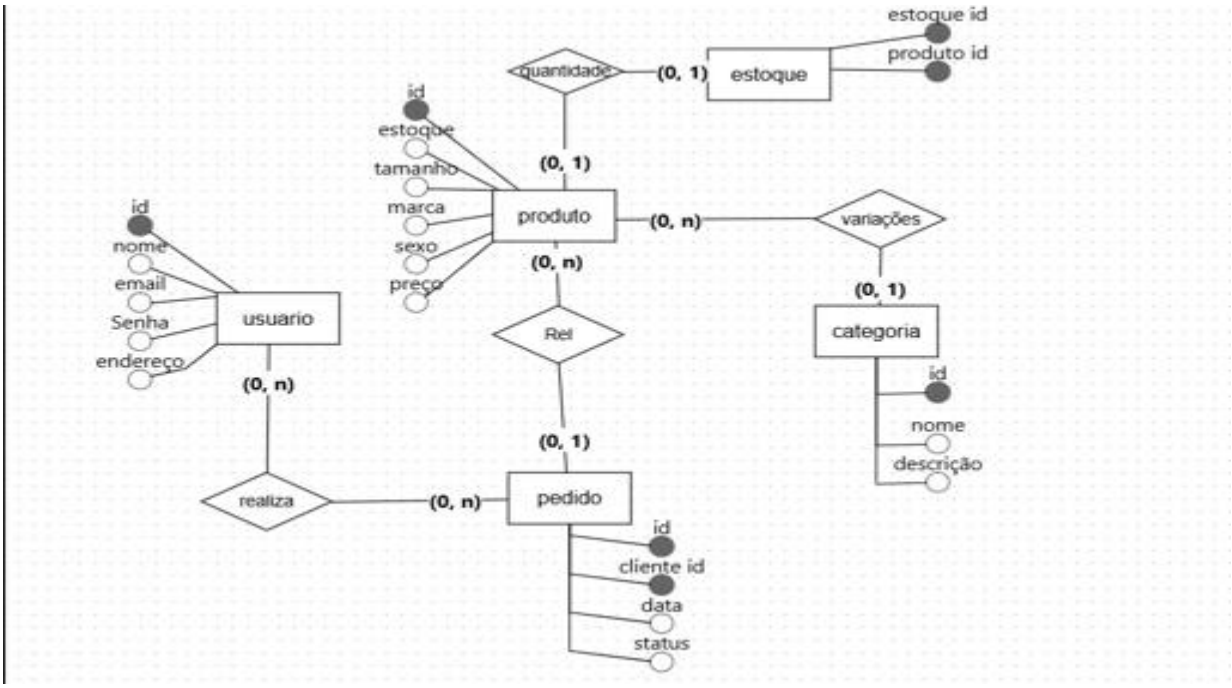
Sumário

INTRODUÇÃO

Este projeto de Banco de Dados tem como foco o desenvolvimento de um sistema relacional para o gerenciamento de uma loja virtual de calçados e acessórios. O objetivo principal é estruturar e organizar as informações relacionadas a usuários, produtos, categorias e pedidos, garantindo integridade e eficiência no armazenamento e recuperação dos dados.

A escolha desse tema se justifica pela ampla aplicabilidade no comércio eletrônico, um setor em constante crescimento, e pela oportunidade de aplicar na prática os conceitos aprendidos sobre modelagem de dados, normalização e implementação de bancos relacionais. Além disso, o projeto possibilita o uso de linguagens de manipulação de dados como SQL, promovendo uma visão integrada entre teoria e prática na área de sistemas de informação.

Modeladem Conceitual



Descrição das Entidades e Relacionamentos

Usuario:

- **Descrição**: Representa os clientes cadastrados na loja.
- **Atributos**: id, nome, email, senha, endereço
- **Relacionamentos**:
    - Um usuário pode realizar zero ou mais pedidos.
    - Cada pedido é obrigatoriamente vinculado a um usuário.

Produto:

- **Descrição**: Representa os produtos disponíveis para venda na loja.
- **Atributos**: id, marca, sexo, tamanho, preço
- **Relacionamentos**:
    - Cada produto pertence a uma única categoria.
    - Um produto pode estar associado a um ou mais pedidos.
    - Cada produto possui uma entrada no estoque com sua quantidade disponível.

Categoria

- **Descrição**: Agrupa os produtos em tipos como calçados, tênis, acessórios etc.
- **Atributos**: id, nome, descrição

- **Relacionamentos**:
    - Uma categoria pode conter vários produtos.
    - Um produto pertence a apenas uma categoria.

Pedido

- **Descrição**: Armazena informações das compras realizadas pelos usuários.
- **Atributos**: id, cliente_id, data, status
- **Relacionamentos**:
    - Cada pedido está vinculado a um único usuário.
    - Um pedido pode conter vários produtos por meio de uma relação N:N (item_pedido).
    -

Estoque

- **Descrição**: Controla a quantidade disponível de cada produto no sistema.
- **Atributos**: estoque_id, produto_id, quantidade
- **Relacionamentos**:
    - Cada produto possui uma única entrada de estoque associada.
    - Relacionamento de um para um com o produto.

.

Modelagem Lógica

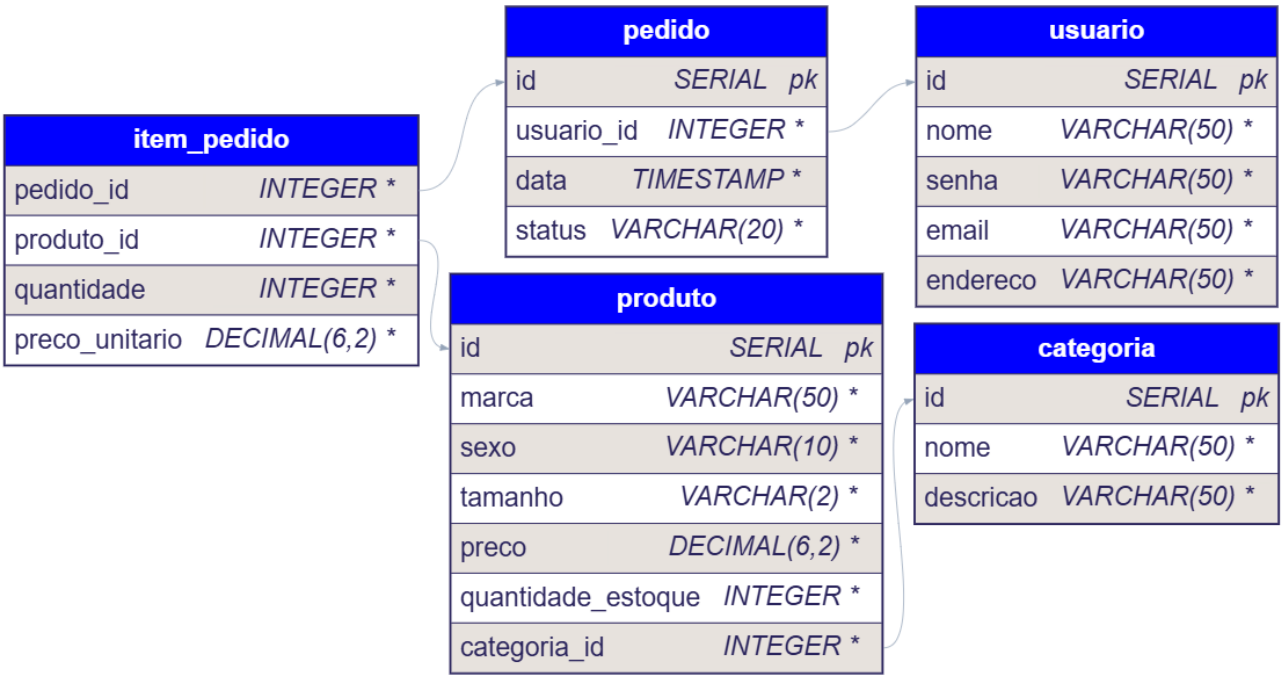
Transformações do DER para o Modelo Relacional

A modelagem lógica foi realizada a partir do Diagrama Entidade-Relacionamento (DER), convertendo entidades e relacionamentos em tabelas relacionais com integridade referencial.

A seguir, as principais transformações aplicadas:

- **Entidades (usuario, produto, categoria, pedido)** se tornaram tabelas com atributos diretamente correspondentes aos identificadores e dados descritos no DER.
- Os **relacionamentos entre entidades** foram transformados em **chaves estrangeiras**:
    - produto.categoria_id faz referência à tabela categoria.
    - pedido.usuario_id faz referência à tabela usuario.
    - A tabela associativa item_pedido foi criada para representar o relacionamento N:N entre produto e pedido, incluindo os atributos adicionais quantidade e preco_unitario.
- As **chaves primárias** (id) foram atribuídas com o tipo SERIAL para gerar valores automáticos.
- As **chaves estrangeiras** foram definidas com restrições de integridade, garantindo consistência dos dados entre tabelas relacionadas.

Figura: Modelo Lógico do Banco de Dados da Loja Virtual



Discussão sobre Normalização

Durante a modelagem lógica, foram aplicados os princípios de normalização com o objetivo de:

- Eliminar redundâncias,
- Garantir a integridade dos dados,
- Melhorar a estrutura e facilitar a manutenção.

Forma Normal Aplicada

- **1ª Forma Normal (1FN)**: Todos os atributos são atômicos e não repetitivos.
  Exemplo: Atributos como nome, email, preco, tamanho são campos únicos e indivisíveis.
- **2ª Forma Normal (2FN)**: Todas as tabelas dependem totalmente da chave primária.
  Exemplo: Na tabela produto, todos os atributos (marca, sexo, etc.) dependem exclusivamente do id do produto.
- **3ª Forma Normal (3FN)**: Não existem dependências transitivas.
  Exemplo: A descrição da categoria está armazenada apenas na tabela categoria e não duplicada nos produtos.

Justificativa da Normalização

A normalização foi adotada até a 3ª Forma Normal, pois:

- Atende bem às necessidades de consistência e organização do sistema.
- Evita redundância de dados (como duplicação de nomes de categoria em produtos).
- Mantém a flexibilidade para futuras alterações, como inserção de novos atributos ou tabelas.

Estrutura do Banco de Dados

Scripts de Criação (DDL)

A seguir, apresentamos os comandos SQL utilizados para criar as tabelas do banco de dados da loja virtual:

```
CREATE TABLE usuario (
    id SERIAL PRIMARY KEY,
    nome VARCHAR(50) NOT NULL,
    senha VARCHAR(50) NOT NULL,
    email VARCHAR(50) NOT NULL UNIQUE,
    endereco VARCHAR(50) NOT NULL
);

CREATE TABLE categoria (
    id SERIAL PRIMARY KEY,
    nome VARCHAR(50) NOT NULL UNIQUE,
    descricao VARCHAR(50) NOT NULL
);

CREATE TABLE produto (
    id SERIAL PRIMARY KEY,
    marca VARCHAR(50) NOT NULL,
    sexo VARCHAR(10) NOT NULL,
    tamanho VARCHAR(2) NOT NULL,
    preco DECIMAL(6,2) NOT NULL,
    quantidade_estoque INTEGER NOT NULL DEFAULT 0,
    categoria_id INTEGER NOT NULL,
    FOREIGN KEY (categoria_id) REFERENCES categoria(id)
);

CREATE TABLE pedido (
    id SERIAL PRIMARY KEY,
    usuario_id INTEGER NOT NULL,
    data TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
    status VARCHAR(20) NOT NULL,
    FOREIGN KEY (usuario_id) REFERENCES usuario(id) ON DELETE CASCADE
);

CREATE TABLE item_pedido (
    pedido_id INTEGER NOT NULL,
    produto_id INTEGER NOT NULL,
    quantidade INTEGER NOT NULL,
```

```
    preco_unitario DECIMAL(6,2) NOT NULL,
    PRIMARY KEY (pedido_id, produto_id),
    FOREIGN KEY (pedido_id) REFERENCES pedido(id) ON DELETE CASCADE,
    FOREIGN KEY (produto_id) REFERENCES produto(id)
);
```

Descrição das Tabelas

Tabela: usuario

Armazena informações dos clientes cadastrados na loja.

- id: Identificador único do usuário (chave primária)
- nome: Nome completo do usuário
- senha: Senha de acesso
- email: Endereço de e-mail (único)
- endereco: Endereço físico do usuário

Tabela: categoria

Define os grupos aos quais os produtos pertencem.

- id: Identificador da categoria (chave primária)
- nome: Nome da categoria (ex: Tênis, Acessórios)
- descricao: Descrição resumida da categoria

Tabela: produto

Contém os itens disponíveis para compra.

- id: Identificador do produto (chave primária)
- marca: Marca do produto
- sexo: Gênero (ex: Masculino, Feminino, Unissex)
- tamanho: Tamanho (ex: 38, 40, M)
- preco: Valor do produto
- quantidade_estoque: Quantidade atual disponível
- categoria_id: Referência à tabela categoria (chave estrangeira)

Tabela: pedido

Registra as compras feitas pelos usuários.

- id: Identificador do pedido (chave primária)
- usuario_id: Referência ao usuário que fez o pedido (chave estrangeira)
- data: Data e hora da realização do pedido
- status: Situação do pedido (ex: Pendente, Enviado, Concluído)

Tabela: item_pedido

Representa a associação entre produtos e pedidos (relacionamento N:N).

- pedido_id: Referência ao pedido (chave estrangeira)
- produto_id: Referência ao produto (chave estrangeira)
- quantidade: Quantidade do produto no pedido
- preco_unitario: Preço do produto no momento da compra

Relacionamentos entre as Tabelas

- usuario → pedido: Um usuário pode fazer vários pedidos, mas cada pedido pertence a apenas um usuário.
- categoria → produto: Cada produto pertence a uma categoria. Uma categoria pode conter diversos produtos.
- produto → item_pedido: Um produto pode estar em vários pedidos. Cada pedido pode conter vários produtos. Isso forma um relacionamento muitos-para-muitos, intermediado pela tabela item_pedido.
- pedido → item_pedido: Cada pedido pode conter vários produtos (itens).
- As chaves estrangeiras garantem a integridade referencial entre as tabelas.

Manipulação de Dados

Scripts de Inserção de Dados (INSERT) – Categorias

--categoria
INSERT INTO categoria (nome, descricao) VALUES
('Calçados', 'Calçados em geral'),
('Tênis', 'Tênis de diversas marcas'),
('Acessórios', 'Óculos, bonés e mais');

-- Usuários
INSERT INTO usuario (nome, senha, email, endereco) VALUES
('Ana Costa', 'ana321', 'ana.costa@email.com', 'Rua das Flores, 101'),
('Bruno Lima', 'bruno654', 'bruno.lima@email.com', 'Av. Central, 202'),
('Fernanda Rocha', 'fern789', 'fernanda.rocha@email.com', 'Rua do Sol, 303'),
('Lucas Martins', 'lucas987', 'lucas.martins@email.com', 'Av. Paulista, 404'),
('Patrícia Mendes', 'patri123', 'patricia.mendes@email.com', 'Rua Verde, 505'),
('Diego Fernandes', 'dieg456', 'diego.fernandes@email.com', 'Rua Azul, 606'),
('Juliana Pires', 'juli789', 'juliana.pires@email.com', 'Rua do Comércio, 707'),
('Rafael Nogueira', 'rafa987', 'rafael.nogueira@email.com', 'Av. dos Andradas, 808'),
('Camila Ribeiro', 'cami654', 'camila.ribeiro@email.com', 'Rua das Palmeiras, 909'),
('Thiago Alves', 'thi321', 'thiago.alves@email.com', 'Av. Brasil, 1001');

-- Produtos
INSERT INTO produto (marca, sexo, tamanho, preco, quantidade_estoque, categoria_id) VALUES
('Nike', 'Masculino', '40', 149.90, 10, 1),
('Adidas', 'Feminino', '36', 129.99, 5, 1),
('Puma', 'Unissex', '42', 199.50, 8, 1),
('Under Armour', 'Masculino', '43', 179.90, 12, 1),
('Reebok', 'Feminino', '38', 119.90, 6, 1),
('Fila', 'Unissex', '44', 139.90, 4, 1),
('Vans', 'Masculino', '42', 299.99, 7, 2),
('Nike', 'Feminino', '38', 349.99, 9, 2),
('Adidas', 'Unissex', '40', 319.90, 5, 2),
('New Balance', 'Masculino', '41', 279.90, 6, 2),
('Asics', 'Feminino', '37', 289.90, 8, 2),
('Converse', 'Unissex', '39', 199.99, 10, 2),
('Ray-Ban', 'Unissex', 'U', 399.90, 2, 3),
('Oakley', 'Masculino', 'U', 359.90, 3, 3),
('Guess', 'Feminino', 'U', 249.90, 4, 3),
('Tommy Hilfiger', 'Unissex', 'U', 299.99, 6, 3),
('Nike', 'Masculino', '44', 159.90, 5, 1),
('Adidas', 'Feminino', '39', 139.90, 6, 1),
('Puma', 'Masculino', '36', 149.90, 7, 1),
('Vans', 'Unissex', '42', 169.90, 8, 1),
('Reebok', 'Masculino', '40', 129.90, 5, 1),
('Fila', 'Feminino', '36', 119.90, 4, 1),
('New Balance', 'Unissex', '43', 139.90, 3, 1),
('Oakley', 'Masculino', '41', 309.90, 2, 2),

('Tommy Hilfiger', 'Feminino', '39', 269.90, 3, 3);

-- Pedidos
INSERT INTO pedido (usuario_id, status) VALUES
(1, 'Em processamento'),
(2, 'Enviado'),
(3, 'Entregue'),
(1, 'Cancelado'),
(4, 'Entregue'),
(5, 'Em processamento'),
(2, 'Em processamento'),
(3, 'Enviado');

-- Itens do pedido
INSERT INTO item_pedido (pedido_id, produto_id, quantidade, preco_unitario)
VALUES
(1, 1, 2, 149.90),
(1, 7, 1, 299.99),
(2, 2, 1, 129.99),
(2, 8, 1, 349.99),
(3, 3, 1, 199.50),
(3, 13, 1, 399.90),
(4, 5, 2, 119.90),
(5, 4, 1, 179.90),
(5, 9, 1, 319.90),
(6, 6, 2, 139.90),
(6, 14, 1, 359.90),
(7, 11, 1, 289.90),
(7, 10, 1, 279.90),
(8, 18, 1, 139.90),
(8, 24, 1, 309.90);

-- Atualizando o status do pedido 1 para 'Enviado'
UPDATE pedido
SET status = 'Enviado'
WHERE id = 1;

-- Removendo o item do pedido 5 referente ao produto 9
DELETE FROM item_pedido
WHERE pedido_id = 5 AND produto_id = 9;

Consultas SQL (DQL)

Esta seção apresenta as principais consultas utilizadas para extrair informações do banco de dados.

Listar todos os usuários cadastrados

```
SELECT * FROM usuario;
```

Listar todos os produtos junto às suas categorias

```
SELECT p.*, c.nome AS categoria
FROM produto p
JOIN categoria c ON p.categoria_id = c.id;
```

Listar todos os pedidos com o nome do usuário que os realizou

```
SELECT pedido.*, usuario.nome AS nome_usuario
FROM pedido
JOIN usuario ON pedido.usuario_id = usuario.id;
```

Listar os itens de um pedido específico (ex: pedido 1), com o nome do produto e a quantidade

```
SELECT item_pedido.quantidade,
item_pedido.preco_unitario, produto.marca
FROM item_pedido
JOIN produto ON item_pedido.produto_id = produto.id
WHERE item_pedido.pedido_id = 1;
```

Somar o valor total de cada pedido

```
SELECT pedido_id, SUM(quantidade * preco_unitario) AS
total_pedido
FROM item_pedido
GROUP BY pedido_id;
```

Listar todos os produtos da categoria "Calçado”

```
SELECT p.*
FROM produto p
JOIN categoria c ON p.categoria_id = c.id
WHERE c.nome = 'Calçados';
```

Listar pedidos com status “Em processamento”

```
SELECT *
FROM pedido
WHERE status = 'Em processamento';
```

Ordenar os produtos do mais caro para o mais barato

```
SELECT *
FROM produto
ORDER BY preco DESC;
```

Quantidade de pedidos realizados por cada usuário

```
SELECT u.nome, COUNT(p.id) AS total_pedidos
FROM pedido p
JOIN usuario u ON p.usuario_id = u.id
GROUP BY u.nome;
```

Conclusão

Resumo do Processo de Desenvolvimento

O projeto consistiu na criação de um banco de dados relacional para uma loja virtual, englobando desde o planejamento das entidades e seus relacionamentos até a implementação dos scripts de DDL (estrutura), DML (manipulação de dados) e DQL (consultas). O modelo foi baseado em um cenário realista de comércio eletrônico, contemplando usuários, produtos, categorias, pedidos e itens de pedido.

Além da modelagem e implementação do banco, também foram aplicadas consultas para análise dos dados, o que permitiu visualizar informações úteis como: produtos mais caros, pedidos em andamento, clientes mais ativos e o valor total por pedido.

Lições Aprendidas

Durante o desenvolvimento, foram adquiridas e reforçadas as seguintes competências:

- Estruturação de bancos de dados relacionais com integridade referencial.
- Escrita de scripts SQL para criação, inserção, consulta e atualização de dados.
- Entendimento sobre modelagem lógica e conceitual com DER.
- Valorização do planejamento prévio antes da implementação.

Possíveis Melhorias no Projeto

- Criação de uma interface gráfica para o sistema.
- Inclusão de novas tabelas, como métodos de pagamento ou histórico de alterações nos pedidos.
- Implementação de gatilhos (triggers) e procedimentos armazenados (stored procedures).
- Normalização adicional para garantir maior eficiência e evitar redundâncias.

Referências

- HEUSER, Carlos A. *Projeto de Banco de Dados*. 6. ed. Bookman, 2009.
- SILBERSCHATZ, Abraham; KORTH, Henry F.; SUDARSHAN, S. *Sistemas de Banco de Dados*. 6. ed. São Paulo: Pearson, 2013.
- W3Schools. SQL Tutorial
- PostgreSQL Documentation. https://www.postgresql.org/docs/