Fundamentos de Modelagem Relacional e SQL [25E1_3]

Prof. Elberth Moraes

Elias Matos Martins

Teste de Performance - 1

Instituto Infnet

04 de março de 2025

Projeto de Banco de Dados Relacional

Tema: Sistema de Gerenciamento de Biblioteca

Este projeto tem como objetivo desenvolver um sistema que auxilie no gerenciamento do acervo de uma biblioteca, controle de empréstimos e cadastro de usuários. A seguir, apresenta-se a estrutura do projeto, abordando desde a evolução dos bancos de dados relacionais até a documentação e sugestões de melhorias.

Etapa 1: Modelagem de Dados Relacionais

Exercício 1: Evolução dos Bancos de Dados Relacionais

Contexto Histórico:
Em 1970, Edgar F. Codd apresentou o modelo relacional, revolucionando a forma de organizar e acessar dados. Essa abordagem permitiu a independência dos dados, facilitando a manipulação e integridade das informações.

Marcos Importantes:

- Modelo Relacional (1970): Base para os SGBDs modernos;
- Popularização do SQL: Linguagem padrão para consulta e manipulação de dados;
- Adoção em larga escala: Com o surgimento de sistemas como Oracle, MySQL e PostgreSQL, a abordagem relacional se consolidou em aplicações corporativas e web.

Relevância Atual:
Apesar da ascensão de bancos NoSQL, os bancos relacionais continuam sendo referência em integridade, confiabilidade e suporte a transações, essenciais para sistemas críticos.

Exercício 2: Vantagens das Boas Práticas

- Boas Práticas e Problemas Evitados:
  - Normalização dos Dados:
    Evita a redundância e inconsistências, garantindo que cada informação seja armazenada uma única vez.
  - Definição Clara de Chaves (Primárias e Estrangeiras):
    Assegura a integridade referencial, prevenindo a existência de registros órfãos ou duplicados.
  - Aplicação Adequada de Restrições (NOT NULL, UNIQUE):
    Garante que dados essenciais não sejam nulos e que informações críticas, como e-mails ou ISBNs, sejam únicos, evitando duplicidade e erros na recuperação dos dados.

Exercício 3: Terminologia de Banco de Dados

- **Tabela:**
  Conjunto de dados organizados em linhas (registros) e colunas (atributos).
  Exemplo: Tabela `Livros` contendo informações de cada obra.
- **Atributo:**
  Cada coluna de uma tabela que representa uma característica dos dados.
  Exemplo: `Título`, `Ano`, `ISBN` na tabela `Livros`.
- **Chave Primária:**
  Atributo (ou conjunto de atributos) que identifica unicamente cada registro de uma tabela.
  Exemplo: `id_livro` na tabela `Livros`.
- **Chave Estrangeira:**
  Atributo que referencia a chave primária de outra tabela, estabelecendo a relação entre elas.
  Exemplo: `id_usuario` na tabela `Emprestimos` que referencia o `id` da tabela `Usuarios`.
- **Normalização:**
  Processo de organizar os dados para minimizar redundâncias e dependências indesejadas, dividindo as informações em tabelas adequadas (1FN, 2FN, 3FN).

Exercício 4: Definição do Projeto

- **Tema e Problema:**
  Desenvolver um sistema para gerenciamento de biblioteca que permita o cadastro de livros, controle de empréstimos e registro dos usuários, facilitando o acesso e a manutenção do acervo.
- **Entidades e Atributos Principais:**
  - Livros:
    - id_livro (chave primária)
    - titulo
    - autor
    - ano_publicacao
    - editora
    - ISBN
    - estoque
  - Usuários:
    - id_usuario (chave primária)
    - nome
    - email
    - telefone
    - data_cadastro

    - Empréstimos:
        - id_emprestimo (chave primária)
        - data_emprestimo
        - data_devolucao
        - id_usuario (chave estrangeira)
        - id_livro (chave estrangeira)

Exercício 5: Relacionamento entre Tabelas

- **Diagrama de Relacionamento:**
    - Relação entre Usuários e Empréstimos:
      Um usuário pode ter vários empréstimos (1:N).
    - Relação entre Livros e Empréstimos:
      Um livro pode ser emprestado várias vezes, mas cada empréstimo se refere a um único livro (1:N).

Exercício 6: Normalização

- **Aplicação da Normalização:**
    - **Primeira Forma Normal (1FN):**
      Cada tabela armazena dados atômicos e únicos; não há grupos repetitivos.
    - **Segunda Forma Normal (2FN):**
      Todos os atributos não-chave dependem integralmente da chave primária. Por exemplo, na tabela `Livros`, atributos como `titulo` e `ISBN` dependem diretamente do `id_livro`.
    - **Terceira Forma Normal (3FN):**
      Remoção de dependências transitivas. Caso haja atributos que dependam de outros não-chave, estes são separados em novas tabelas (por exemplo, se informações detalhadas sobre o autor forem necessárias, pode ser criada a tabela `Autores`).

Etapa 2: Projeto de uma Base de Dados

Exercício 7: Uso de UNIQUE e NOT NULL

- **Exemplos de Restrições:**
  - **Tabela `Usuarios`:**
    - `email` deve ser **UNIQUE** para evitar duplicidade.
    - `nome` e `email` devem ser **NOT NULL** para garantir que dados essenciais sejam fornecidos.
  - **Tabela `Livros`:**
    - `ISBN` deve ser **UNIQUE** e **NOT NULL** para identificar cada livro de forma única.
  - **Tabela `Emprestimos`:**
  - Campos de data (`data_emprestimo`, `data_devolucao`) devem ser **NOT NULL** para manter o controle correto do empréstimo.

Essas restrições asseguram a integridade dos dados e evitam inconsistências como cadastros duplicados ou registros incompletos.

Exercício 8: Implementação das Tabelas em SQL

```
-- Criação da tabela Usuarios
CREATE TABLE Usuarios (
    id_usuario SERIAL PRIMARY KEY, -- Identificador único
    nome VARCHAR(100) NOT NULL,
    email VARCHAR(100) NOT NULL UNIQUE,
    telefone VARCHAR(20),
    data_cadastro DATE NOT NULL DEFAULT CURRENT_DATE
);

-- Criação da tabela Livros
CREATE TABLE Livros (
    id_livro SERIAL PRIMARY KEY,
    titulo VARCHAR(200) NOT NULL,
    autor VARCHAR(100) NOT NULL,
    ano_publicacao INT,
    editora VARCHAR(100),
    ISBN VARCHAR(20) NOT NULL UNIQUE,
    estoque INT NOT NULL DEFAULT 0
);
```

```
-- Criação da tabela Emprestimos
CREATE TABLE Emprestimos (
    id_emprestimo SERIAL PRIMARY KEY,
    data_emprestimo DATE NOT NULL,
    data_devolucao DATE NOT NULL,
    id_usuario INT NOT NULL,
    id_livro INT NOT NULL,
    CONSTRAINT fk_usuario FOREIGN KEY (id_usuario) REFERENCES
Usuarios(id_usuario),
    CONSTRAINT fk_livro FOREIGN KEY (id_livro) REFERENCES
Livros(id_livro)
);
```

Exercício 9: Inserção de Dados

```
-- Inserção de dados na tabela Usuarios

INSERT INTO Usuarios (nome, email, telefone) VALUES
('Ana Silva', 'ana.silva@example.com', '11999990001'),
('Bruno Costa', 'bruno.costa@example.com', '11999990002'),
('Carla Souza', 'carla.souza@example.com', '11999990003'),
('Diego Lima', 'diego.lima@example.com', '11999990004'),
('Elisa Ferreira', 'elisa.ferreira@example.com',
'11999990005');

-- Inserção de dados na tabela Livros

INSERT INTO Livros (titulo, autor, ano_publicacao, editora,
ISBN, estoque) VALUES
('Introdução ao SQL', 'João Pereira', 2015, 'TechBooks',
'9781234567897', 5),
('Modelagem de Dados', 'Maria Santos', 2018, 'InfoPress',
'9789876543210', 3),
('Banco de Dados Avançado', 'Carlos Mendes', 2020,
'DataBooks', '9781122334455', 4),
('Teoria dos Bancos de Dados', 'Mariana Rocha', 2012,
'EducaTech', '9786677889900', 2),
('Sistemas de Informação', 'Rafael Alves', 2019, 'SoftBooks',
'9785544332211', 6);
```

```
-- Inserção de dados na tabela Emprestimos

INSERT INTO Emprestimos (data_emprestimo, data_devolucao,
id_usuario, id_livro) VALUES
('2025-01-10', '2025-01-20', 1, 2),
('2025-01-15', '2025-01-25', 2, 1),
('2025-01-18', '2025-01-28', 3, 3),
('2025-01-20', '2025-01-30', 4, 5),
('2025-01-22', '2025-02-01', 5, 4);
```

Exercício 10: Consultas SQL

Consulta 1 – Listar todos os livros disponíveis:

```
SELECT * FROM Livros;
```

Consulta 2 – Consultar empréstimos de um determinado usuário:

```
SELECT e.id_emprestimo, e.data_emprestimo, e.data_devolucao,
l.titulo
FROM Emprestimos e
JOIN Livros l ON e.id_livro = l.id_livro
WHERE e.id_usuario = 1;
```

Consulta 3 – Verificar histórico de empréstimos e dados do usuário:

```
SELECT u.nome, u.email, e.data_emprestimo, e.data_devolucao,
l.titulo
FROM Emprestimos e
JOIN Usuarios u ON e.id_usuario = u.id_usuario
JOIN Livros l ON e.id_livro = l.id_livro;
```

Exercício 11: Regras de Negócio

- **Regra 1: Limite de Empréstimos por Usuário**
  Descrição: Um usuário não pode ter mais de 3 livros emprestados simultaneamente.
  Implementação: Pode ser controlada via aplicação ou com uma trigger que conte os empréstimos ativos.
- **Regra 2: Validação de Datas**
  Descrição: A data de devolução deve ser sempre posterior à data de empréstimo.
  Implementação: Restrições e validações na aplicação, ou uma CHECK constraint (se suportada pelo SGBD).
- **Regra 3: Atualização do Estoque**
  Descrição: Ao realizar um empréstimo, o estoque do livro deve ser decrementado; na devolução, incrementado.
  Implementação: Através de procedures ou triggers que atualizem o campo `estoque` na tabela `Livros`.

Exercício 12: Uso de Índices

- **Campos Sugeridos para Indexação:**
  - **Livros:**
    - `ISBN` (para buscas rápidas e garantia de unicidade)
    - `titulo` (melhora o desempenho em consultas por nome)
  - **Usuarios:**
    - `email` (otimiza pesquisas e validações de login)
  - **Emprestimos:**
    - `data_emprestimo` e `data_devolucao` (facilitam consultas baseadas em período)
- **Justificativa:**
  Índices auxiliam na melhoria do desempenho das consultas, sobretudo em colunas frequentemente utilizadas em cláusulas WHERE e JOIN.

Exercício 13: Backup e Restauração

- **Comandos (exemplo para PostgreSQL):**
  - **Backup:**
    `pg_dump -U usuario -h localhost nome_banco > backup_biblioteca.sql`
  - **Restauração:**
    `psql -U usuario -h localhost -d nome_banco -f backup_biblioteca.sql`
- **Importância:**
  Realizar backups periódicos garante a recuperação dos dados em caso de falhas, corrupção ou perda acidental.

Exercício 14: Testes de Integridade

- **Métodos de Teste:**

Verificação de Chaves Estrangeiras:
Executar consultas que realizem JOIN entre tabelas para confirmar que os relacionamentos estão corretos.
Exemplo:

```
SELECT e.id_emprestimo, u.nome, l.titulo
FROM Emprestimos e
JOIN Usuarios u ON e.id_usuario = u.id_usuario
JOIN Livros l ON e.id_livro = l.id_livro;
```

  - **Validações de Regras de Negócio:**
    Testar a lógica de empréstimo (ex.: tentar registrar um empréstimo que infrinja o limite de 3 livros).
  - **Utilização de Constraints:**
    Verificar, por exemplo, que não é possível inserir um registro em `Livros` sem um `ISBN`.

Exercício 15: Documentação do Banco de Dados

- **Elementos da Documentação:**
  - **Diagramas:**
    Incluir o diagrama entidade-relacionamento (ER) completo, ilustrando entidades e seus relacionamentos.
  - **Descrição das Tabelas:**
    Para cada tabela, listar os atributos, tipo de dado, restrições (chave primária, estrangeira, UNIQUE, NOT NULL) e comentários sobre o propósito.
  - **Dicionário de Dados:**
    Tabela detalhada com definição de cada campo, tamanho, domínio de valores e exemplos.
  - **Histórico de Ações:**
    Registro das principais alterações e decisões adotadas durante o desenvolvimento do projeto.

Exercício 16: Avaliação e Melhorias

- **Possíveis Melhorias:**
  - **Otimização de Consultas e Índices:**
    Revisar índices existentes e ajustar conforme o perfil de acesso para melhorar o desempenho.
  - **Implementação de Views e Procedures:**
    Criar views para facilitar a extração de relatórios e procedures para automatizar processos como atualização de estoque e envio de notificações.
  - **Aprimoramento da Segurança:**
    Incluir mecanismos de controle de acesso e auditoria para monitorar alterações e acessos ao banco de dados.

- **Justificativa:**
  Cada melhoria proposta visa não apenas otimizar a performance e a usabilidade do sistema, mas também garantir maior segurança, manutenção e escalabilidade do projeto.
