Centro Universitário de Brasília
Faculdade de Tecnologia e Ciências Sociais Aplicadas - FATECS

Projeto Final da Disciplina Banco de Dados II: Sistema de Banco de Dados Completo

Arthur Gabriel Moreira Clemente
João Gabriel Souto Guimarães de Souza

Brasília - DF
2025

Sumário

1. Informações Gerais

Tema do Projeto: Sistema de Biblioteca com empréstimos, devoluções e cálculo automático de multas.

Integrantes do Grupo: João Gabriel Souto Guimarães de Souza e Arthur Gabriel Moreira Clemente

Disciplina: Banco de Dados II

Professor: Leonardo Rodrigues de Deus

Banco Utilizado: PostgreSQL 17

2. Dicionário de Dados — Justificativas das Restrições

2.1 Tabela TB01_LEITOR

Armazena informações dos leitores cadastrados no sistema.

Justificativa das colunas e restrições

- **id_leitor: SERIAL, PK**
  Chave primária necessária para identificar de forma única cada leitor. O uso de SERIAL automatiza a geração do identificador.
- **nome: VARCHAR (100), NOT NULL**
  nome é obrigatório para fins de identificação e comunicação com o leitor. A restrição NOT NULL garante que nenhum leitor seja cadastrado sem nome.
- **email: VARCHAR (120), UNIQUE**
  email é opcional, mas caso informado deve ser único para evitar duplicidade de contas e permitir futuras funcionalidades como notificações.
- **status: VARCHAR (20), CHECK**
  status controla se o leitor está "**ativo**", "**suspenso**" ou "**bloqueado**".
  CHECK evita que valores inválidos sejam inseridos.
- **data_cadastro: DATE DEFAULT CURRENT_DATE**
  Registra a data de entrada no sistema de forma automática, útil para relatórios e auditorias.

2.2 Tabela TB02_AUTOR

Armazena os autores dos livros.

Justificativa das colunas e restrições

- **id_autor: SERIAL, PK**
  Identificação única de cada autor, gerada automaticamente.
- **nome_autor: VARCHAR (100), NOT NULL**

Necessário para identificar o autor. A restrição NOT NULL garante consistência e evita registros incompletos.

2.3 Tabela TB03_LIVRO

Cadastro dos livros disponíveis no acervo.

Justificativa das colunas e restrições

- **id_livro: SERIAL, PK**
  Identificador único do livro, gerado automaticamente.
- **titulo: VARCHAR (150), NOT NULL**
  título é obrigatório, pois identifica o livro. A ausência desse dado tornaria o acervo ilegível.
- **id_autor: INT, FK**
  Estrutura relacional obrigatória para vincular cada livro a um autor cadastrado.
  A foreign key garante integridade referencial.
- **ano_publicacao: INT, CHECK >=1500 AND <= ano atual**
  Evita dados absurdos (ex.: ano 0, ano futuro ou negativos).
  limite inferior considera o período em que a imprensa começou a se popularizar.
- **status_livro: VARCHAR (20), CHECK**
  Controla se o livro está "**disponível**", "**emprestado**", "**reservado**", etc.
  CHECK impede valores fora da lista pré-definida, protegendo regras do sistema.

2.4 Tabela TB04_EMPRESTIMO

Registra todos os empréstimos realizados.

Justificativa das colunas e restrições

- **id_emprestimo: SERIAL, PK**
  Identificação única para cada operação de empréstimo.
- **id_leitor: INT, FK**
  Garante que o empréstimo só ocorra para leitores válidos.
- **id_livro: INT, FK**
  Garante integridade: só livros cadastrados podem ser emprestados.
- **data_emprestimo: DATE**
  Necessária para cálculos de prazo e controle histórico.
- **data_prevista: DATE**
  Data limite para devolução; usada nas regras de multas.
- **data_devolucao: DATE**

Indica quando o item foi devolvido; utilizada em relatórios e regras de atraso.

2.5 Tabela TB05_MULTA

Armazena multas geradas após atraso na devolução.

Justificativa das colunas e restrições

- **id_multa: SERIAL, PK**
  Identificação única para cada multa
- **id_emprestimo: INT, FK**
  Liga a multa diretamente ao empréstimo específico, mantendo integridade do histórico.
- **valor: NUMERIC (10,2), CHECK (valor > 0)**
  A restrição CHECK impede valores negativos ou nulos, garantindo coerência financeira.
- **motivo: VARCHAR (200)**
  Campo para registrar explicações adicionais, como "**atraso na devolução**" ou "**livro danificado**".

data_registro: DATE

Indica quando a multa foi criada, útil para auditoria e relatórios mensais.

3. Justificativa da Lógica de Negócio

3.1 Função fn_calcular_multa_atraso(p_id_emprestimo INT)

O que faz

Calcula o valor da multa por atraso de um empréstimo identificado por **p_id_emprestimo**. A função obtém **data_prevista** e **data_devolucao** do registro de empréstimo. Se a devolução ainda não ocorreu, a função considera **CURRENT_DATE** como data de devolução para cálculo do atraso.

Assinatura

- Parâmetros: **p_id_emprestimo INT** — identificador do empréstimo.
- Retorno: **NUMERIC (10,2)** — valor da multa calculada (em reais). Retorna 0 se não houver atraso ou NULL se o empréstimo não existir.

Lógica principal e pontos de atenção

- Busca as colunas **data_prevista** e **data_devolucao** na tabela **tb04_emprestimo**.
- Se o empréstimo não existir, retorna NULL (sinalizando chamada inválida).
- Se **data_devolucao** for NULL, usa **CURRENT_DATE** para calcular o atraso até hoje.

- Calcula **v_dias_atraso** = **data_devolucao** - **data_prevista**.
- Se **v_dias_atraso** <= 0, retorna 0.
- Caso exista atraso, aplica regra do negócio: atualmente R$ 1,00 por dia **(v_multa := v_dias_atraso * 1.00)**.

Justificativa de Complexidade

A função centraliza a regra de cálculo de multas, permitindo reuso e manutenção simples caso a política (valor por dia, tolerância, escalonamento) mude. Não é trivial porque trata de ausência de devolução, validação de existência do empréstimo e conversões de data — e porque é consumida por triggers e procedimentos.

3.2 Procedure sp_registrar_emprestimo(p_id_leitor INT, p_id_livro INT, p_data_prevista DATE)

O que faz

Registra um novo empréstimo de forma transacional, realizando todas as validações necessárias antes de persistir qualquer alteração e garantindo que o sistema mantenha um estado consistente (atomicidade).

Assinatura

- Parâmetros:

  p_id_leitor INT — identificador do leitor que solicita o empréstimo.

  p_id_livro INT — identificador do livro a ser emprestado.

  p_data_prevista DATE — data prevista para devolução.

Passos executados

1. Busca status do leitor em **tb01_leitor**. Se o leitor não existir, lança exceção.
2. Verifica se o leitor está suspenso; se sim, lança exceção informando que não pode pegar livros.
3. Busca **status_livro em tb03_livro**. Se o livro não existir, lança exceção.
4. Verifica se o livro está Disponivel; se não estiver, lança exceção.
5. Insere um novo registro em **tb04_emprestimo** com data_emprestimo = **CURRENT_DATE** e a data_prevista informada.
6. Atualiza **tb03_livro.status_livro** para Emprestado.
7. Em caso de sucesso, emite um **RAISE NOTICE** confirmando o empréstimo.

Atomicidade / Transação

- A procedure é escrita em **PL/pgSQL** e conta com a transação implícita do bloco; qualquer erro gera EXCEPTION que interrompe a execução e faz o **ROLLBACK** automático do bloco, garantindo que nenhuma mudança parcial (ex.: inserir empréstimo sem atualizar status do livro) permaneça no banco.
- A seção EXCEPTION WHEN OTHERS THEN RAISE EXCEPTION 'Erro...: %', SQLERRM; unifica o tratamento de erros e evita que o banco finalize em estado inconsistente.

Justificativa de Complexidade

A procedure contém validações de existência, regras de negócio (bloqueio por status do leitor, disponibilidade do livro), alterações de múltiplas tabelas e controle explícito de erros — vai além de um simples **INSERT** e portanto atende ao requisito de complexidade não-trivial.

3.3 Trigger trg_registrar_multa + função disparadora fn_trg_registrar_multa

Comportamento implementado

- Trigger **trg_registrar_multa** é definido como **AFTER UPDATE** em **tb04_emprestimo** e **chama fn_trg_registrar_multa()** para cada linha (FOR EACH ROW).
- A função **trigger** verifica se houve mudança de **data_devolucao** de NULL para um valor não nulo — ou seja, detecta o momento em que o empréstimo é efetivamente devolvido.
- Ao detectar devolução, chama fn_calcular_multa_atraso(**NEW.id_emprestimo**).
    - Se a multa calculada for maior que 0, insere um registro em **tb05_multa** com **id_emprestimo**, valor e motivo.
- Atualiza **tb03_livro.status_livro** para Disponivel, liberando o livro para novos empréstimos.
- Retorna NEW.

Por que AFTER e não BEFORE?

- O cálculo da multa depende do valor final de data_devolucao (e possivelmente de outras colunas atualizadas). Usando **AFTER UPDATE**, temos a garantia de que o

NEW já foi validado/aplicado e que os dados persistidos correspondem ao estado final do registro.

- Além disso, operações que geram efeitos em outras tabelas (inserção de multa, atualização do status do livro) fazem sentido ocorrer somente após a confirmação da atualização do empréstimo.

Justificativa funcional

- O trigger automatiza tarefas administrativas que, se feitas manualmente, estariam sujeitas a erro humano (esquecer de criar multa, não liberar o livro, etc.).
- Mantém consistência entre histórico de empréstimos, multas e disponibilidade do acervo.
- Evita duplicação de lógica, delegando o cálculo de multa para **fn_calcular_multa_atraso()**.

4. Justificativa de Segurança

4.1 Roles e usuários criados

- **admins** (role, NOLOGIN) — grupo com privilégios administrativos.
- **professor** (role, NOLOGIN) — grupo com privilégios restritos para uso acadêmico.
- **Usuários** com LOGIN:
    - **gabriel** (LOGIN) — membro do grupo admins. **Senha** no script: '**gabriel123**'.
    - **artur** (LOGIN) — membro do grupo admins. **Senha** no script: **'artur123'**.
    - **professor_1** (LOGIN) — membro do grupo professor. **Senha** no script: '**123456**'.

4.2 Permissões atribuídas e justificativas

Permissões do grupo admins

- **GRANT ALL PRIVILEGES** no schema **biblioteca** e no public (conforme script).
- Permissões totais em todas as tabelas e sequences.
- **EXECUTE** em procedures e functions.

Justificativa: administradores precisam gerir estrutura e dados do BD (criar/alterar tabelas, ajustar schemas, recuperar dados, executar scripts de manutenção). Esse perfil é reservado a responsáveis pela infraestrutura e manutenção.

Permissões do grupo professor

- **Bloqueios**: **REVOKE CREATE** no schema **biblioteca** e **public**; **REVOKE ALL** no schema **biblioteca**.
- **Concessões:** GRANT USAGE ON SCHEMA biblioteca TO professor; SELECT em todas as tabelas; INSERT, UPDATE nas tabelas principais (**tb01_leitor, tb02_autor**, **tb03_livro, tb04_emprestimo**, **tb05_multa**); **GRANT EXECUTE** em todas as funções e procedures do schema; **REVOKE DELETE** explicitamente.
- **Senha do professor: '123456'** (presente no script para **professor_1**).

Justificativa detalhada:

- **Sem CREATE/ALTER/DROP:** impede que o professor altere a estrutura do banco (evita perda de integridade, criação de artefatos não autorizados ou desligamento de regras de negócio). Em ambientes acadêmicos é comum separar ambiente de desenvolvimento/produção; professor não deve alterar a modelagem central do projeto.
- **SELECT amplo:** permite que o professor visualize dados para avaliação, correção e análise sem risco de modificação estrutural.
- **INSERT e UPDATE liberados:** permitem atividades pedagógicas (incluir registros de teste, atualizar status, demonstrar uso), sem comprometer históricos críticos.
- **Sem DELETE: DELETE** é perigoso porque remove histórico (ex.: empréstimos e multas) que pode ser necessário para avaliação e auditoria. Rejeitar DELETE protege o histórico e evita perda acidental de dados.
- **EXECUTE em funções/procedures:** permite que o professor utilize as rotinas de negócio (testar procedimentos, simular empréstimos) sem necessidade de permissões elevadas.

5. Conclusão

O sistema implementado atende aos requisitos do projeto e da disciplina, oferecendo:

- Modelo de dados consistente com restrições adequadas **(PK, FK, UNIQUE, CHECK)** que preservam a integridade referencial e a qualidade dos dados.
- Lógica de negócio significativa e **não-trivial**, centralizada em **função**, **procedure** e **trigger**, que realizam **validações**, garantem atomicidade das operações e automatizam **regras (cálculo de multa, geração automática de multas, liberação do acervo)**.
- Política de segurança coerente com cenários acadêmicos: separação clara entre administradores **(controle total**) e professores **(acesso controlado para avaliação e uso, sem risco de alteração estrutural ou exclusão de dados)**.
- Facilidade de manutenção: funções reutilizáveis **(ex.: fn_calcular_multa_atraso)** reduzem duplicação e facilitam alteração de regras de negócio no futuro.