Ministério da Educação
Universidade Tecnológica Federal do Paraná
Câmpus Pato Branco
Bacharelado em Engenharia da Computação

TRABALHO DE BANCO DE DADOS II

Atividade Prática Supervisionada

Relatório técnico

Kevin Azevedo Lopes [1]
Marcos Henrique Gomes Vieira Saito [2]
Mariana da Costa Zatta [3]

PATO BRANCO
JUNHO/2024

[1] kevinlopes@alunos.utfpr.edu.br
[2] marcossaito@alunos.utfpr.edu.br
[3] marianazatta@alunos.utfpr.edu.br

SUMÁRIO

1. INTRODUÇÃO

O presente relatório técnico tem como objetivo documentar o desenvolvimento de um sistema de gerenciamento de biblioteca utilizando PostgreSQL. Este sistema envolve a modelagem de um banco de dados que gerencia livros, usuários, funcionários, empréstimos e reservas. As etapas do projeto incluem a definição da aplicação, a elaboração do diagrama entidade-relacionamento (ER), o mapeamento para o modelo relacional, a geração de scripts de criação e povoamento de tabelas, a criação de índices, funções, visões e triggers.

2. APLICAÇÃO ESCOLHIDA

A aplicação escolhida para este projeto é um Sistema de Gestão de Biblioteca, similar ao que é utilizado na UTFPR. Este sistema é responsável por gerenciar o cadastro de livros, usuários, funcionários, empréstimos, devoluções e reservas. Cada livro possui informações como título, autor, ISBN e categoria. Os usuários têm dados como nome, endereço e data de nascimento. Os funcionários possuem dados pessoais e detalhes de emprego. O sistema permite que usuários façam empréstimos e reservas de livros, enquanto os funcionários gerenciam esses processos.

3. DIAGRAMA ENTIDADE-RELACIONAMENTO

As principais entidades do diagrama são:

Livro: Representa os livros da biblioteca.

Usuário: Representa os usuários que podem emprestar e reservar livros.

Funcionário: Representa os funcionários que gerenciam empréstimos e reservas.

Empréstimo: Representa os empréstimos de livros realizados por usuários.

Reserva: Representa as reservas de livros realizadas por usuários.

3.1. Entidades e Atributos

Livro

- livro_id (PK)
- título
- autor
- isbn
- categoria

Usuário

- usuario_id (PK)
- nome
- endereço
- data_nascimento
- telefone

Funcionário

- funcionario_id (PK)
- nome
- cargo

Empréstimo

- emprestimo_id (PK)
- usuario_id (FK)
- livro_id (FK)
- funcionario_id (FK)
- data_emprestimo
- data_devolucao

Reserva

- reserva_id (PK)
- usuario_id (FK)
- livro_id (FK)
- funcionario_id (FK)
- data_reserva

3.2. Relacionamentos

Usuário-Emprestimo: Um usuário pode fazer vários empréstimos. Cada empréstimo está relacionado a um único usuário.

Livro-Emprestimo: Um livro pode ser emprestado várias vezes. Cada empréstimo está relacionado a um único livro.

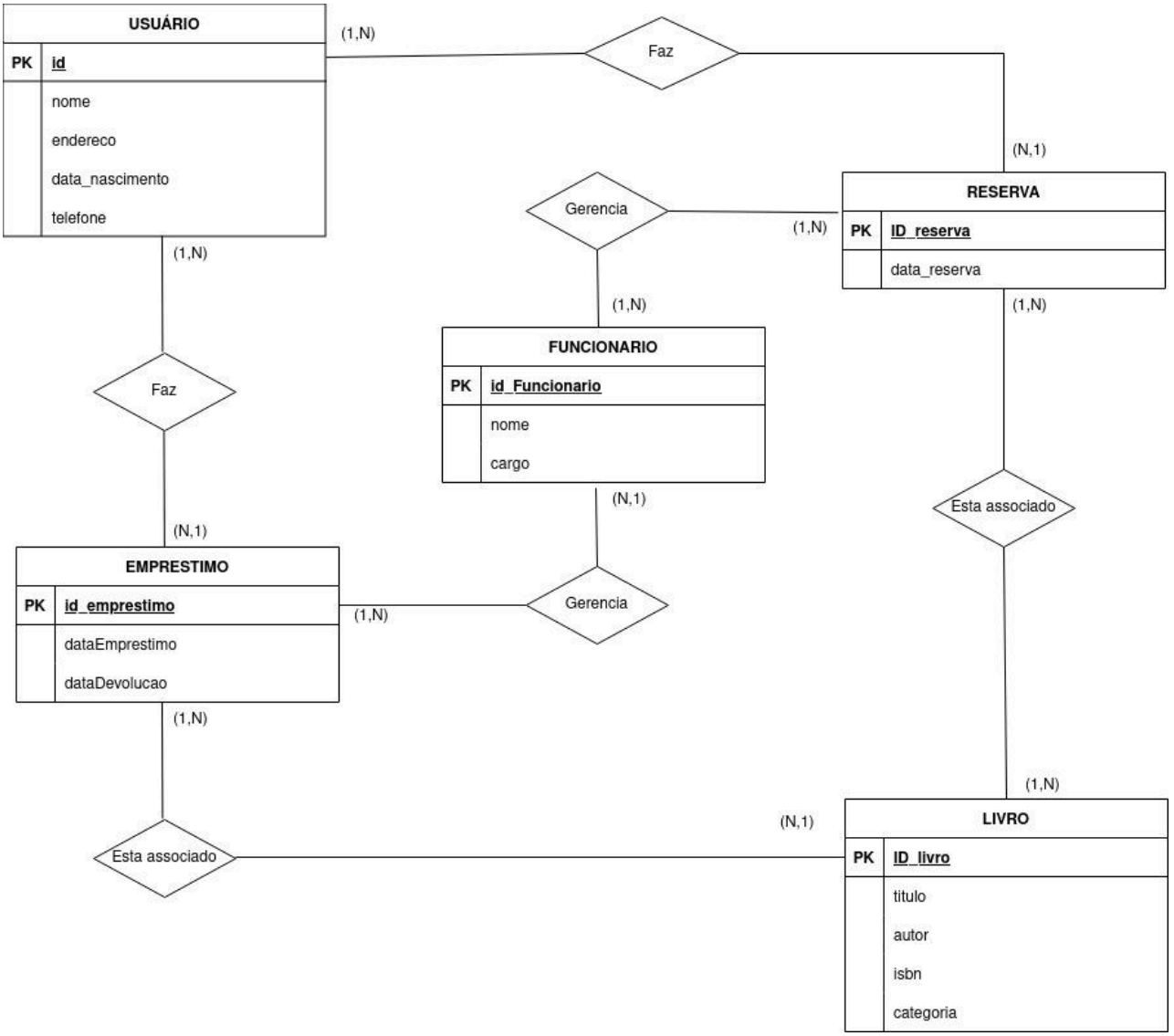
Funcionário-Emprestimo: Um funcionário pode gerenciar vários empréstimos. Cada empréstimo está relacionado a um único funcionário.

Usuário-Reserva: Um usuário pode fazer várias reservas. Cada reserva está relacionada a um único usuário.

Livro-Reserva: Um livro pode ser reservado várias vezes. Cada reserva está relacionada a um único livro.

Funcionário-Reserva: Um funcionário pode gerenciar várias reservas. Cada reserva está relacionada a um único funcionário.

3.3. Diagrama


USUÁRIO
PK
id
nome
endereco
data_nascimento
telefone
(1,N)
Faz
(N,1)
RESERVA
PK
ID_reserva
data_reserva
Gerencia
(1,N)
(1,N)
(1,N)
FUNCIONARIO
PK
id_Funcionario
nome
cargo
Faz
(N,1)
(N,1)
Esta associado
EMPRESTIMO
PK
id_emprestimo
dataEmprestimo
dataDevolucao
(1,N)
Gerencia
(1,N)
(1,N)
(N,1)
Esta associado
(1,N)
LIVRO
PK
ID_livro
titulo
autor
isbn
categoria

4. MAPEAMENTO PARA O MODELO RELACIONAL

O diagrama ER foi mapeado para o modelo relacional com as seguintes tabelas e restrições de integridade:

Livro

- **livro_id** SERIAL PRIMARY KEY
- **titulo** VARCHAR(255) NOT NULL
- **autor** VARCHAR(255)
- **isbn** VARCHAR(13)
- **categoria** VARCHAR(100)

Usuário

- **usuario_id** SERIAL PRIMARY KEY
- **nome** VARCHAR(255) NOT NULL
- **endereco** VARCHAR(255)
- **data_nascimento** DATE
- **telefone** VARCHAR(255)

Funcionário

- **funcionario_id** SERIAL PRIMARY KEY
- **nome** VARCHAR(255) NOT NULL
- **cargo** VARCHAR(100)

Empréstimo

- **emprestimo_id** SERIAL PRIMARY KEY
- **usuario_id** INT REFERENCES Usuario(usuario_id)
- **livro_id** INT REFERENCES Livro(livro_id)
- **funcionario_id** INT REFERENCES Funcionario(funcionario_id)
- **data_emprestimo** DATE NOT NULL
- **data_devolucao** DATE

Reserva

- **reserva_id** SERIAL PRIMARY KEY
- **usuario_id** INT REFERENCES Usuario(usuario_id)
- **livro_id** INT REFERENCES Livro(livro_id)
- **funcionario_id** INT REFERENCES Funcionario(funcionario_id)

data_reserva DATE NOT NULL

5. SCRIPTS

Abaixo estão os scripts de criação de tabelas no PostgreSQL com as restrições de integridade:

```
Unset
--1. Script de Criação de Tabelas com Restrições
-- Criação da tabela Livro
      CREATE TABLE Livro (
          livro_id SERIAL PRIMARY KEY,
          titulo VARCHAR(255) NOT NULL,
          autor VARCHAR(255),
          isbn VARCHAR(13) UNIQUE,  -- ISBN deve ser único
            categoria VARCHAR(100) NOT NULL CHECK (categoria IN ('Ficção',
'Romance', 'Não Ficção'))  -- Categoria restrita a valores específicos
      );

      -- Criação da tabela Usuario
      CREATE TABLE Usuario (
          usuario_id SERIAL PRIMARY KEY,
          nome VARCHAR(255) NOT NULL,
          endereco VARCHAR(255),
          data_nascimento DATE,
          telefone VARCHAR(20) UNIQUE  -- Telefone deve ser único
      );

      -- Criação da tabela Funcionario
      CREATE TABLE Funcionario (
          funcionario_id SERIAL PRIMARY KEY,
          nome VARCHAR(255) NOT NULL,
          cargo VARCHAR(100) NOT NULL
      );

      -- Criação da tabela Emprestimo
      CREATE TABLE Emprestimo (
          emprestimo_id SERIAL PRIMARY KEY,
```

```
            usuario_id INT REFERENCES Usuario(usuario_id) ON DELETE CASCADE,
-- Chave estrangeira para Usuario com exclusão em cascata
            livro_id INT REFERENCES Livro(livro_id) ON DELETE CASCADE,
            funcionario_id INT REFERENCES Funcionario(funcionario_id),
            data_emprestimo DATE NOT NULL,
            data_devolucao DATE,
             CONSTRAINT data_devolucao_check CHECK (data_devolucao IS NULL OR
data_devolucao >= data_emprestimo)  -- Data de devolução deve ser maior ou
igual à data de empréstimo, se especificada
        );

-- Criação da tabela Reserva
        CREATE TABLE Reserva (
            reserva_id SERIAL PRIMARY KEY,
            usuario_id INT REFERENCES Usuario(usuario_id) ON DELETE CASCADE,
            livro_id INT REFERENCES Livro(livro_id) ON DELETE CASCADE,
            funcionario_id INT REFERENCES Funcionario(funcionario_id),
            data_reserva DATE NOT NULL
        );
```

6. CRIAÇÃO DE ÍNDICES

Criação de índices para otimização de consultas:

```
Unset
        -- 2. Criação de índices
        CREATE INDEX idx_usuario_nome ON Usuario(nome);  -- Índice no nome dos
usuários
        CREATE INDEX idx_livro_categoria ON Livro(categoria);  -- Índice na
categoria dos livros
        CREATE INDEX idx_livro_titulo ON Livro(titulo);  -- Índice no título
dos livros
```

7. POPULAÇÃO DAS TABELAS

Unset

```
-- 3. População das tabelas

-- Inserção de dados na tabela Livro
INSERT INTO Livro (titulo, autor, isbn, categoria) VALUES
('Dom Casmurro', 'Machado de Assis', '9788535914848', 'Ficção'),
('Memórias Póstumas de Brás Cubas', 'Machado de Assis', '9788535914855',
'Ficção'),
('O Guarani', 'José de Alencar', '9788520934012', 'Romance'),
('Iracema', 'José de Alencar', '9788520934029', 'Romance'),
('Capitães da Areia', 'Jorge Amado', '9788520935019', 'Ficção'),
('Grande Sertão: Veredas', 'João Guimarães Rosa', '9788520935026', 'Ficção'),
('A Moreninha', 'Joaquim Manuel de Macedo', '9788520935033', 'Romance'),
('Senhora', 'José de Alencar', '9788520935040', 'Romance');

-- Inserção de dados na tabela Usuario
INSERT INTO Usuario (nome, endereco, data_nascimento) VALUES
('Alice Silva', 'Rua das Flores, 123', '2000-01-15'),
('Marco Santos', 'Avenida das Acácias, 456', '1998-05-22'),
('Carlos Oliveira', 'Rua das Palmeiras, 789', '2004-03-30'),
('Daniela Almeida', 'Avenida Central, 101', '1999-12-11');

-- Inserção de dados na tabela Funcionario
INSERT INTO Funcionario (nome, cargo) VALUES
('João Pereira', 'Bibliotecário'),
('Maria Oliveira', 'Assistente de Biblioteca'),
('Pedro Lima', 'Gerente de Biblioteca'),
('Ana Souza', 'Atendente');
```

8. FUNÇÕES

Funções que realizam consultas comuns, como obter livros por autor, ver total de empréstimos por usuário e também quais títulos foram pegos por cada usuário.

```
Unset
	--4. Criação de Funções

	-- Função para obter livros por autor
	CREATE FUNCTION obter_livros_por_autor(nome_autor VARCHAR) RETURNS TABLE(livroid INT, titulo_livro VARCHAR) AS $$
	BEGIN
		RETURN QUERY SELECT livro_id, titulo FROM Livro WHERE autor = nome_autor;
	END;
	$$ LANGUAGE plpgsql;

	drop function obter_livros_por_autor(nome_autor VARCHAR);

	-- Função para contar o total de empréstimos por usuário
	CREATE FUNCTION total_emprestimos_por_usuario(usuarioid INT) RETURNS INT AS $$
	BEGIN
		RETURN (SELECT COUNT(*) FROM Emprestimo WHERE usuario_id = usuario_id);
	END;
	$$ LANGUAGE plpgsql;

	drop function total_emprestimos_por_usuario(usuarioid INT);

	-- Função para listar livros emprestados por usuário
	CREATE FUNCTION livros_emprestados_por_usuario(usuarioid INT) RETURNS TABLE(titulo VARCHAR, data_emprestimo DATE, data_devolucao DATE) AS $$
	BEGIN
	    RETURN QUERY
	    SELECT l.titulo, e.data_emprestimo, e.data_devolucao
	    FROM Emprestimo e
```

```
        JOIN Livro l ON e.livro_id = l.livro_id
        WHERE e.usuario_id = usuarioid;
    END;
    $$ LANGUAGE plpgsql;

    drop function livros_emprestados_por_usuario(usuarioid INT);
```

9. VISÕES E TABELA DE AUDITORIA

Foram também feitas as criações de visões e uma tabela de auditoria:

```
Unset
    --5. Criação de VIEW

    -- Visão para listar livros emprestados
    CREATE VIEW livros_emprestados AS
    SELECT u.nome AS usuario, l.titulo, e.data_emprestimo,
e.data_devolucao
    FROM Emprestimo e
    JOIN Usuario u ON e.usuario_id = u.usuario_id
    JOIN Livro l ON e.livro_id = l.livro_id;

    -- Visão para listar reservas ativas
    CREATE VIEW reservas_ativas AS
    SELECT u.nome AS usuario, l.titulo, r.data_reserva
    FROM Reserva r
    JOIN Usuario u ON r.usuario_id = u.usuario_id
    JOIN Livro l ON r.livro_id = l.livro_id
    WHERE r.data_reserva >= CURRENT_DATE;

    -- Visão para listar livros disponíveis
    CREATE VIEW livros_disponiveis AS
    SELECT l.titulo, l.autor, l.categoria
    FROM Livro l
```

```
        LEFT JOIN Emprestimo e ON l.livro_id = e.livro_id AND e.data_devolucao
IS NULL
        WHERE e.emprestimo_id IS NULL;

        -- Criação da tabela de auditoria para a tabela Emprestimo
        CREATE TABLE Auditoria_Emprestimo (
            auditoria_id SERIAL PRIMARY KEY,
             emprestimo_id INT REFERENCES Emprestimo(emprestimo_id) ON DELETE
CASCADE,
             operacao CHAR(1) CHECK (operacao IN ('I', 'U', 'D')),  -- Operação
deve ser I (Inserir), U (Atualizar) ou D (Deletar)
            data_operacao TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
            funcionario_id INT REFERENCES Funcionario(funcionario_id)
        );
```

10. TRIGGERS

Criação de triggers para tratar eventos específicos

```
Unset
        -- 6. Criação de Triggers para Tratamento de Eventos e Auditoria

        -- Função para log de inserção na tabela Emprestimo
        CREATE FUNCTION log_inserir_emprestimo() RETURNS TRIGGER AS $$
        BEGIN
                INSERT INTO Auditoria_Emprestimo (emprestimo_id, operacao,
data_operacao, funcionario_id)
                    VALUES (NEW.emprestimo_id, 'I', CURRENT_TIMESTAMP,
NEW.funcionario_id);
            RETURN NEW;
        END;
        $$ LANGUAGE plpgsql;

        -- Trigger para logar inserções na tabela Emprestimo
        CREATE TRIGGER trigger_inserir_emprestimo
```

```
        AFTER INSERT ON Emprestimo
        FOR EACH ROW
        EXECUTE FUNCTION log_inserir_emprestimo();

        -- Função para log de atualização na tabela Emprestimo
        CREATE FUNCTION log_atualizar_emprestimo() RETURNS TRIGGER AS $$
        BEGIN
                INSERT INTO Auditoria_Emprestimo (emprestimo_id, operacao,
data_operacao, funcionario_id)
                    VALUES (NEW.emprestimo_id, 'U', CURRENT_TIMESTAMP,
NEW.funcionario_id);
            RETURN NEW;
        END;
        $$ LANGUAGE plpgsql;

        -- Trigger para logar atualizações na tabela Emprestimo
        CREATE TRIGGER trigger_atualizar_emprestimo
        AFTER UPDATE ON Emprestimo
        FOR EACH ROW
        EXECUTE FUNCTION log_atualizar_emprestimo();

        -- Função para log de deleção na tabela Emprestimo
        CREATE FUNCTION log_deletar_emprestimo() RETURNS TRIGGER AS $$
        BEGIN
                INSERT INTO Auditoria_Emprestimo (emprestimo_id, operacao,
data_operacao, funcionario_id)
                    VALUES (OLD.emprestimo_id, 'D', CURRENT_TIMESTAMP,
OLD.funcionario_id);
            RETURN OLD;
        END;
        $$ LANGUAGE plpgsql;

        -- Trigger para logar deleções na tabela Emprestimo
        CREATE TRIGGER trigger_deletar_emprestimo
        AFTER DELETE ON Emprestimo
        FOR EACH ROW
```

```
EXECUTE FUNCTION log_deletar_emprestimo();
```

11. SCRIPT PARA TESTES

Script utilizado para a realização dos testes:

```
Unset
-- APS Kevin Azevedo Lopes,  Mariana da Costa Zatta, Marcos Henrique Gomes Vieira Saito

-- Seleciona todos os registros da tabela Livro
SELECT * FROM Livro;

-- Seleciona todos os registros da tabela Usuario
SELECT * FROM Usuario;

-- Seleciona todos os registros da tabela Funcionario
SELECT * FROM Funcionario;

-- Inserção de dados na tabela Emprestimo
-- Cada linha representa um empréstimo de um livro por um usuário com a ajuda de um funcionário
INSERT INTO Emprestimo (usuario_id, livro_id, funcionario_id, data_emprestimo, data_devolucao) VALUES
(1, 1, 1, '2024-06-01', '2024-06-15'),  -- Alice Silva emprestou Dom Casmurro com o funcionário João Pereira
(2, 2, 2, '2024-06-02', '2024-06-16'),  -- Marco Santos emprestou Memórias Póstumas de Brás Cubas com a funcionária Maria Oliveira
(3, 3, 3, '2024-06-03', NULL),  -- Carlos Oliveira emprestou O Guarani com o funcionário Pedro Lima (ainda não devolvido)
(4, 4, 4, '2024-06-04', '2024-06-18'),  -- Daniela Almeida emprestou Iracema com a funcionária Ana Souza
```

```
    (1, 5, 1, '2024-06-05', NULL),  -- Alice Silva emprestou Capitães da Areia com o funcionário João Pereira (ainda não devolvido)
    (2, 6, 2, '2024-06-06', NULL),  -- Marco Santos emprestou Grande Sertão: Veredas com a funcionária Maria Oliveira (ainda não devolvido)
    (3, 7, 3, '2024-06-07', '2024-06-21'),  -- Carlos Oliveira emprestou A Moreninha com o funcionário Pedro Lima
    (4, 8, 4, '2024-06-08', '2024-06-22');  -- Daniela Almeida emprestou Senhora com a funcionária Ana Souza

    -- Inserção de dados na tabela Reserva
    -- Cada linha representa uma reserva de um livro por um usuário com a ajuda de um funcionário
    INSERT INTO Reserva (usuario_id, livro_id, funcionario_id, data_reserva) VALUES
    (1, 2, 2, '2024-07-01'),  -- Alice Silva reservou Memórias Póstumas de Brás Cubas com a funcionária Maria Oliveira
    (2, 3, 3, '2024-07-02'),  -- Marco Santos reservou O Guarani com o funcionário Pedro Lima
    (3, 4, 4, '2024-07-03'),  -- Carlos Oliveira reservou Iracema com a funcionária Ana Souza
    (4, 5, 1, '2024-07-04'),  -- Daniela Almeida reservou Capitães da Areia com o funcionário João Pereira
    (1, 6, 2, '2024-07-05'),  -- Alice Silva reservou Grande Sertão: Veredas com a funcionária Maria Oliveira
    (2, 7, 3, '2024-07-06'),  -- Marco Santos reservou A Moreninha com o funcionário Pedro Lima
    (3, 8, 4, '2024-07-07'),  -- Carlos Oliveira reservou Senhora com a funcionária Ana Souza
    (4, 1, 1, '2024-07-08');  -- Daniela Almeida reservou Dom Casmurro com o funcionário João Pereira

    -- Seleciona todos os registros da tabela Emprestimo
    SELECT * FROM Emprestimo;

    -- Seleciona todos os registros da tabela Reserva
    SELECT * FROM Reserva;
```

```
-- Seleciona todos os registros da tabela Auditoria_Emprestimo
SELECT * FROM Auditoria_Emprestimo;

-- Verifica o plano de execução da consulta para verificar se o índice está sendo usado
EXPLAIN ANALYZE SELECT * FROM Usuario WHERE nome = 'Alice Silva';

-- Verifica o plano de execução da consulta para verificar se o índice está sendo usado
EXPLAIN ANALYZE SELECT * FROM Livro WHERE categoria = 'Ficção';

-- Verifica o plano de execução da consulta para verificar se o índice está sendo usado
EXPLAIN ANALYZE SELECT * FROM Livro WHERE titulo = 'Dom Casmurro';

-- Chama a função obter_livros_por_autor para obter todos os livros de um determinado autor
SELECT * FROM obter_livros_por_autor('Machado de Assis');

-- Chama a função total_emprestimos_por_usuario para obter o total de empréstimos de um usuário
SELECT total_emprestimos_por_usuario(1);

-- Chama a função livros_emprestados_por_usuario para obter todos os livros emprestados por um usuário
SELECT * FROM livros_emprestados_por_usuario(1);

-- Seleciona todos os registros da visão livros_emprestados
SELECT * FROM livros_emprestados;

-- Seleciona todos os registros da visão reservas_ativas
SELECT * FROM reservas_ativas;

-- Seleciona todos os registros da visão livros_disponiveis
SELECT * FROM livros_disponiveis;
```

```
	-- Atualiza a data de devolução de um empréstimo específico e verifica
a tabela de auditoria
	UPDATE Emprestimo SET data_devolucao = '2024-07-10' WHERE
emprestimo_id = 1;

	SELECT * FROM auditoria_emprestimo;

	-- Seleciona todos os registros da tabela Auditoria_Emprestimo para
verificar o log após a atualização
	SELECT * FROM Auditoria_Emprestimo;

	-- Deleta um empréstimo específico e verifica a tabela de auditoria
	DELETE FROM Emprestimo WHERE emprestimo_id = 1;

	-- Seleciona todos os registros da tabela Auditoria_Emprestimo para
verificar o log após a exclusão
	SELECT * FROM Auditoria_Emprestimo;
```

11.1. Seleção de Registros

Consultas foram feitas para selecionar todos os registros das tabelas Livro, Usuario, e Funcionario para verificar a correta inserção de dados.

11.2. Inserção de Dados:

Foram inseridos dados na tabela Emprestimo representando empréstimos de livros por usuários, com a ajuda de funcionários.
Inserção de dados na tabela Reserva para representar reservas de livros pelos usuários.

11.3. Verificação de Dados:

Consultas foram executadas para selecionar todos os registros das tabelas Emprestimo e Reserva para garantir que os dados foram inseridos corretamente.

11.4. Auditoria e Índices:

Consultas EXPLAIN ANALYZE foram utilizadas para verificar o plano de execução das consultas e garantir que os índices foram utilizados eficientemente.

11.5. Funções e Visões:

Foram criadas e testadas funções para consultas corriqueiras, como obter livros por autor, total de empréstimos por usuário, e livros emprestados por usuário.
Visões foram criadas para facilitar o acesso a informações específicas como livros emprestados, reservas ativas e livros disponíveis.

11.6. Auditoria de Transações:

Uma tabela de auditoria (Auditoria_Emprestimo) foi implementada com triggers para registrar operações de inserção, atualização e exclusão na tabela Empréstimo.

11.7. Atualização e Exclusão de Dados:

Foram realizadas operações de atualização e exclusão de registros na tabela Empréstimo, seguidas da verificação dos registros na tabela de auditoria para garantir a integridade das operações.

12. CONSIDERAÇÕES FINAIS

O objetivo deste trabalho foi criar um sistema de gerenciamento de biblioteca utilizando PostgreSQL, documentando todas as etapas do desenvolvimento. O sistema proposto atende aos requisitos especificados e demonstra a aplicação prática de conceitos de modelagem de dados, criação de índices, funções, visões e triggers. Durante o projeto encontramos erros e desafios para serem superados, e para o resultado final foram necessárias correções para que ele atendesse as expectativas e as especificações.

O trabalho atingiu seus objetivos e proporcionou uma experiência no desenvolvimento de um sistema de gerenciamento de dados

13. REFERÊNCIAS

Moodle da disciplina BD27CP - Banco de Dados II. Lecionada pelo professor Ives Renê Venturini Pola. Acesso: https://moodle.utfpr.edu.br/course/view.php?id=14182#section-1