Projeto de Banco de Dados - Sistema de Bibliotecas

Discente: Allan Silva Borges

Docente: Robespierre Pita

Disciplina: MATA60 - Banco de Dados

Introdução ao Projeto:

Este projeto tem como objetivo registrar o processo de migração de um banco de dados PostgreSQL feito no pgadmin para o DuckDB utilizando o DBeaver, explorando as diferenças nos códigos de ambos os sistemas de gerenciamento de banco de dados. Durante a migração, serão comparadas as sintaxes e as abordagens utilizadas em consultas SQL. Através deste relatório, será possível observar como as mudanças impactam a estrutura das consultas, proporcionando uma visão clara das vantagens e desafios envolvidos nesse processo de transição.

1. Códigos para a criação e população das tabelas:

link github: https://github.com/of-allan-borges/Projeto2--Banco-de-dados

2. Criando Indexação:

2.1. Índice em titulo: CREATE INDEX idx_livro_titulo ON livro(titulo);

- **Buscas por título serão bastante comuns na biblioteca porque quando o usuário for pegar um livro emprestado, ele vai identificar o livro pelo título, então assim o funcionário vai procurar a disponibilidade deste livro através do seu título.**
- **Exemplo de consulta: SELECT titulo, editora, quantidade_disponivel FROM livro WHERE titulo = 'Livro 1';**

2.2. Índice em id_local na tabela livro: CREATE INDEX idx_livro_local ON livro(id_local);

2.2.1. Índice em id_local e id_biblioteca na tabela local: CREATE INDEX idx_local_biblioteca ON local(id_local, fk_local_biblioteca);

- O primeiro índice ajuda a acelerar a busca dos locais dos livros, o segundo ajuda a otimizar o JOIN entre livro e local, além de agilizar a pesquisa pela biblioteca associada a um local. Os funcionários precisam pesquisar o local do livro sempre que um usuário solicitar o empréstimo de algum livro, pois os livros estão associados a um local específico e cada local pertence a uma das três bibliotecas. Então esses índices vão agilizar esse processo.
- Exemplo de consulta:

```
SELECT livro.titulo, local.andar, local.prateleira,
biblioteca.nome_biblioteca

FROM livro

JOIN local ON livro.id_local = local.id_local

JOIN biblioteca ON local.fk_local_biblioteca = biblioteca.id_biblioteca

WHERE livro.id_livro = 2;
```

2.3. Índice para Busca de Livros com Base em Quantidade Total e Quantidade Disponível: CREATE INDEX idx_livro_quantidades ON livro(quantidade_total, quantidade_disponivel);

- Essa consulta será feita frequentemente para verificar se há livros disponíveis para empréstimo com base na quantidade total e controlar o estoque de livros.
- Exemplo de consulta: SELECT titulo, quantidade_total, quantidade_disponivel FROM livro WHERE quantidade_disponivel > 0;

3. Materialize View

1. View para Empréstimos e Reservas de um Usuário

```
CREATE VIEW mv_emprestimos_reservas_usuario AS
SELECT
    u.id_usuario AS usuario_id,
    u.nome AS usuario_nome,
    l.titulo AS livro_titulo,
    e.data_emprestimo AS data_emprestimo,
    e.data_devolucao AS data_devolucao,
    e.status AS status_emprestimo,
    r.data_reserva AS data_reserva,
    r.status_reserva AS status_reserva,
    r.data_validade AS data_validade_reserva
FROM
    usuario u
JOIN emprestimo e ON u.id_usuario = e.id_usuario
JOIN livro l ON e.id_livro = l.id_livro
LEFT JOIN reserva r ON u.id_usuario = r.id_usuario AND l.id_livro = r.id_livro;
```

-- Consulta exemplo para testar a view de empréstimos e reservas de um usuário

```
SELECT * FROM mv_emprestimos_reservas_usuario LIMIT 10; -- Limita o resultado a 10 linhas para uma visualização rápida
```

2. View para Livros Disponíveis por Biblioteca

```
CREATE VIEW mv_livros_disponiveis_por_biblioteca AS
SELECT
    b.id_biblioteca AS biblioteca_id,
```

```
    b.nome_biblioteca AS biblioteca_nome,
    l.descricao AS local_descricao,
    l.andar AS local_andar,
    l.prateleira AS local_prateleira,
    l.tipo_local AS tipo_local,
    l.capacidade AS capacidade_local,
    livro.titulo AS livro_titulo,
    livro.quantidade_disponivel AS quantidade_disponivel
FROM
    biblioteca b
JOIN local l ON b.id_biblioteca = l.fk_local_biblioteca
JOIN livro ON l.id_local = livro.id_local
WHERE
    livro.quantidade_disponivel > 0;
-- Consulta exemplo para testar a view de livros disponíveis por biblioteca
SELECT *
FROM mv_livros_disponiveis_por_biblioteca
WHERE biblioteca_id = 1
LIMIT 10;  -- Limita o resultado a 10 linhas
```

3. View para Multas Aplicadas por Usuário

```
CREATE VIEW mv_multas_por_usuario AS
SELECT
    u.id_usuario AS usuario_id,
    u.nome AS usuario_nome,
    m.valor AS valor_multa,
```

```
        m.status_multa AS status_multa,
        m.descricao AS descricao_multa,
        m.data_multa AS data_multa,
        m.data_pagamento AS data_pagamento
FROM
        multa m
JOIN usuario u ON m.id_usuario = u.id_usuario;
-- Consulta exemplo para testar a view de multas aplicadas por usuário
SELECT *
FROM mv_multas_por_usuario
WHERE usuario_id = 1  -- Substitua por um ID válido de usuário para filtrar
LIMIT 10;  -- Limita o resultado a 10 linhas
```

4. Perguntas Analiticas

Descrição da Query	Pergunta Analítica	Query	Justificativa da Query
Obter quantidade de livros por biblioteca	"Quantos livros cada biblioteca possui?"	`SELECT b.id_biblioteca AS biblioteca_id, b.nome_biblioteca AS biblioteca_nome, COUNT(lv.id_livro) AS quantidade_livros FROM biblioteca b JOIN local l ON b.id_biblioteca = l.fk_local_biblioteca`	Visão geral sobre a distribuição de livros entre as bibliotecas.

		JOIN livro lv ON l.id_local = lv.id_local GROUP BY b.id_biblioteca, b.nome_biblioteca ORDER BY b.id_biblioteca;	
Verificar quantos livros foram emprestados por cada biblioteca	"Quantos livros foram emprestados por cada biblioteca?"	SELECT b.nome_biblioteca, COUNT(e.id_emprestimo) AS num_emprestimos FROM biblioteca b JOIN local l ON b.id_biblioteca = l.fk_local_biblioteca JOIN livro lv ON l.id_local = lv.id_local JOIN emprestimo e ON lv.id_livro = e.id_livro GROUP BY b.id_biblioteca, b.nome_biblioteca ORDER BY num_emprestimos DESC;	Esse dado pode ser útil para avaliar o desempenho das bibliotecas em termos de empréstimos e para ajudar na gestão de acervo.
Listar livros mais reservados	"Quais são os livros mais reservados?"	SELECT id_livro, COUNT(*) AS num_reservas FROM reserva GROUP BY id_livro ORDER BY num_reservas DESC;	Ajuda a identificar os livros mais populares entre os usuários.

Consultar tipos de empréstimos realizados	"Quantos empréstimos de cada tipo foram realizados?"	`SELECT e.tipo_emprestimo, COUNT(e.id_emprestimo) AS num_emprestimos FROM emprestimo e GROUP BY e.tipo_emprestimo ORDER BY num_emprestimos DESC;`	Auxilia a identificar quais tipos são mais populares ou mais solicitados pelos usuários
Gêneros mais populares	"Quais gêneros literários são mais populares?"	SELECT id_genero, COUNT(*) AS num_livros FROM livro_genero GROUP BY id_genero ORDER BY num_livros DESC;	Fornece insights sobre a preferência dos usuários por gêneros.
Livros em locais com alta capacidade	"Quais livros estão em locais com capacidade acima de 100?"	SELECT l.* FROM livro l INNER JOIN local loc ON l.id_local = loc.id_local WHERE loc.capacidade > 100;	Avalia o uso do espaço nas bibliotecas.
Empréstimos por funcionário	"Quantos empréstimos cada funcionário realizou?"	SELECT id_funcionario_responsavel, COUNT(*) AS num_emprestimos FROM emprestimo GROUP BY id_funcionario_responsavel;	Mede a performance dos funcionários no atendimento ao público.
Livros disponíveis por gênero	Quantos livros estão disponíveis por gênero?	SELECT g.nome_genero, COUNT(lg.id_livro) AS num_livros_disponiveis FROM genero g INNER JOIN livro_genero lg ON g.id_genero = lg.id_genero INNER JOIN livro l ON lg.id_livro = l.id_livro	Esta consulta fornece insights sobre a disponibilidade de livros por gênero, ajudando a identificar quais gêneros possuem

		WHERE l.quantidade_disponivel > 0 GROUP BY g.nome_genero ORDER BY num_livros_disponiveis DESC;	mais ou menos opções disponíveis para os usuários.
Funcionários que mais processam reservas	Quais funcionários processam mais reservas?	SELECT f.nome, COUNT(r.id_reserva) AS num_reservas FROM funcionario f INNER JOIN reserva r ON f.id_funcionario = r.id_funcionario_responsavel GROUP BY f.nome ORDER BY num_reservas DESC;	Ajuda a avaliar a performance dos funcionários no processamento de reservas e a identificar quem está mais ativo nesta atividade.
Funcionários com mais multas aplicadas	Quantas multas cada funcionário aplicou?	SELECT f.nome AS funcionario_nome, COUNT(m.id_multa) AS num_multas FROM funcionario f JOIN multa m ON f.id_funcionario = m.id_funcionario_responsavel GROUP BY f.id_funcionario, f.nome ORDER BY num_multas DESC;	Identifica os funcionários com mais multas aplicadas no sistema.

5. Consultas Avançadas

```sql
-- Buscar o total de livros emprestados por um usuário e o valor total das multas:

SELECT usuario.nome, COUNT(emprestimo.id_emprestimo) AS total_emprestimos, SUM(multa.valor) AS total_multas FROM usuario JOIN emprestimo ON usuario.id_usuario = emprestimo.id_usuario JOIN multa ON emprestimo.id_emprestimo = multa.id_emprestimo WHERE usuario.id_usuario = 1 GROUP BY usuario.nome;

-- Buscar o número de livros disponíveis em cada biblioteca, separados por tipo de local:

SELECT biblioteca.nome_biblioteca, local.tipo_local, SUM(livro.quantidade_disponivel) AS total_disponivel FROM livro JOIN local ON livro.id_local = local.id_local JOIN biblioteca ON local.fk_local_biblioteca = biblioteca.id_biblioteca GROUP BY biblioteca.nome_biblioteca, local.tipo_local ORDER BY biblioteca.nome_biblioteca, total_disponivel DESC;



-- Buscar todos os livros, seus autores e o número de empréstimos realizados:

SELECT livro.titulo, autor.nome_autor, COUNT(emprestimo.id_emprestimo) AS total_emprestimos FROM livro

JOIN livro_autor ON livro.id_livro = livro_autor.id_livro

JOIN autor ON livro_autor.id_autor = autor.id_autor

LEFT JOIN emprestimo ON livro.id_livro = emprestimo.id_livro

GROUP BY livro.titulo, autor.nome_autor

ORDER BY total_emprestimos DESC;
```

6. Consultas Intermediárias

```
1. Buscar os livros disponíveis para empréstimo de uma biblioteca
específica:



SELECT livro.titulo, livro.quantidade_disponivel,
biblioteca.nome_biblioteca

FROM livro

JOIN local ON livro.id_local = local.id_local

JOIN biblioteca ON local.fk_local_biblioteca = biblioteca.id_biblioteca

WHERE biblioteca.id_biblioteca = 1

AND livro.quantidade_disponivel > 0;



2. Buscar todos os empréstimos de um usuário específico:



SELECT emprestimo.id_emprestimo, livro.titulo,
emprestimo.data_emprestimo, emprestimo.data_devolucao,
emprestimo.status

FROM emprestimo

JOIN livro ON emprestimo.id_livro = livro.id_livro

WHERE emprestimo.id_usuario = 1;



3. Buscar livros e seus autores:



SELECT livro.titulo, autor.nome_autor

FROM livro
```

```
JOIN livro_autor ON livro.id_livro = livro_autor.id_livro

JOIN autor ON livro_autor.id_autor = autor.id_autor;
```

7. Consultas Básicas

```
1. Inserir um novo livro na tabela:



INSERT INTO livro (titulo, editora, data_publicacao,
quantidade_disponivel, id_local, quantidade_total)

VALUES ('Dom Casmurro', 'Editora A', '1899-01-01', 5, 1, 10);



2. Alterar a quantidade disponível de um livro específico:



UPDATE livro

SET quantidade_disponivel = 3

WHERE id_livro = 1;



3. Excluir um livro da tabela livro:



DELETE FROM livro

WHERE id_livro = 1;



4. Alterar o telefone de um usuário:



UPDATE usuario

SET telefone = '(11) 99876-5432'
```

```
WHERE id_usuario = 1;
```

5. Inserir um novo usuário:

```
INSERT INTO usuario (nome, email, telefone, data_registro, endereco,
data_nascimento)

VALUES ('Maria Silva', 'maria@example.com', '(11) 98765-4321',
'2025-01-12', 'Rua das Flores, 123', '1995-02-25');
```

6. Excluir um usuário:

```
DELETE FROM usuario

WHERE id_usuario = 1;
```

7. Inserir uma nova multa:

```
INSERT INTO multa (id_funcionario_responsavel, id_usuario, valor,
data_multa, status_multa, descricao)

VALUES (2, 3, 50.00, '2025-01-12', 'Pendente', 'Multa por atraso no
retorno do livro');
```

8. Alterar o status de uma multa:

```
UPDATE multa

SET status_multa = 'Pago', data_pagamento = '2025-01-13'

WHERE id_multa = 1;
```

9. Excluir uma multa:

```sql
DELETE FROM multa
WHERE id_multa = 1;
```

10. Inserir um novo funcionário:

```sql
INSERT INTO funcionario (nome, cargo, data_contratacao, salario, email, telefone, id_biblioteca)
VALUES ('Carlos Oliveira', 'Assistente', '2025-01-01', 3000.00, 'carlos@biblioteca.com', '(11) 99876-5432', 1);
```

11. Alterar o salário de um funcionário:

```sql
UPDATE funcionario
SET salario = 3200.00
WHERE id_funcionario = 1;
```

12. Excluir um funcionário:

```sql
DELETE FROM funcionario
WHERE id_funcionario = 1;
```