UNIVERSIDADE DO ESTADO DO RIO DE JANEIRO
CIÊNCIA DA COMPUTAÇÃO

RAFAEL MANTEIGA BALBINO

SISTEMA DE GERENCIAMENTO DE ACERVOS DE DOCUMENTOS HISTÓRICOS

RIO DE JANEIRO
2024

1 MINIMUNDO: LABORATÓRIO DE HISTÓRIA E MEMÓRIA DA PSICOLOGIA

Este Laboratório é um espaço dedicado ao estudo e preservação da história da psicologia. O ambiente abriga diversos acervos de documentos históricos, incluindo artigos, livros, cartas, fotografias e outros materiais relacionados ao desenvolvimento da disciplina ao longo do século XX e XXI.

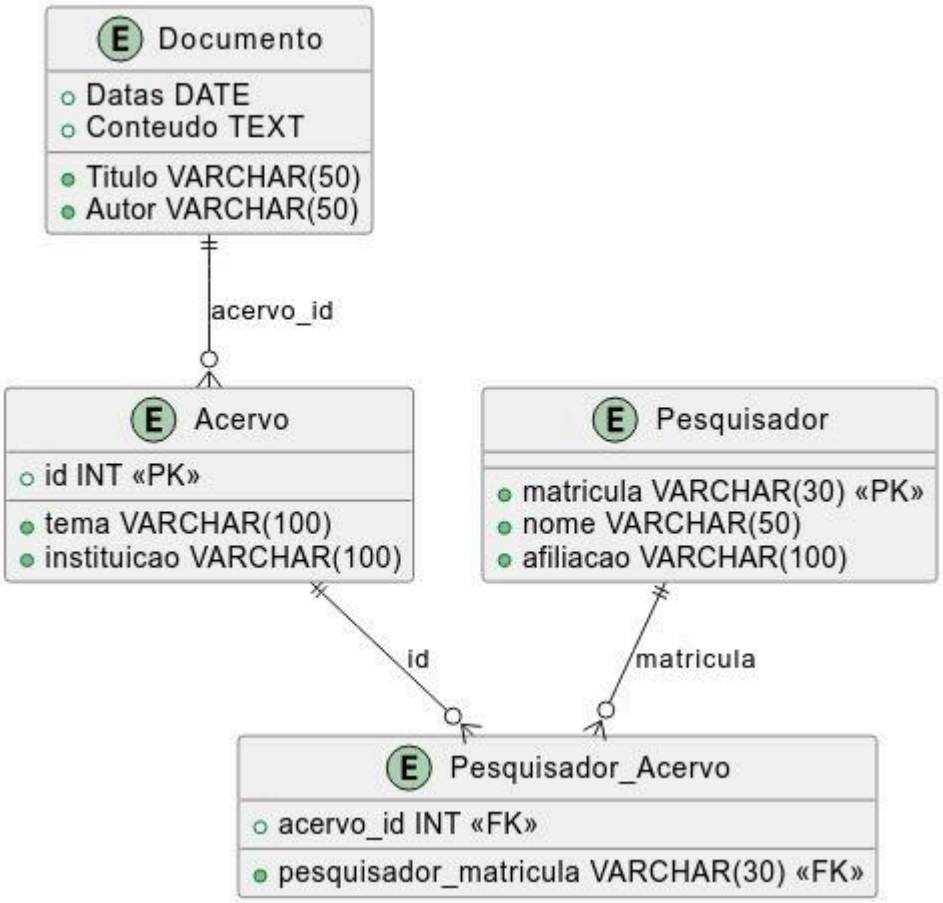
Entidades Principais

a. Documento: Representa um documento histórico, que pode incluir manuscritos, cartas, registros, etc. Cada documento possui um título, uma data, um autor (se conhecido) e um conteúdo.
b. Acervo: Refere-se a uma coleção de documentos. Cada acervo pode conter múltiplos documentos e é associado a um tema específico ou a uma instituição.
c. Pesquisador: Usuário do sistema que está envolvido na pesquisa e na catalogação de documentos. Cada pesquisador tem um nome, uma afiliação (universidade, instituto de pesquisa, etc.) e pode ter várias consultas de pesquisa.

Relacionamentos

a. Um documento pertence a um único acervo.
b. Um acervo pode conter múltiplos documentos.
c. Um pesquisador pode realizar várias consultas de pesquisa.

2 MODELAGEM DO BANCO DE DADOS



Foram criadas as entidades de acordo com as tabelas do banco de dados.

Um acervo pode possuir vários documentos, mas cada documento faz parte de apenas um acervo, uma relação de 1 para n. Logo, a tabela *Documento* possui a chave estrangeira chamada *acervo_id* que referencia a chave primária *id* da tabela *Acervo*.

Um acervo pode possuir diversos pesquisadores envolvidos, assim como cada pesquisador pode fazer parte de diversos acervos. Dessa forma, foi criada uma tabela auxiliar *Pesquisador_Acervo* para gerar uma relação de n para n entre pesquisadores e acervos. O pesquisador é vinculado ao acervo pela tabela *Pesquisador_Acervo* através de sua matrícula em *pesquisador_matricula*; já o acervo é vinculado ao pesquisador pela tabela *Pesquisador_Acervo* através de seu id, através de *acervo_id*.

3 IMPLEMENTAÇÃO DO BANCO DE DADOS

```
CREATE TABLE Documento(
  Titulo VARCHAR(50),
  Datas DATE,
  Autor VARCHAR(50),
  Conteudo TEXT
);
CREATE TABLE Acervo(
  id SERIAL PRIMARY KEY,
  tema VARCHAR(100),
  instituicao VARCHAR(100)
);
ALTER TABLE
  documento
ADD
  COLUMN acervo_id INT,
ADD
  FOREIGN KEY (acervo_id) REFERENCES Acervo(id);
CREATE TABLE pesquisador(
  nome VARCHAR(50),
  afiliacao VARCHAR(100),
  matricula VARCHAR(30) PRIMARY KEY
);
CREATE TABLE Pesquisador_Acervo(
  pesquisador_matricula VARCHAR(30) REFERENCES Pesquisador(matricula),
  acervo_id INT REFERENCES Acervo(id),
  PRIMARY KEY (
    pesquisador_matricula, acervo_id
  )
);
CREATE INDEX idx_datas_brin ON Documento USING BRIN (Datas);
INSERT INTO Acervo (tema, instituicao)
VALUES
  ('Emilio Mira y López', 'UERJ');
INSERT INTO Acervo (tema, instituicao)
VALUES
  ('Celso Pereira de Sá', 'UERJ');
INSERT INTO Acervo (tema, instituicao)
VALUES
  ('Isabel Adrados', 'UERJ');
INSERT INTO Acervo (tema, instituicao)
VALUES
  ('Eliezer Schneider', 'UERJ');
INSERT INTO Acervo (tema, instituicao)
VALUES
  ('Antonio Gomes Penna', 'UERJ');
INSERT INTO Documento(
  Titulo, Datas, Autor, Conteudo, acervo_id
)
```

```
VALUES
  (
    'Teoria y práctica del psicoanálisis',
    '01/02/1926', 'Emilio Mira y López',
    'Visão geral da psicanálise freudiana.',
    1
  );
INSERT INTO Documento(
  Titulo, Datas, Autor, Conteudo, acervo_id
)
VALUES
  (
    'Manual de Psicología Jurídica',
    '01/01/1932', 'Emilio Mira y Lopez',
    'Explora a interface entre psicologia e direito.',
    1
  );
INSERT INTO Documento(
  Titulo, Datas, Autor, Conteudo, acervo_id
)
VALUES
  (
    'Manual de Psiquiatria', '03/06/1935',
    'Emilio Mira y Lopez', 'Percepção dos transtornos mentais da época.',
    1
  );
INSERT INTO Documento(
  Titulo, Datas, Autor, Conteudo, acervo_id
)
VALUES
  (
    'Psicodiagnóstico Miocinético (PMK)',
    '07/11/1940', 'Emilio Mira y Lopez',
    'Teste projetivo para avaliar o indivíduo.',
    1
  );
INSERT INTO Documento(
  Titulo, Datas, Autor, Conteudo, acervo_id
)
VALUES
  (
    'Cuatro gigantes del alma', '09/08/1959',
    'Emilio Mira y Lopez', 'Explora as 4 emoções básicas.',
    1
  );
INSERT INTO Documento(
  Titulo, Datas, Autor, Conteudo, acervo_id
)
VALUES
```

```
  (
    'Estudos de Psicologia Social',
    '23/05/2015', 'Celso Pereira de Sá',
    'Manual sobre vertentes da Psicologia Social.',
    2
  );
INSERT INTO Documento(
  Titulo, Datas, Autor, Conteudo, acervo_id
)
VALUES
  (
    'Pesquisa em Representações Sociais',
    '03/09/1998', 'Celso Pereira de Sá',
    'Metodologia em representações sociais.',
    2
  );
INSERT INTO Documento(
  Titulo, Datas, Autor, Conteudo, acervo_id
)
VALUES
  (
    'Memórias do Descobrimento do Brasil',
    '25/05/2005', 'Celso Pereira de Sá',
    'Disserta sobre a história do Brasil.',
    2
  );
INSERT INTO Documento(
  Titulo, Datas, Autor, Conteudo, acervo_id
)
VALUES
  (
    'Psicologia do Controle Social',
    '03/09/1973', 'Celso Pereira de Sá',
    'Perspectivas de questões sociológicas na sociologia.',
    2
  );
INSERT INTO Documento(
  Titulo, Datas, Autor, Conteudo, acervo_id
)
VALUES
  (
    'Sobre a Psicologia Social no Brasil',
    '21/10/2007', 'Celso Pereira de Sá',
    'História da Psicologia Social no Brasil.',
    2
  );
INSERT INTO Documento(
  Titulo, Datas, Autor, Conteudo, acervo_id
)
```

```
VALUES
  (
    'Orientação Infantil', '09/03/1988',
    'Isabel Adrados', 'Explanação sobre questões da infância.',
    3
  );
INSERT INTO Documento(
  Titulo, Datas, Autor, Conteudo, acervo_id
)
VALUES
  (
    'A Técnica de Rorschach em crianças',
    '08/04/1985', 'Isabel Adrados',
    'Manual de aplicação de testes projetivos em crianças.',
    3
  );
INSERT INTO Documento(
  Titulo, Datas, Autor, Conteudo, acervo_id
)
VALUES
  (
    'Teoria e Prática do Teste de Rorschach',
    '09/11/1982', 'Isabel Adrados',
    'Manual teórico de testes projetivos em crianças.',
    3
  );
INSERT INTO Documento(
  Titulo, Datas, Autor, Conteudo, acervo_id
)
VALUES
  (
    'Rorschach na adolescência normal e patológica',
    '07/06/1976', 'Isabel Adrados',
    'Aplicação de Rorschach na adolescência.',
    3
  );
INSERT INTO Documento(
  Titulo, Datas, Autor, Conteudo, acervo_id
)
VALUES
  (
    'Manual de Psicodiagnóstico', '08/08/1982',
    'Isabel Adrados', 'Instrução da atuação do psicólogo frente a diagnósticos.',
    3
  );
INSERT INTO Documento(
  Titulo, Datas, Autor, Conteudo, acervo_id
)
VALUES
```

```
  (
    'Eliezer Schneider', '03/06/2001',
    'Ana Maria Jacó-Vilela', 'Coleção: Pioneiros da Psicologia Brasileira',
    4
  );
INSERT INTO Documento(
  Titulo, Datas, Autor, Conteudo, acervo_id
)
VALUES
  (
    'Psicologia Social - Histórica Cultural e Política',
    '28/09/1978', 'Eliezer Schneider',
    'Manual de Psicologia Social',
    4
  );
INSERT INTO Documento(
  Titulo, Datas, Autor, Conteudo, acervo_id
)
VALUES
  (
    'Eliezer Schneider, professor, 81 anos',
    '04/09/1998', 'Antônio Gomes Penna',
    'Jornal O Globo - Obituário',
    4
  );
INSERT INTO Documento(
  Titulo, Datas, Autor, Conteudo, acervo_id
)
VALUES
  (
    'Eliezer Schneider', '23/10/1996',
    'Elza Maria do Socorro Dutra',
    'Entrevista com Eliezer Schneider',
    4
  );
INSERT INTO Documento(
  Titulo, Datas, Autor, Conteudo, acervo_id
)
VALUES
  (
    'Eliezer Schneider', '31/01/1971',
    'Jornal do Brasil', 'Palestra de Eliezer Schneider sobre conflito de gerações.',
    4
  );
INSERT INTO Documento(
  Titulo, Datas, Autor, Conteudo, acervo_id
)
VALUES
  (
```

```
    'História da Psicologia no Rio de Janeiro',
    '05/04/1992', 'Antonio Gomes Penna',
    'Desenvolvimento da Psicologia no Rio de Janeiro',
    5
  );
INSERT INTO Documento(
  Titulo, Datas, Autor, Conteudo, acervo_id
)
VALUES
  (
    'História das ideias psicológicas',
    '05/01/1991', 'Antonio Gomes Penna',
    'Comentários epistemológicos sobre as ideias psicológicas.',
    5
  );
INSERT INTO Documento(
  Titulo, Datas, Autor, Conteudo, acervo_id
)
VALUES
  (
    'Antonio Gomes Penna: Uma trajetória (...)',
    '04/08/2010', 'Ana Maria Jacó-Vilela',
    'Artigo biográfico sobre o Antonio Gomes Penna.',
    5
  );
INSERT INTO Documento(
  Titulo, Datas, Autor, Conteudo, acervo_id
)
VALUES
  (
    'Ciclo de palestra sobre Psicologia do futebol',
    '01/09/1970', 'Nilton Ribeiro',
    'Notícia  - O Jornal', 5
  );
INSERT INTO Documento(
  Titulo, Datas, Autor, Conteudo, acervo_id
)
VALUES
  (
    'Introdução à Psicologia Contemporânea',
    '09/08/1978', 'Antonio Gomes Penna',
    'Livro sobre a constituição da psicologia moderna.',
    5
  );
INSERT INTO Pesquisador(nome, afiliacao, matricula)
VALUES
  (
    'Camila Rocha da Silva', 'UFRJ',
    '1236987'
```

```
  );
INSERT INTO Pesquisador(nome, afiliacao, matricula)
VALUES
  (
    'Maria Luiza Oliveira dos Santos',
    'UFRRJ', '9865321'
  );
INSERT INTO Pesquisador(nome, afiliacao, matricula)
VALUES
  (
    'Felipe Cavalcante de Souza', 'UFPR',
    '2301478'
  );
INSERT INTO Pesquisador(nome, afiliacao, matricula)
VALUES
  (
    'Bianca Gonçalves Lima', 'UERJ',
    '7453200'
  );
INSERT INTO Pesquisador(nome, afiliacao, matricula)
VALUES
  (
    'Mariana Almeida Araújo', 'UFRS',
    '3698745'
  );
INSERT INTO pesquisador_acervo(
  pesquisador_matricula, acervo_id
)
VALUES
  ('1236987', 1);
INSERT INTO pesquisador_acervo(
  pesquisador_matricula, acervo_id
)
VALUES
  ('9865321', 1);
INSERT INTO pesquisador_acervo(
  pesquisador_matricula, acervo_id
)
VALUES
  ('2301478', 2);
INSERT INTO pesquisador_acervo(
  pesquisador_matricula, acervo_id
)
VALUES
  ('7453200', 2);
INSERT INTO pesquisador_acervo(
  pesquisador_matricula, acervo_id
)
VALUES
```

```
  ('3698745', 3);
INSERT INTO pesquisador_acervo(
  pesquisador_matricula, acervo_id
)
VALUES
  ('1236987', 3);
INSERT INTO pesquisador_acervo(
  pesquisador_matricula, acervo_id
)
VALUES
  ('2301478', 4);
INSERT INTO pesquisador_acervo(
  pesquisador_matricula, acervo_id
)
VALUES
  ('3698745', 4);
INSERT INTO pesquisador_acervo(
  pesquisador_matricula, acervo_id
)
VALUES
  ('7453200', 5),
  ('9865321', 5);

-- Consulta do cenário de utilização
SELECT
  d.Titulo,
  d.Datas,
  d.Autor,
  d.Conteudo,
  a.tema AS Acervo_Tema,
  STRING_AGG(p.nome, ', ') AS Pesquisador_Nome
FROM
  Documento d
  JOIN Acervo a ON d.acervo_id = a.id
  JOIN Pesquisador_Acervo pa ON d.acervo_id = pa.acervo_id
  JOIN Pesquisador p ON pa.pesquisador_matricula = p.matricula
WHERE
  d.Datas BETWEEN '1970-01-01'
  AND '1979-12-31'
GROUP BY
  d.Titulo,
  d.Datas,
  d.Autor,
  d.Conteudo,
  a.tema;
```

4 ÍNDICE NÃO PADRÃO

A tabela Documento foi escolhida para criar um índice não convencional. Vamos supor que, para otimizar consultas específicas, queremos indexar a coluna Datas para realizar buscas rápidas por documentos em um intervalo de datas específico.

A escolha da coluna Datas para a criação do índice não padrão se baseia na necessidade comum de pesquisadores de filtrar documentos por período de tempo. Isso é particularmente útil em estudos históricos, onde os documentos são frequentemente organizados e pesquisados por sua data de criação.

Para este caso, podemos utilizar a função de indexação BRIN (Block Range Index) do PostgreSQL. Esta função salva os valores indexados de forma resumida em blocos consecutivos. Desta forma, torna-se mais conveniente para as colunas em que os valores se correlacionam com a ordem das linhas da tabela (Indexes [...], 2024).

Essa função é útil quando os valores de uma coluna são distribuídos uniformemente ao longo da tabela, como no caso de datas. Para usar o índice BRIN em consultas, foi apenas criado o índice e, após a criação do índice, o otimizador de consultas do PostgreSQL deve automaticamente considerar o índice BRIN ao decidir o plano de execução para consultas que se beneficiam dele. Dessa forma, não foi necessário fazer nada específico além da criação do BRIN.

5 CENÁRIO DE UTILIZAÇÃO

Um pesquisador está envolvido em um estudo sobre a evolução das teorias da psicologia durante o século XX. Ele está interessado em encontrar documentos relevantes, como artigos, cartas e ensaios, produzidos por psicólogos na década de 1970, período crucial para o desenvolvimento dessa disciplina.

Com o sistema de gerenciamento de acervos do Laboratório de História e Memória da Psicologia, o pesquisador pode utilizar o índice não padrão implementado na coluna de datas dos documentos para realizar consultas eficientes. Ao realizar uma busca utilizando o intervalo de datas específico (1970-1979), o pesquisador pode rapidamente acessar documentos que abordam as teorias e contribuições dos teóricos.

Esses documentos podem incluir trabalhos pioneiros de psicólogos como Emilio Mira y Lopez, Celso Pereira de Sá, entre outros, fornecendo ideias valiosas sobre as influências, debates e desenvolvimentos na psicologia ao longo do tempo. Com acesso rápido e eficiente aos documentos relevantes, o pesquisador pode avançar em sua análise e contribuir para o entendimento da história e evolução da psicologia.

6 VISÕES

Visão é uma consulta que é armazenada para produzir o resultado posteriormente, quando necessário e quantas vezes forem necessárias. A visão fica gravada no banco de dados e realiza a consulta com os dados da tabela atualizados, mesmo que a tabela seja atualizada após a criação da visão, pois a visão não grava os dados em si, mas a definição da consulta (Using [...], 2024).

6.1 DOCUMENTOS DO ACERVO DO CELSO PEREIRA DE SÁ

A visão demonstrada foi criada para melhorar e facilitar a visualização dos documentos do acervo do Celso Pereira de Sá. Para maior clareza na visualização, foram selecionadas apenas as colunas *Titulo*, *Datas* e *Conteudo*. O autor não foi incluído na consulta porque o próprio Celso é o autor de todos os documentos de seu acervo; além disso, faz-se necessário mostrar que é possível ocultar colunas de tabelas na visão. O ID do acervo também não se faz necessário, já que a visão é criada para a visualização de um autor específico.

Esta visão nada mais é do que a seleção de três colunas da tabela *Documento*. A condição imposta é que o *acervo_id* seja igual a 2 para buscar apenas documentos do Celso, conforme mencionado acima. Além disso, os documentos foram ordenados por data (em ordem crescente, por padrão).

Segue o código da criação da visão e seu uso:

```
CREATE VIEW documentoscelso AS
SELECT
  d.Titulo,
  d.Datas,
  d.Conteudo
FROM
  Documento d
WHERE
  d.acervo_id = 2
ORDER BY
  d.Datas;

-- Mostrando a visão criada
SELECT
  *
FROM
  documentoscelso;
```

6.2 PESQUISADORES DO ACERVO DO EMILIO MIRA Y LÓPEZ

A visão demonstrada foi criada para salvar uma tabela de visualização fácil de informações sobre os pesquisadores envolvidos especificamente no acervo do Myra.

A facilidade de visualização é consequência do fato de selecionar apenas *nome* da tabela *pesquisador* e *pesquisador_matricula* da tabela *pesquisador_matricula*. Com o uso do comando *JOIN*, é possível vincular a matrícula (presente nas duas tabelas) e gerar informações de duas tabelas diferentes em apenas uma *view*.

Dessa forma, as informações requeridas são selecionadas de cada tabela e vinculadas. Além disso, cria-se a condição do *acervo_id* ser igual a 1, pois os pesquisadores requeridos são apenas os do acervo do Mira.

Segue o código da criação da visão e seu uso:

```
CREATE VIEW pesquisadoresemilio AS
SELECT
  p.nome,
  pa.pesquisador_matricula
FROM
  pesquisador p
  JOIN pesquisador_acervo pa ON pa.pesquisador_matricula = p.matricula
WHERE
  pa.acervo_id = 1;

-- Mostrando a visão criada
SELECT
  *
FROM
  pesquisadoresemilio;
```

6.3 DOCUMENTOS POR PESQUISADORES

A visão demonstrada foi criada para salvar uma tabela contendo todos os documentos acompanhados dos pesquisadores envolvidos neles. A ideia é conseguir verificar quais os pesquisadores envolvidos em cada documento de forma rápida, sem precisar averiguar a quais acervos os pesquisadores estão relacionados ou a quais acervos os documentos estão relacionados também. A visão mostra apenas o título do documento e os nomes dos pesquisadores.

A melhoria na visualização vem da ideia de selecionar apenas as colunas *titulo* e *nome* das tabelas *documento* e *pesquisador*, respectivamente. Com o uso do comando *JOIN*, foram vinculados *pesquisador_acervo* com *acervo* pela coluna *id* e *pesquisador_acervo* com *pesquisador* pela coluna *matricula*. O JOIN duplo fez-se necessário porque o relacionamento

entre *acervo* e *pesquisador* é criado pela tabela auxiliar *pesquisador_acervo*, fazendo com que houvesse necessidade de criar a junção com esta tabela para gerar a conexão entre pesquisador e documento.

Por fim, para evitar repetição de linhas, foi utilizado o agrupamento por título e conteúdo, fazendo com que cada documento aparecesse apenas uma vez.

Segue o código da criação da visão e seu uso:

```
CREATE VIEW documentos_por_pesquisadores AS
SELECT
  titulo,
  nomes_pesquisadores
FROM
  (
    SELECT
      d.titulo,
      d.conteudo,
      STRING_AGG(p.nome, ', ') AS nomes_pesquisadores
    FROM
      documento d
      JOIN pesquisador_acervo pa ON d.acervo_id = pa.acervo_id
      JOIN pesquisador p ON pa.pesquisador_matricula = p.matricula
    GROUP BY
      d.titulo,
      d.conteudo
  ) sub
ORDER BY
  titulo;

-- Mostrando a visão criada
SELECT
  *
FROM
  documentos_por_pesquisadores;
```

7 FUNÇÕES

Função é uma consulta que encapsula uma lista de declarações para retornar um resultado dinâmico (ou seja, o resultado condiz com o momento em que a função é chamada) de acordo com um parâmetro de entrada e/ou com a declaração específica dela (Extending [...], [2024?]).

A função se diferencia da visão pelo fato de que uma função pode realizar procedimentos de manipulações de dados, cálculos complexos, validações etc., não retornando apenas dados para visualização, mas também modificando, adicionando e calculando dados, se necessário (Extending [...], [2024?]).

A função possui declaração de fim com a palavra chave END e um identificador no início e no final, que é o *$$*. Este identificador é usado para marcar o início das declarações em linguagem *plpgsql* e para marcar onde termina o uso da linguagem (Extending [...], [2024?]).

7.1 DOCUMENTOS POR AUTOR

A função demonstrada foi criada para facilitar a pesquisa na tabela *documento* por autor. Para isso, foi criado um parâmetro chamado *autor_param* para receber o nome digitado na pesquisa e foi definido o retorno do tipo TABLE, pois a função retornará a tabela com as informações requeridas, que foram definidas como sendo *titulo*, *datas*, *conteudo* e *acervo_id*.

Após isso, foi definido que a função retorna uma busca. A busca começa selecionando as colunas desejadas da tabela *documento* e, nos casos de textos, foi definido o tipo TEXT, pois as colunas são do tipo VARCHAR, mas foi preferido retornar como TEXT para não haver limites no tamanho da *string*.

A consulta seguiu selecionando a tabela *documento*, como dito anteriormente, e limitou a busca à condição do parâmetro *autor_param* ser igual ao dado da coluna *autor* da tabela *documento*, filtrando apenas os dados do autor pesquisado. Após isso, por questão de organização, foi definida a ordem da tabela pela data do documento, usando a coluna *datas*.

Segue o código da criação da função e seu uso:

```
CREATE
OR REPLACE FUNCTION obter_documentos_por_autor(
  autor_param VARCHAR(50)
) RETURNS TABLE (
  titulo TEXT, datas DATE, conteudo TEXT,
  acervo_id INT
) AS $$ BEGIN RETURN QUERY
SELECT
  documento.titulo :: TEXT,
  documento.datas,
  documento.conteudo :: TEXT,
  documento.acervo_id
FROM
  documento
WHERE
  autor_param = autor
ORDER BY
  datas;
END;
$$ LANGUAGE plpgsql;
```

```
-- Usando a função criada
SELECT
  *
FROM
  obter_documentos_por_autor('Celso Pereira de Sá');
```

7.2 PESQUISADORES POR ACERVO

A função demonstrada foi criada para facilitar a busca por pesquisadores que estão associados ao acervo desejado. Esta é uma função simples. Para realizar a busca desejada, ela recebe o parâmetro *id_param* e retorna *nome* e *tema* de uma tabela.

A função seleciona o nome e o tema, sendo nome de *pesquisador*, que foi referenciado como *pp*; e tema de *acervo*, que foi referenciado como *aa*. O tema foi escolhido para ser selecionado para também ser possível verificar o acervo que condiz com o ID pesquisado, tornando a consulta mais informativa e clara.

Para fazer a pesquisa correta, foi feita a junção de *pesquisador* com *pesquisador_acervo* utilizando a matrícula como vínculo. Após isto, foi realizada a junção de *pesquisador_acervo* com *acervo* utilizando a mesma lógica, mas usando o ID do acervo como vínculo.

Com a criação destes vínculos, pesquisador e acervo puderam ser vinculados indiretamente através da tabela *pesquisador_acervo*.

Após isso, foi criado o filtro desejado utilizando o parâmetro criado anteriormente, *id_param*. O parâmetro foi comparado ao *acervo_id* da tabela *pesquisador_acervo* para filtrar os pesquisadores. Como anteriormente foram selecionados *nome* e *tema*, foi possível ordenar pelo nome para organizar melhor os dados.

Segue o código da criação da função e seu uso:

```
CREATE
OR REPLACE FUNCTION obter_pesquisadores_por_acervoid(id_param INT)
RETURNS TABLE(
  nome VARCHAR(50),
  tema VARCHAR(50)
) AS $$ BEGIN RETURN QUERY
SELECT
  pp.nome,
  aa.tema
FROM
  pesquisador pp
  JOIN pesquisador_acervo pa ON pp.matricula = pa.pesquisador_matricula
  JOIN acervo aa ON pa.acervo_id = aa.id
```

```
WHERE
  pa.acervo_id = id_param
ORDER BY
  pp.nome;
end;
$$ LANGUAGE plpgsql;

-- Usando a função criada
SELECT
  *
FROM
  obter_pesquisadores_por_acervoid(1);
```

7.3 CONTAGEM DE DOCUMENTOS POR TEMA

A função demonstrada foi criada para conseguir visualizar a quantidade de documentos em todos os acervos que possuem um tema específico.

A função recebe um tema como parâmetro para verificá-lo e retorna um número inteiro, pois retorna apenas a quantidade de documentos. Após receber o tema, a função seleciona a coluna *id*, pois é a chave primária da tabela *documento*, e, por isso, há garantia de não estar nula. Porém, a seleção utiliza a função *COUNT* que conta o número de linhas naquela coluna. A seleção se faz na tabela *documento*.

O *JOIN* é utilizado para posteriormente utilizar a coluna *tema* da tabela *acervo*. O *JOIN* faz a junção do acervo com o documento pelo *id* de *acervo* (chave primária) e *acervo_id* de *documento* (chave estrangeira de *documento*).

Após a junção feita, é possível utilizar o filtro para selecionar apenas o tema digitado. Isto é feito no comando *WHERE* comparando o parâmetro recebido *temaparam* e o tema do acervo que corresponde ao id selecionado no início.

```
CREATE
OR REPLACE FUNCTION obter_quantidade_documentos_por_tema(
  temaparam VARCHAR(50)
) RETURNS INT AS $$ BEGIN RETURN(
  SELECT
    COUNT(d.id)
  FROM
    documento d
    JOIN acervo aa ON aa.id = d.acervo_id
  WHERE
    temaparam = aa.tema
);
END;
$$ LANGUAGE plpgsql;
```

```
-- Usando a função criada
SELECT
  *
FROM
  obter_quantidade_documentos_por_tema('Isabel Adrados');
```

8 TRIGGERS

Trigger é uma tarefa a ser executada no banco de dados de forma automática antes ou depois da execução de algum comando especificado. A trigger é disparada e, após verificar a condição que indicada, executa uma função específica caso esta condição seja atendida, fazendo com que seja desnecessária a execução manual (PostgreSQL [...], [2024?]).

8.1 VERIFICAR DATA DOS DOCUMENTOS

Por se tratar de um banco de dados voltado para pesquisa em História, não faz sentido que o documento tenha data futura. Esta *trigger* foi criada para verificar se a data de um documento que foi inserido é de uma data posterior à data atual.

Para isso, foi criada uma função que verifica se a data do documento inserido é posterior à data atual e, caso seja, gera uma exceção, a mensagem da exceção é indicada e o documento é impedido de ser inserido.

Para isso, a *trigger* precisa ser disparada antes do comando, por isso foi utilizado o *BEFORE*. Também foi indicada a verificação antes de um *UPDATE* para que não modifiquem para uma data futura após o documento já ter sido criado.

```
CREATE
OR REPLACE FUNCTION validar_data() RETURNS TRIGGER AS $$ BEGIN IF
NEW.datas > CURRENT_DATE THEN RAISE EXCEPTION 'A data do documento não
pode ser posterior à data atual.';
END IF;
RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER verificar_data BEFORE INSERT
OR
UPDATE
  ON documento FOR EACH ROW EXECUTE FUNCTION validar_data();
```

8.2 IMPEDIR A EXCLUSÃO DE ACERVOS QUE POSSUEM DOCUMENTOS

Esta *trigger* foi criada com o objetivo de impedir a exclusão de acervos que possuem documentos associados. Caso algum documento tenha sido atribuído a um acervo, ele deve ter seu relacionamento trocado para outro acervo antes da exclusão do acervo que o detém. Isso impede que haja exclusão de acervos por descuido ou por engano. Além disso, garante que os documentos tenham suas configurações tratadas em relação aos acervos sem que eles fiquem desassociados.

Para isso, foi criada uma simples função que verifica se existe pelo menos um documento associado ao acervo que será excluído. Usando *WHERE* é possível verificar se o documento selecionado possui algum vínculo pelo *acervo_id* e *id*. Caso o vínculo exista, o *SELECT* encontrará um documento e o *IF EXISTS* indicará que existe, caindo no *THEN* que cria uma exceção, parando o bloco que código sem realizar a exclusão e mostrando a mensagem.

Ao criar a *trigger*, foi também incluído o comando *BEFORE* para garantir que a verificação aconteça antes do comando *DELETE*.

```
CREATE
OR REPLACE FUNCTION impedir_exclusao_acervo() RETURNS TRIGGER AS $$
BEGIN IF EXISTS (
  SELECT
    1
  FROM
    documento
  WHERE
    documento.acervo_id = OLD.id
) THEN RAISE EXCEPTION 'Um acervo com documentos associados não pode ser
excluído.';
END IF;
RETURN OLD;
END;
$$ LANGUAGE plpgsql;

CREATE
OR REPLACE TRIGGER verificar_acervo BEFORE DELETE ON acervo FOR EACH
ROW EXECUTE FUNCTION impedir_exclusao_acervo();
```

8.3 LOG PARA ATUALIZAÇÕES NO TÍTULO DO DOCUMENTO SEM ALTERAÇÕES NO CONTEÚDO

Esta *trigger* foi criada com o objetivo de guardar alterações no título dos documentos onde não tenha havido alterações no conteúdo. Este *log* gerado faz-se necessário pois é incomum um documento mudar de nome, sendo provável que tenha sido modificado por engano ou por descuido.

Para salvar os *logs*, a tabela *log_atualizacao* foi criada. Ela guarda o id do documento, o título antigo, o título novo e a data da modificação. Assim, caso tenha havido um engano na mudança, o erro facilmente poderá ser encontrado e corrigido.

Para chegar ao objetivo, a função compara o título e conteúdo antigos com os novos sempre que há um *UPDATE* na tabela *documento*. Se o título novo seja diferente do antigo e o conteúdo seja o mesmo, ele entra no *THEN* e insere os dados na tabela de *log* junto com a data da modificação.

Para a criação da trigger, foi utilizado o *BEFORE* para garantir que não haja mudanças na tabela sem que as modificações atendam ao critério da *trigger*.

Ao fim, foi realizado um *UPDATE* que gera um *log* com o objetivo de mostrar o funcionamento.

```
CREATE TABLE log_atualizacao(
  iddocumento INT,
  tituloantigo VARCHAR(50),
  titulonovo VARCHAR(50),
  datadamodificacao DATE
);

CREATE
OR REPLACE FUNCTION registro_atualiza() RETURNS TRIGGER AS $$ BEGIN IF
NEW.titulo <> OLD.titulo
AND NEW.conteudo = OLD.conteudo THEN INSERT INTO log_atualizacao(
  iddocumento, tituloantigo, titulonovo,
  datadamodificacao
)
VALUES
  (
    OLD.id, OLD.titulo, NEW.titulo, CURRENT_DATE
  );
END IF;
RETURN NEW;
END;
$$ LANGUAGE plpgsql;

-- BEFORE, pois caso dê algum problema no UPDATE, as mudanças não serão acionadas.
```

```
CREATE
or REPLACE TRIGGER atualiza_registro
BEFORE
UPDATE
  ON documento FOR EACH ROW EXECUTE FUNCTION registro_atualiza();

UPDATE
  documento
SET
  titulo = 'Novo Título',
  conteudo = 'Visão geral da psicanálise freudiana.'
WHERE
  id = 1;
```

9 PLANO DE EXECUÇÃO

O plano de execução pode ser visualizado com o comando *EXPLAIN*. Ao utilizar o comando, é exibido o plano de execução do comando em que foi aplicado. Este plano pode mostrar detalhes sobre a varredura, a quantidade de tabelas referenciadas e sobre as junções realizadas (EXPLAIN [...], [2012?]).

Além disso, também pode ser mostrado o custo estimado da execução, com detalhes sobre o tempo levado para executar. Além disso, com o uso do comando ANALYZE em conjunto, o comando pode ser executado enquanto gera o plano de execução (EXPLAIN [...], [2012?]).

9.1 PLANO DE EXECUÇÃO DA CONSULTA DO CENÁRIO DE UTILIZAÇÃO

O comando escolhido para gerar o plano de execução foi o do cenário de utilização do índice *idx_datas_brin* criado. Este comando foi escolhido por ser o maior e mais complexo comando de todo o código do Banco de Dados.

Ele começa pelo comando *GROUP BY*: neste comando, a saída (Output) é representada pelos campos que estão no *SELECT*. Após mostrar a saída, ele mostra os campos escolhidos para o agrupamento (Group Key). Este comando apresentou um custo total estimado de 29,30.

O *Sort* foi realizado antes do *GROUP BY* para ordenar os dados e realizar o agrupamento. É uma preparação para a realização do agrupamento. Neste comando o custo total foi estimado em 29,19.

O uso de diversos comandos do tipo *Nested Loop* mostra que o PostgreSQL utilizou uma estratégia de junção (*JOIN*) para combinar as tabelas. O otimizador escolhe,

possivelmente, a melhor estratégia de junção para o caso. Estes loops condizem com cada *JOIN* inserido no comando. Nestes comandos o custo médio estimado foi de 26,51.

Após isso, o comando faz uma busca sequencial pelos documentos. Após a busca, ele gera uma saída com as colunas que foram escolhidas: título, data, autor, conteúdo e o ID do acervo. Para realizar a busca, foi utilizado o filtro de datas apresentado. Neste comando o custo total foi estimado em 13,90.

Ele busca em seguida o acervo pela chave primária *id* e gera a saída com as colunas *id*, *tema* e *instituicao* de acordo com a condição exposta de igualdade de *id's*. Neste comando o custo total foi estimado em 8,16.

Após isso, ele faz o mesmo com *pesquisador_acervo* usando a chave primária e retornando a matrícula e o ID do pesquisador, mas com a condição dos *id's* da tabela *Acervo* e *pesquisador_acervo*. Neste comando o custo total foi estimado em 6,02.

Após isto, ele faz uma busca por pesquisadores utilizando a chave primária, gerando a saída com nome, afiliação e matrícula. A condição estabelecida para o índice foi a de que *matricula* e *pesquisador_matricula* fossem iguais em tipo *TEXT*.

```
EXPLAIN VERBOSE
SELECT
  d.Titulo,
  d.Datas,
  d.Autor,
  d.Conteudo,
  a.tema AS Acervo_Tema,
  STRING_AGG(p.nome, ', ') AS Pesquisador_Nome
FROM
  Documento d
  JOIN Acervo a ON d.acervo_id = a.id
  JOIN Pesquisador_Acervo pa ON d.acervo_id = pa.acervo_id
  JOIN Pesquisador p ON pa.pesquisador_matricula = p.matricula
WHERE
  d.Datas BETWEEN '1970-01-01'
  AND '1979-12-31'
GROUP BY
  d.Titulo,
  d.Datas,
  d.Autor,
  d.Conteudo,
  a.tema;
```

```
QUERY PLAN
-------------------------------------------------------------------------------------------------------------
 GroupAggregate  (cost=29.18..29.30 rows=4 width=522)
   Output: d.titulo, d.datas, d.autor, d.conteudo, a.tema, string_agg((p.nome)::text, ', '::text)
   Group Key: d.titulo, d.datas, d.autor, d.conteudo, a.tema
   ->  Sort  (cost=29.18..29.19 rows=4 width=608)
         Output: d.titulo, d.datas, d.autor, d.conteudo, a.tema, p.nome
         Sort Key: d.titulo, d.datas, d.autor, d.conteudo, a.tema
         ->  Nested Loop  (cost=0.44..29.14 rows=4 width=608)
               Output: d.titulo, d.datas, d.autor, d.conteudo, a.tema, p.nome
               Inner Unique: true
               ->  Nested Loop  (cost=0.29..28.23 rows=4 width=568)
                     Output: d.titulo, d.datas, d.autor, d.conteudo, a.tema,
pa.pesquisador_matricula
                     Join Filter: (d.acervo_id = pa.acervo_id)
                     ->  Nested Loop  (cost=0.14..22.16 rows=1 width=498)
                           Output: d.titulo, d.datas, d.autor, d.conteudo, d.acervo_id, a.tema, a.id
                           Inner Unique: true
                           ->  Seq Scan on public.documento d  (cost=0.00..13.90 rows=1
width=276)
                                 Output: d.titulo, d.datas, d.autor, d.conteudo, d.acervo_id
                                 Filter: ((d.datas >= '1970-01-01'::date) AND (d.datas <=
'1979-12-31'::date))
                           ->  Index Scan using acervo_pkey on public.acervo a  (cost=0.14..8.16
rows=1 width=222)
                                 Output: a.id, a.tema, a.instituicao
                                 Index Cond: (a.id = d.acervo_id)
                     ->  Index Only Scan using pesquisador_acervo_pkey on
public.pesquisador_acervo pa  (cost=0.15..6.02 rows=4 width=82)
                           Output: pa.pesquisador_matricula, pa.acervo_id
                           Index Cond: (pa.acervo_id = a.id)
               ->  Index Scan using pesquisador_pkey on public.pesquisador p  (cost=0.14..0.23
rows=1 width=196)
                     Output: p.nome, p.afiliacao, p.matricula
                     Index Cond: ((p.matricula)::text = (pa.pesquisador_matricula)::text)
```

REFERÊNCIAS

EXPLAIN: Description. In: PostgreSQL 8.1.23 Documentation. 8.1.23. ed. [S. l.]: The PostgreSQL Global Development Group, [2012?]. Disponível em: https://www.postgresql.org/docs/8.1/sql-explain.html. Acesso em: 27 jun. 2024.

EXTENDING SQL: Query Language (SQL) FunctionsPostgreSQL 16.3 Documentation. In: PostgreSQL 16.3 Documentation: The PostgreSQL Global Development Group. [S. l.]: The PostgreSQL Global Development Group, [2024?]. Disponível em: https://www.postgresql.org/files/documentation/pdf/16/postgresql-16-A4.pdf. Acesso em: 20 jun. 2024.

INDEXES: Index Types. [S. l.]: PostgreSQL, 8 fev. 2024. Disponível em: https://www.postgresql.org/docs/current/indexes-types.html. Acesso em: 1 maio 2024.

POSTGRESQL 16.3 Documentation: The PostgreSQL Global Development Group. 16.3. [S. l.]: The PostgreSQL Global Development Group, [2024?]. Disponível em: https://www.postgresql.org/files/documentation/pdf/16/postgresql-16-A4.pdf. Acesso em: 25 jun. 2024.

USING Views: View Syntax. In: MySQL 8.4 Reference Manual: Including MySQL NDB Cluster 8.4. 8.4. ed. [S. l.]: Oracle, 30 abr. 2024. Disponível em: https://downloads.mysql.com/docs/refman-8.4-en.a4.pdf. Acesso em: 15 jun. 2024.