UNIVERSIDADE DE SÃO PAULO
Instituto de Ciências Matemáticas e de Computação

Desenvolvimento de uma Técnica para Estender um SGBD Relacional com Consultas por Similaridade

Marcos Roberto Nesso Junior

Dissertação de Mestrado do Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional (PPG-CCMC)

ICMC USP
SÃO CARLOS

Marcos Roberto Nesso Junior

Desenvolvimento de uma Técnica para Estender um SGBD Relacional com Consultas por Similaridade

Dissertação apresentada ao Instituto de Ciências Matemáticas e de Computação – ICMC-USP, como parte dos requisitos para obtenção do título de Mestre em Ciências – Ciências de Computação e Matemática Computacional. *EXEMPLAR DE DEFESA*

Área de Concentração: Ciências de Computação e Matemática Computacional

Orientador: Prof. Dr. Caetano Traina Junior

USP – São Carlos
Julho de 2019

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados inseridos pelo(a) autor(a)

N467d

Nesso Junior, Marcos Roberto
Desenvolvimento de uma Técnica para Estender um SGBD Relacional com Consultas por Similaridade / Marcos Roberto Nesso Junior; orientador Caetano Traina Júnior. -- São Carlos, 2019.
75 p.

Dissertação (Mestrado - Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional) -- Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, 2019.

1. Método de Acesso Métrico. 2. Sistemas de Gerenciamento de Bases de Dados. 3. PostgreSQL. 4. Estruturas de Indexação. 5. Slim-tree. I. Traina Júnior, Caetano, orient. II. Título.

Bibliotecários responsáveis pela estrutura de catalogação da publicação de acordo com a AACR2:
Gláucia Maria Saia Cristianini - CRB - 8/4938
Juliana de Souza Moraes - CRB - 8/6176

Marcos Roberto Nesso Junior

Development of a Technique for Extending a Relational DBMS with Similarity Queries

Dissertation submitted to the Institute of Mathematics and Computer Sciences – ICMC-USP – in accordance with the requirements of the Computer and Mathematical Sciences Graduate Program, for the degree of Master in Science. *EXAMINATION BOARD PRESENTATION COPY*

Concentration Area: Computer Science and Computational Mathematics

Advisor: Prof. Dr. Caetano Traina Junior

USP – São Carlos
July 2019

RESUMO

NESSO-JR, M. R. **Desenvolvimento de uma Técnica para Estender um SGBD Relacional com Consultas por Similaridade**. 2019. 75 p. Dissertação (Mestrado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2019.

Os Sistemas de Gerenciamento de Bases de Dados (SGBD) baseados na Teoria Relacional foram desenvolvidos para atender às necessidades do armazenamento e recuperação de grandes volumes de dados. Esses dados são representados por valores numéricos, datas e/ou pequenas cadeias de caracteres, e são chamados genericamente "dados escalares". Com a evolução da tecnologia da informação, torna-se cada vez mais necessário organizar, armazenar e recuperar também outros tipos de dados, a que neste trabalho chamamos de "dados complexos", tais como imagens, vídeo, séries temporais e sequências genéticas. As comparações baseadas em Relações de Identidade (RI) ou em Relações de Ordem (RO) são úteis para consultas sobre dados escalares, porém não são adequadas para dados complexos. Para estes, as consultas por similaridade têm sido a opção mais estudada, embora a sua disponibilidade nos SGBDs disponíveis ainda seja limitada. Métodos de Acesso Métrico (MAMs) são usualmente aplicados para a indexação de dados complexos, de modo a agilizar a execução de consultas por similaridade. O presente trabalho de mestrado visou incorporar recursos de um MAM a um SGBD Relacional. Isso foi feito por meio da proposta e implementação de uma técnica para estender um SGBD Relacional de grande utilização. Assim, implementou-se o MAM conhecido como *Slim-Tree* no *PostgreSQL*, que é um SGBD Relacional. A implementação da técnica resultou no RAFIKI, um protótipo capaz de superar a sua antecessora KIARA em termos de velocidade, quando usado para realizar consultas por similaridade. A análise experimental realizada mostrou que o RAFIKI é até 6 vezes mais rápido que a KIARA. Utilizando a técnica proposta, é possível a extensão do PostgreSQL para dar suporte a outros MAMs.

Palavras-chave: Método de Acesso Métrico, Sistemas de Gerenciamento de Bases de Dados, PostgreSQL, Estruturas de Indexação, Slim-tree.

ABSTRACT

NESSO-JR, M. R. **Development of a Technique for Extending a Relational DBMS with Similarity Queries**. 2019. 75 p. Dissertação (Mestrado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2019.

Database Management Systems (DBMS) based on the Relational Theory are designed to meet the needs of storing and retrieving large amounts of data. These data can be represented by numeric values, dates, and/or small strings, and are generically called "scalar data". With the evolution of information technology, it is increasingly necessary to organize, store, and retrieve other types of data. In this work, we call such data "complex data", such as images, videos, time series, and genetic sequences. Comparisons based on Identity Relations or Order Relations are useful for querying scalar data but are not suitable for complex data. For complex data, similarity queries have been the most studied option, although their availability in existing DBMS is still limited. Metric Access Methods (MAMs) usually are applied for indexing complex data to speed-up similarity queries. This Master's project aimed at incorporating MAM resources to a Relational DBMS, by proposing and implementing a technique for extending a widely used Relational DBMS. Thus, we implemented the existing Slim-Tree MAM into PostgreSQL, which is a Relational DBMS. This implementation resulted in RAFIKI, a prototype capable of outperforming its predecessor system KIARA, in terms of speed, in the task of performing similarity queries. The experimental analysis carried showed that RAFIKI is up to 6 times faster than KIARA. Further, using the proposed technique, it is possible to extend PostgreSQL to support other MAMs.

Keywords: Metric Access Method, Relational Database Management System, PostgreSQL, Indexing Structures, Slim-tree.

LISTA DE ILUSTRAÇÕES

LISTA DE CÓDIGOS-FONTE

LISTA DE ABREVIATURAS E SIGLAS

FMI-SiR	User-Defined **F**eatures, **M**etrics and **I**ndexes for **Si**milarity **R**etrieval - características, métricas e índices definidos pelo usuário para recuperação por similaridade
SIREN	**SI**milarity **R**etrieval **EN**gine
CBIR	*Content-Based Image Retrieval*
DDL	*Data Definition Language*
MA	Método de Acesso
MAM	Métodos de Acesso Métrico
RI	Relações de Identidade
RO	Relações de Ordem
SAM	Métodos de Acesso Espaciais
SGBD	Sistemas de Gerenciamento de Bases de Dados
SGBDR	Sistemas de Gerenciamento de Bases de Dados Relacional
SQL	*Structure Query Language*

SUMÁRIO

CAPÍTULO

1

INTRODUÇÃO

O mundo globalizado se encontra em um cenário de intenso compartilhamento da informação por meio da Internet, e a perspectiva é de que mais e mais dados sejam coletados, armazenados e trocados entre as pessoas e as organizações. Cada vez mais as pessoas estão conectadas e consumindo todo o tipo de conhecimento que é freneticamente gerado por diversos canais. Existem inúmeros provedores de conteúdo, serviços digitais e canais de jornalismo. Vivemos uma era onde a busca por conhecimento está a um clique de distância. [1]

Na Internet encontramos informações geradas por diversos meios, alguns podendo não ser confiáveis, como é o caso das redes sociais, que proporcionam a possibilidade das pessoas compartilharem diversos tipos de conteúdos. Um exemplo é o *Youtube*, que possui mais de um bilhão de usuários que geram bilhões de horas de vídeo. Nessa rede é possível navegar em 76 idiomas diferentes (representando 95% dos usuários da Internet). O *Instagram*, outra rede de compartilhamento de conteúdo, possui mais de 700 milhões de contas, onde pessoas compartilham vídeos, fotos e mensagens. A *Social Blade* é uma comunidade que reúne números sobre as principais redes sociais e seus usuários, como o Youtube, Instagram e Twitter (BLADE, 2017).

Com as pessoas produzindo mais dados, aumenta-se tanto a quantidade de dados, quanto a gama de possibilidades para os sistemas de análise que operam sobre esses dados. O principal ponto a destacar é a imensa quantidade de dados que existe, e é uma fonte de conhecimento inimaginável. A partir desta constatação, surgem questionamentos, tais como: "*Como gerenciar e tirar proveito de todos esses dados*?" "*Como organizar todos esses dados para que a informação que eles representam possa ser efetivamente utilizada*?" Essa última questão é o grande foco deste trabalho. Para isso, existem diversos SGBDs que, de alguma forma, tentam organizar esse conteúdo.

Em determinados domínios, os SGBDs disponíveis conseguem oferecer o suporte ne-

[1] http://www.internetlivestats.com/internet-users/

cessário para a manipulação dos dados. Já em domínios que manipulam dados mais elaborados, tais como os provindos de áudios, fotos ou vídeos, por exemplo, os SGBDs ainda não estão adequadamente preparados. Uma forma apropriada para solucionar esse obstáculo pode ser a utilização de um Método de Acesso (MA) específicos para dados complexos, que atendem às necessidades de consultas por similaridade.

O estado da arte tem mostrado que as consultas mais adequadas para a recuperação de dados complexos (tais como imagens, vídeos e áudios), divulgados por meio de diversos trabalhos, são as consultas por similaridade. Os tipos de dados passíveis de serem comparados por similaridade são chamados de Dados Complexos. Métodos de Acesso Métrico (MAM) consistem de estruturas de indexação para dados complexos, que podem ser utilizadas para estruturar os dados de forma a apoiar a execução eficiente de consultas. Na literatura há diversos desses métodos, mas no contexto deste trabalho foi abordado um MAM e um Sistemas de Gerenciamento de Bases de Dados Relacional (SGBDR) em específico: o MAM Slim-Tree e o SGBDR PostgreSQL. A escolha foi baseada na popularidade do SGBD Relacional PostgreSQL, o qual pode ser estendido, pois o mesmo é *open-source*. A MAM Slim-Tree foi proposta e vem sido estudada ao longo dos anos, pelo grupo de pesquisa GBdI, ao qual este trabalho de mestrado está inserido. Além disso, seu código-fonte é mantido em um repositório aberto.

A tarefa de inclusão de um MAM em um SGBDR não é trivial, uma vez que o mesmo deve ser incluído em uma camada intermediária do SGBDR. Também é necessário que o SGBDR faça a leitura de dados, realize tarefas de extração de características e cálculos de distância, além de indexar os dados de maneira persistente. Combinado a tudo isso, deve ser definida uma sintaxe que estenda a *Structure Query Language* (SQL) para permitir a expressão de consultas por similaridade.

Este trabalho implementa o método de acesso métrico Slim-Tree no PostgreSQL, expandindo a ferramenta de consulta KIARA (OLIVEIRA *et al.*, 2016), a qual modifica o SGBDR PostgreSQL para que ele possa interpretar e executar uma versão estendida de SQL, para incluir recursos que possibilitam expressar consultas por similaridade. Atualmente o KIARA suporta algumas funções de distância e possui alguns extratores de características. O código criado neste trabalho mescla as linguagens C e C++.

Após a integração da Slim-Tree no PostgreSQL, uma bateria de testes foram realizadas com o intuito de validar a implementação por meio da avaliação dos resultados esperados. A partir da validação, foi possível aplicar essa modificação a um cenário baseado em um conjunto de dados real, em que o uso de consultas por similaridade no PostgreSQL pudesse ser realizado, e assim avaliar o seu desempenho. Essa aplicação foi realizada no contexto médico onde, a partir das imagem de entrada, são realizadas consultas por similaridade, e verificou-se que dessa forma é possível trazer diversas informações para o usuário, com um desempenho satisfatório. Esses experimentos permitiram validar o desempenho da solução proposta neste trabalho, pela comparação do KIARA com e sem a utilização de um método de acesso métrico para auxiliar as

consultas por similaridade.

1.1 Organização do trabalho

O restante desta monografia possui a seguinte organização tradicional: o Capítulo 2 apresenta os conceitos fundamentais, o Capítulo 3 discute os trabalhos relacionados, o Capítulo 4 apresenta o método proposto, o Capítulo 5 descreve os experimentos realizados, bem como a avaliação dos resultados, e por fim o Capítulo 6 conclui este trabalho.

CAPÍTULO

2

CONCEITOS FUNDAMENTAIS

Para o melhor entendimento deste trabalho de mestrado, neste capítulo são apresentados os principais conceitos envolvendo a definição dos dados complexos, e como é possível definir consultas baseadas nesses elementos. Assim, o capítulo descreve os sistemas de recuperação de imagens por conteúdo, os quais manipulam os dados complexos por meio de outros recursos, como funções de distância e Métodos de Acesso. Por fim o capítulo aborda consultas por similaridade sobre os dados complexos e Métodos de Acesso Métricos, que podem melhorar o desempenho das consultas, e está dentro do foco de estudo deste trabalho de mestrado. Tendo o embasamento destas definições, é possível compreender este trabalho de mestrado e sua contribuição para o estado da arte.

2.1 Dados Complexos

Conforme já destacado na Introdução, os Sistemas Gerenciadores de Banco de Dados (SGBDs) são ferramentas que auxiliam o armazenamento dos dados de forma eficiente. Com o passar dos anos, os SGBDs e as aplicações que utilizam esta tecnologia evoluíram. Agora, esses programas não lidam apenas com dados tradicionais, ou dados convencionais (pequenas cadeias de caracteres ou números). Também lidam com os dados complexos (informação que não pode ser comparada por operadores cuja definição é única, universal, mas que podem ser parametrizados e são executados por funções cuja definição varia dependendo de especificidades dos domínios de dados e do objetivo de cada consulta). Para exemplificar, temos como exemplos de dados complexos imagens, vídeos e áudios cujos operadores de comparação variam de acordo com os aspectos que se quer comparar, no caso das imagens, pode-se avaliar: cores, textura ou formas. Para o contexto deste trabalho de mestrado, o principal dado complexo utilizado foi o tipo imagem, e em particular foi realizado um estudo de caso para tratar as imagens proveniente de exames médicos de tomografias computadorizadas. No entanto o método aqui apresentado não é limitado para um tipo de dado complexo.

Os operadores de comparação utilizados sobre dados escalares não são adequados para a comparação entre dados complexos. Desta forma, há a necessidade do uso de outros operadores para representação e comparação de dados complexos, considerando seu conteúdo. Essa temática é descrita em detalhes na próxima seção.

2.2 Recuperação de Imagens por Conteúdo

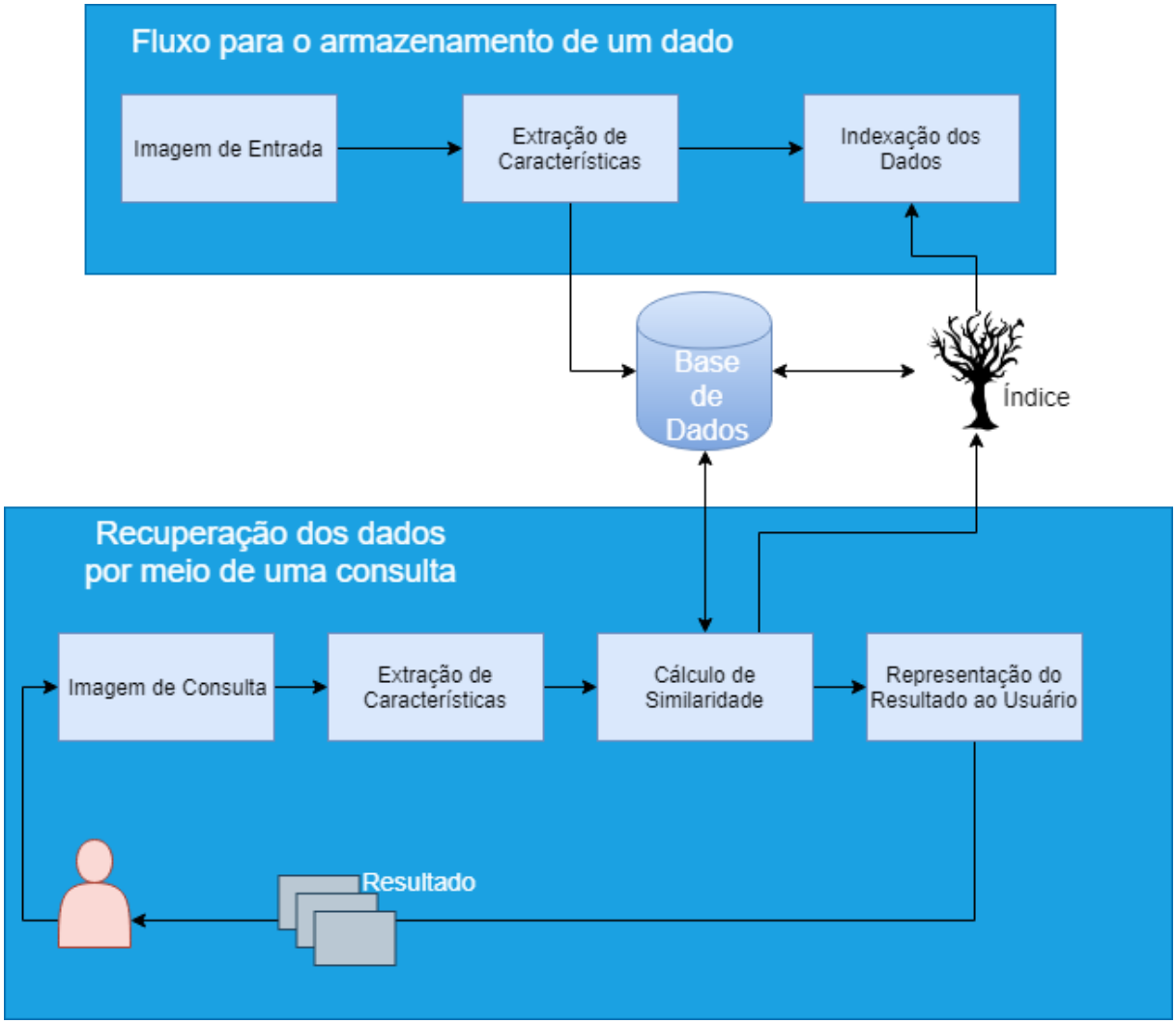
A manipulação de dados complexos é feita por meio de diferentes operações, realizadas sobre os dados. As operações de recuperação sobre dados complexos, tanto podem usar a informação presente no próprio objeto complexo (recuperação por conteúdo) ou a partir do conteúdo associado ao dado (recuperação por descrição). Ao utilizar a técnica de recuperação por descrição (em geral baseada em atributos textuais) para realizar consultas, algumas dificuldades foram notadas na literatura. Isso ocorre pois o trabalho de classificar ou atribuir informações às imagens é realizado por pessoas, o que pode gerar diferentes interpretações com relação ao conteúdo do objeto complexo. Tais dificuldades impulsionaram a busca de outras formas para realizar as recuperações das imagens, e desenvolveram-se as técnicas de busca realizada a partir do conteúdo do dado complexo.

A Recuperação de Imagens baseada em Conteúdo (*Content-Based Image Retrieval* (CBIR)) tem como principal objetivo extrair características (informações), de forma automatizada, de maneira que a informação extraída represente e caracterize a imagem adequadamente. Na literatura são encontrados diversos sistemas que realizam a recuperação de imagens analisando seu conteúdo. Na maioria destes sistemas, algumas etapas são fundamentais. Entre as principais etapas encontramos: a extração de características, para a representação da imagem; o cálculo de similaridade (aplicando uma função de distância); o armazenamento e a indexação dos dados (BLANCO *et al.*, 2016).

Na Figura 1 pode-se visualizar o fluxo de processamento adotado para um sistema CBIR na literatura, para o armazenamento do dado em uma base de dados:

- **Recebimento da Imagem (Armazenamento).** Nesta etapa o sistema recebe uma coleção de imagens, as quais serão processadas nas próximas etapas e serão armazenadas, junto de suas informações relacionadas, em uma base de dados.
- **Extração das Características (Algoritmo).** Nesta etapa as imagens passam pelo processo de extração de características, as quais são processadas por um ou mais algoritmos que buscam representar a imagem. As características devem ser representativas e discriminativas, e são armazenadas como vetores de características.
- **Indexação dos Dados.** Com as imagens já processadas e representadas como vetores de características, tais dados podem ser indexados, para auxiliar na recuperação e manipulação das informações de maneira mais rápida.

Figura 1 – Visão Geral do Fluxo de um Sistema CBIR



Fonte: Elaborada pelo autor.

Após as etapas anteriores, o sistema permite realizar a recuperação por conteúdo seguindo o seguinte fluxo de processamento:

- **Recebimento da Imagem de Consulta.** Esta etapa consiste em receber um objeto de consulta, o qual deve ser processados nas próximas etapas para ser utilizado nas buscas.

- **Extração das Características da imagem recebida.** Esta etapa consiste em representar a imagem de consulta, extraindo suas características da mesma maneira que para as imagens armazenadas.

- **Cálculo de similaridade (função de distância).** Nesta etapa, o objeto de consulta é utilizado para satisfazer a condição informada pelo usuário. Tal condição deve selecionar outras imagens de acordo com a similaridade calculada sobre às características extraídas.

- **Apresentação do resultado ao usuário.** Após a busca dos elementos que satisfação a consulta, a resposta deve ser organizada e entregue ao usuário do sistema.

Apesar do avanço realizado pela comunidade científica, a área de CBIR ainda possui um campo extenso para pesquisas. Um dos principais problemas enfrentados pelos pesquisadores é a representação semântica das características extraídas automaticamente (SHARIF *et al.*, 2019). Ainda hoje, o olho humano possui um poder muito maior do que os sistemas digitais para discernir o significado de uma imagem. E para avançar neste sentido, os pesquisadores estão se concentrando na busca de melhores formas de representação de características (BUGATTI *et al.*, 2014). Além disso, pesquisas atuais têm como foco buscar a melhor forma de calcular a similaridade entre dois objetos a partir das características extraídas, e como realizar consultas por similaridade de forma eficiente. Promover a busca por similaridade de objetos complexos de maneira eficiente em um SGBD Relacional é um dos focos deste trabalho de mestrado.

2.3 Extração de Características

A etapa de extração de características de uma imagem é uma das mais importantes no que se refere a sistemas CBIR. Pois, dependendo da técnica utilizada nesta etapa, pode-se representar muito bem as características visuais, que de fato interessam ao usuário. Caso não se represente adequadamente a imagem de acordo com a necessidade, o sistema CBIR não atingirá o objetivo semântico do usuário (SANTOS *et al.*, 2015).

Na Figura 2 vemos exemplo clássico do problema de representação de imagens por meio de algoritmos. Na imagem, são apresentados exemplos de diferentes semânticas de similaridade. Ao passo que os cachorros são visualmente semelhantes às donas, em termos de elementos visuais das imagens, duas mulheres e dois cachorros podem ser considerados semanticamente distintos entre si. A representação dos dados de acordo com a semântica desejada é parte fundamental para a recuperação por conteúdo adequada. Esse exemplo evidencia que, mesmo com todos os esforços na procura de métodos que possam descrever matematicamente uma imagem e como acessar a similaridade entre duas imagens, ainda não foi desenvolvido um algoritmo que possua a mesma capacidade do olho humano para representar as características de uma imagem e fazer uma análise que dependa também da semântica envolvida. No entanto esses algoritmos estão em constante evolução, tendo sido realizadas inúmeras pesquisas nessa área, sendo que o estado da arte tem sido submetido a um grande avanço no decorrer dos anos.

Considerando o fato da importância da escolha correta de um extrator para representar a imagem, é necessário avaliar bem qual algoritmo será utilizado para realizar a extração de características. Além de utilizar algoritmos que representem a imagem de forma semântica ao extrair as características, é possível determinar qual é a área de interesse da imagem, podendo possuir um escopo global ou até mesmo possuir regiões de interesse (apenas pedaços da imagem são interessantes para a extração de características).

Figura 2 – Exemplo de similaridade entre diferentes imagens e conceitos semânticos.

(a) Exemplo 1. (b) Exemplo 2.

Fonte: BBDO (2017).

Os principais extratores de características encontrados na literatura podem ser classificados em 3 categorias: cor, textura e forma. Na literatura, é possível encontrar diferentes trabalhos que analisam as características extraídas baseadas nestes elementos, por exemplo os trabalhos (SALEMBIER; SIKORA, 2002; SANTOS *et al.*, 2015),

Antes da execução dos processos de extração de características, é interessante utilizar a técnica de segmentação da imagem, pois assim é possível focar nas regiões de interesse de cada imagem. Essa técnica auxilia na remoção de áreas que podem influenciar negativamente na utilização dessas características e que afetam, por exemplo, um algoritmo de aprendizado de máquina. Um domínio de aplicação no qual a aplicação da segmentação de imagens provê resultados interessantes é para o processamento de coleções de imagens médicas. Com essa técnica, há a possibilidade de especificar a área de interesse, e assim outras partes da imagem não afetam negativamente o resultado (DATTA *et al.*, 2008) (CAZZOLATO *et al.*, 2017).

Neste trabalho de mestrado foram utilizadas imagens segmentadas, no entanto, tais imagens foram geradas a partir de outra abordagem (mais detalhes são apresentados no Capítulo 5). Estas imagens segmentadas foram utilizadas tanto para a extração de características quanto como sendo informação complementar para outras consultas.

Uma vez que as imagens possuem seus respectivos vetores de características, um CBIR pode utilizar tal recurso para comparar os vetores por meio de funções de distância, e assim calcular a (dis)similaridade entre os elementos. Os assuntos relacionados ao espaço métrico e as funções de distância são abordados na próxima seção.

2.4 Espaço Métrico e Funções de Distância

Conforme já abordado na seção anterior, a etapa de extração de características procura representar o conteúdo da imagem em um vetor de características. Tal representação possibilita a aplicação de funções de distância sobre os elementos.

Um espaço métrico é definido como um par $\langle \mathbb{S}, d \rangle$, onde $\mathbb{S}$ é o conjunto de todos os elementos que atendem aos requisitos do espaço (o chamado domínio de dados), e $d : \mathbb{S} \times \mathbb{S} \rightarrow \mathbb{R}^+$ é uma função de distância, chamada também de métrica, que atende às seguintes propriedades (LIMA, 1993):

- **Identidade dos indiscerníveis**: $d(s_1, s_2) = 0 \Leftrightarrow s_1 = s_2$;
- **Não-negatividade**: $0 < d(s_1, s_2) < \infty, \forall s_1 \neq s_2$;
- **Simetria**: $d(s_1, s_2) = d(s_2, s_1)$; e
- **Desigualdade triangular**: $d(s_1, s_2) \leq d(s_1, s_3) + d(s_3, s_2)$, $\forall s_1, s_2, s_3 \in \mathbb{S}$.

Um conjunto de dados S é dito estar em um espaço métrico quando $S \subset \mathbb{S}$. A função de distância é usada para quantificar o quão similar são dois elementos, habilitando que se expressem consultas baseadas em similaridade. Por exemplo, quando os vetores de características são dimensionais, é comum usar uma função da família *Minkovsky*, tal como a distância Euclidiana ou *Manhattan*, as quais atendem às propriedades de uma métrica (WILSON; MARTINEZ, 1997). Se o vetor é uma sequência genérica, podem ser usadas métricas diferentes, tais como as métricas L_{edit} para dados textuais lineares, e métricas baseadas em programação dinâmica para estruturas hierárquicas tais como textos em XML (JAGADISH; MENDELZON; MILO, 1995; TEKLI *et al.*, 2011) e para séries temporais (PAPAPETROU *et al.*, 2011; KEOGH, 2002).

Neste trabalho, chamamos o par ⟨extrator de características, função de distância⟩ de "Descritor", seguindo a designação colocada pelo padrão de descrição de conteúdo multimídia definido pelo Grupo de Especialistas em Imagens com Movimento (Moving Picture Experts Group – MPEG) ISO/IEC (JTC1/SC29/WG11, 2000), o qual propõe um conjunto de descritores para imagens, áudios e vídeos. Extrair características dos objetos complexos para então compará-los não altera o conceito fundamental de que a função de comparação deve atender às propriedades de uma métrica. Assim, cada descritor forma um espaço métrico (TRAINA *et al.*, 2010; SAMET, 2006).

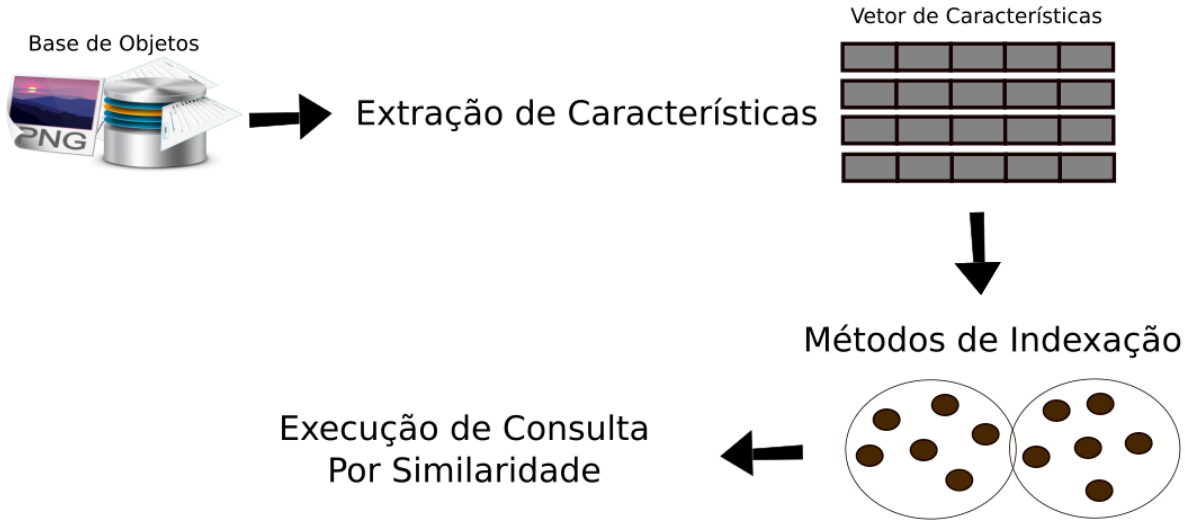
Definindo-se funções de distância adequadas sobre as características extraídas dos elementos de determinado domínio de dados complexos, os espaços métricos tornam-se excelente alternativa para executar consultas por similaridade com eficiência (POLA; TRAINA; TRAINA CAETANO, 2009; MALIK *et al.*, 2009). Para aplicações em Bases de Dados, a função de distância é considerada uma "caixa preta", geralmente definida por um especialista no domínio da aplicação. As características são extraídas dos objetos complexos e armazenadas junto a

eles, de maneira que executar uma comparação sempre corresponde a executar apenas a função de distância, mesmo que d corresponda a um descritor. Portanto, uma vez definida uma função de distância adequada que respeite as propriedades de uma métrica, os operadores de comparação por similaridade se aplicam a muitos tipos de dados complexos, incluindo dados multidimensionais (DOMENICONI *et al.*, 2007).

2.5 Consultas por Similaridade

A execução de uma consulta por similaridade é construída a partir de uma sequência de etapas, envolvendo um ou mais objetos. Esse processo pode ser estruturado da maneira mostrada na Figura 3, em que pode-se identificar a manutenção da base de dados, a extração de características e a utilização de métodos de indexação.

Figura 3 – Principais componentes encontrados em um sistema CBIR.



Fonte: Elaborada pelo autor.

A sequência de etapas é executada em um sistema CBIR: o primeiro passo corresponde a realizar a extração das características de um conjunto de objetos armazenados na base de dados. Essa extração é feita sobre os elementos do conjunto em que a consulta por similaridade será aplicada, usando os vetores de características obtidos (p.ex. de cor, forma, textura, etc. se for uma imagem). Nesta etapa, o conhecimento do especialista sobre o domínio dos dados é interessante e desejável, já que o mesmo poderá indicar um extrator adequado, ou seja, que represente o conjunto de dados de tal maneira que imagens consideradas mais semelhantes tenham características mais semelhantes e, ao mesmo tempo, tenham bom poder de identificação das imagens.

Após a extração das características e a representação dos objetos, é necessário definir qual é a função de distância que deverá ser aplicada para comparar as características, de forma que elementos similares possuam uma menor distância e fiquem mais próximos entre si. O inverso deve ocorrer com os objetos mais dissimilares, ou seja, quanto mais diferente for um objeto de outro, maior deve ser a distância entre eles.

As estruturas de índices servem para auxiliar a organização dos objetos, de maneira que se consiga acelerar a execução de uma busca tanto por similaridade, quanto por dissimilaridade. As estruturas tendem a reduzir mais rapidamente o espaço de busca, e esse é o motivo de se conseguir uma execução mais rápida em consultas por similaridade.

Com a base de dados organizada em uma estrutura de indexação e uma distância definidas, o processo de extração de características se repete para o objeto da consulta quando este é solicitado. Por fim, a consulta por similaridade é definida e executada na estrutura, utilizando a mesma função de distância. Os tipos de consultas mais comuns são: a busca por abrangência e a busca pelos k vizinhos mais próximos (BARIONI *et al.*, 2011).

A consulta por abrangência (*Range Query*) define uma distância máxima dos objetos no espaço métrico ao objeto central da consulta, usualmente chamada de raio da consulta. Todos os objetos que se encontram dentro desse raio serão retornados como parte do resultado da busca.

A consulta pelos k vizinhos mais próximos (*k-Nearest-Neighbor query*) define a quantidade k de elementos mais próximos que deverão ser retornados pela busca. Os objetos deverão ser ordenados de acordo com a sua distância ao centro de consulta e será retornada a quantidade k pedida.

2.6 Métodos de Indexação

Um Método de Acesso (MA) oferece um modo ágil de acesso aos registros de dados, não sendo afetado pelo posicionamento físico no arquivo, pois os índices são recursos adicionais que disponibilizam caminhos de acesso secundários aos dados. Os primeiros métodos de acesso foram propostos para operar sobre dados que possuem relação de ordem total, como a família de métodos de acesso B-tree. Os Métodos de Acesso tradicionais, implementados por SGBDs são adequados apenas para dados escalares, visto que como o seu funcionamento é baseado na relação de ordem total.

Para dados espaciais, encontram-se na literatura os Métodos de Acesso Espaciais (SAM), sendo comuns principalmente a *R-tree* e suas variações. Esse métodos têm sido incorporados aos SGBDs comerciais e já se encontram disponíveis no PostgreSQL. No entanto, essa estrutura foi desenvolvida para espaços de pequena dimensionalidade (como dados geográficos que usam somente duas ou três dimensões), e não se mostrou eficiente para dados de grande dimensionalidade (ATHANASIOU *et al.*, 2017).

Existem inúmeros tipos de dados usados nas aplicações modernas, como áudios, vídeos e imagens, que são representados através de vetores de características que podem incluir muitas dimensões. A manipulação desses tipos dados em um SGBD Relacional precisa de uma estrutura preparada para lidar com eles, e os mais indicados são os Métodos de Acesso Métrico (MAM). Essas estruturas são mais eficientes para realizar consultas por similaridade sobre dados com alta dimensionalidade, sendo um dos principais focos deste trabalho de mestrado (OLIVEIRA; JR.; KASTER, 2017). A seguir são explicados os MAMs M-Tree e Slim-Tree, que estão diretamente ligados a este projeto de mestrado.

2.6.1 M-tree e estruturas relacionadas

A *M-tree* é um Método de Acesso Métrico que foi proposto como uma organização dinâmica, onde há a possibilidade de inserção e remoção de objetos frequentemente. A *M-tree* é construída na forma *bottom-up*, ou seja, com a inclusão de elementos ela cresce de baixo para cima. Trata-se de uma árvore balanceada, fazendo com que haja o mesmo comprimento para todos os caminhos desde a raiz a todas as folhas da árvore.

Todos os elementos na *M-tree* são mantidos nos nós folhas, e são replicados nos nós diretório conforme necessário. Cada nó tem um elemento central (chamado representante), e cada elemento do nó tem a sua distância ao representante armazenada. A distância do representante ao elemento mais distante é o raio de cobertura do nó. Todos os nós possuem uma capacidade máxima C_{max}, um representante e um raio de cobertura. A partir da estrutura *M-tree*, diversas outras estruturas foram desenvolvidas com o intuito de aumentar a eficiência da árvore (ZEZULA *et al.*, 2006).

A primeira extensão da *M-tree* foi um algoritmo para a inserção de objetos por *bulk-loading*. Essa é uma técnica baseada na construção da árvore, em que realizam-se otimizações tanto do processo de construção da árvore quanto para o acesso posterior à árvore para realizar as consultas. Para a execução desse processo, é necessário já dispor de todos os elementos do conjunto de dados, pois esse método realiza uma análise da distribuição da carga completa dos dados e os pré-processa, obtendo como resultado a construção de uma *M-tree* mais eficiente (CIACCIA; PATELLA, 1998).

Outra extensão chamada *Multi-Way Insertion* foi um algoritmo proposto com a intenção de construir árvores mais compactas. A diferença entre o *bulk-loading* e a *Multi-Way Insertion* é que o *bulk-loading* assume a coleção de dados de forma estática, enquanto que o *multi-way* é capaz de lidar com dados que são acrescentados dinamicamente, deixando-o mais próximo do algoritmo de inserção original da *M-tree* (SKOPAL *et al.*, 2003).Experimentos que comparam as duas abordagens da *M-tree* mostram que o algoritmo *multi-way* requer 25% mais acessos a disco do que o *single-way* e 40% mais acesso que o algoritmo de *bulk-loading*. Entretanto, as árvores criadas utilizando o algoritmo *multi-way*, em média, realizaram consultas com 15% menos acessos a disco. Em resumo, a inserção por meio do algoritmo *multi-way* realiza uma

maior utilização dos nós, sendo assim vantajoso quando os custos da construção não são tão importantes quanto às execuções de consultas (SKOPAL *et al.*, 2003).

Outra abordagem da família da *M-tree* é a *Pivoting M-tree* (*PM-tree*) (SKOPAL, 2004). O objetivo dessa técnica é unir as regiões de cobertura de forma mais bem-ajustada, definindo regiões em anel. Em comparação com a *M-tree* original, os autores estudaram diversos parâmetros diferentes para a estrutura *PM-tree*, e a melhor combinação das variáveis indica que a *PM-tree* pode economizar de 15% a 35% dos acessos a disco quando comparada com a *M-tree*.

A proposta seguinte para a *M-tree* foi a M^+*-tree* (ZHOU *et al.*, 2003). Sua principal diferença é em relação à sua estratégia de particionamento. Em relação a *M-tree*, a M^+*-tree* obtém alguns ganhos de processamento em casos específicos, como quando o raio de pesquisa possui um alcance pequeno, tornando essa estrutura interessante. Outra variação encontrada é a M^2*-tree*, que visa a consulta por similaridade mais complexa, envolvendo diversos recursos como uma busca por uma *tag* e por um objeto de uma imagem, utilizando outro extrator de características, como um histograma de cor (ZEZULA *et al.*, 2006).

2.6.2 Slim-tree

O método de acesso métrico *Slim-Tree* foi inicialmente apresentado no artigo Traina-Jr. *et al.* (2000). Trata-se de uma estrutura com o objetivo de ser um método eficiente de acesso a dados em espaços métricos, tais como imagens. Ela visa obter a diminuição da sobreposição entre as sub-árvores da estrutura, sendo essa uma das vantagens da Slim-Tree em relação a *M-tree*, da qual a mesma deriva.

A *Slim-Tree* também é uma estrutura dinâmica, significando que ela aceita a inserção e exclusão de objetos sem a necessidade de reconstrução. Isso é importante, já que em seu uso em bases de dados, deve-se permitir que os dados sejam atualizados frequentemente, sendo essa uma característica necessária para lidar com aplicações em SGBDRs.

A *Slim-Tree* possui dois tipos de nós: folhas e índices. Todos os objetos são armazenados nos nós folha da árvore. Os nós índice apenas armazenam o roteamento das operações de busca. Todos os nós têm um elemento que atua como representante da sub-árvore correspondente, e a escolha desse elemento visa obter sub-árvores com a menor área de cobertura possível. Para a organização da árvore, é necessário utilizar uma função de distância métrica, que precisa obedecer às propriedades do espaço métrico, ou seja, desigualdade triangular, não-negatividade e simetria, para o funcionamento adequado da estrutura.

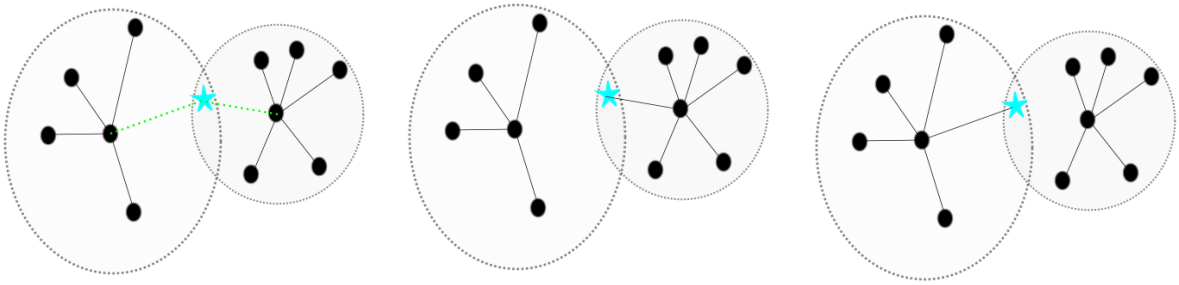
De maneira distinta da *B-tree* (mas da mesma maneira que na *R-tree*), é possível que mais de um nó se qualifique para armazenar um dado elemento. Assim, na inserção, o algoritmo procura o nó mais indicado para armazenar o novo elemento. Caso mais de um nó se qualifique para receber o elemento, é necessário utilizar uma política de escolha do nó destino, executada pelo algoritmo *ChooseSubtree*. Esse processo é repetido em cada nível da árvore, desde a raiz,

até encontrar a folha mais adequada.

Na Figura 4a vemos um exemplo em que dois nós se qualificam para receber o objeto. O algoritmo *ChooseSubtree* é governado por três opções de escolha: *random, mindist* e *minoccup*. Na opção *random*, o nó é escolhido arbitrariamente, sem nenhum critério. A opção *Mindist* escolhe o nó que possui a menor distância entre o novo elemento e o representante de cada nó, dentre os nós qualificados. A política *minoccup* escolhe o nó que possui a menor taxa de ocupação dentre os nós qualificados.

Na Figura 4b vemos a política *Mindist* para a escolha do nó em que o objeto será inserido. Essa política observa a menor distância entre o objeto, no caso a estrela em azul e o centro do nó. O nó que possuir a menor distância recebe o elemento. No exemplo da figura, o objeto é adicionado ao nó da direita.

Figura 4 – Política *ChooseSubtree*



(a) Inserção de um novo elemento, onde dois nós se qualificam.

(b) Escolha do nó responsável pelo novo objeto usando *Mindist*.

(c) Escolha do nó responsável pelo novo objeto usando *Minoccup*.

Fonte: Elaborada pelo autor.

Utilizando o *MinOccup*, o algoritmo se comporta como mostrado na Figura 4c, em que o objeto de consulta (estrela em azul) será adicionado ao nó à esquerda, pois é ele que possui uma menor quantidade de elementos. Essa é uma política que se destaca por sempre deixar os nós com uma maior taxa de ocupação. Ter os nós com maior taxa de ocupação nem sempre é positivo, pois isso também tende a causar maior sobreposição entre os nós da árvore, resultando em mais acessos aos níveis inferiores da estrutura durante a realização de buscas. Por outro lado, essa política tende a levar a árvore a ser mais enxuta (menor sobreposição).

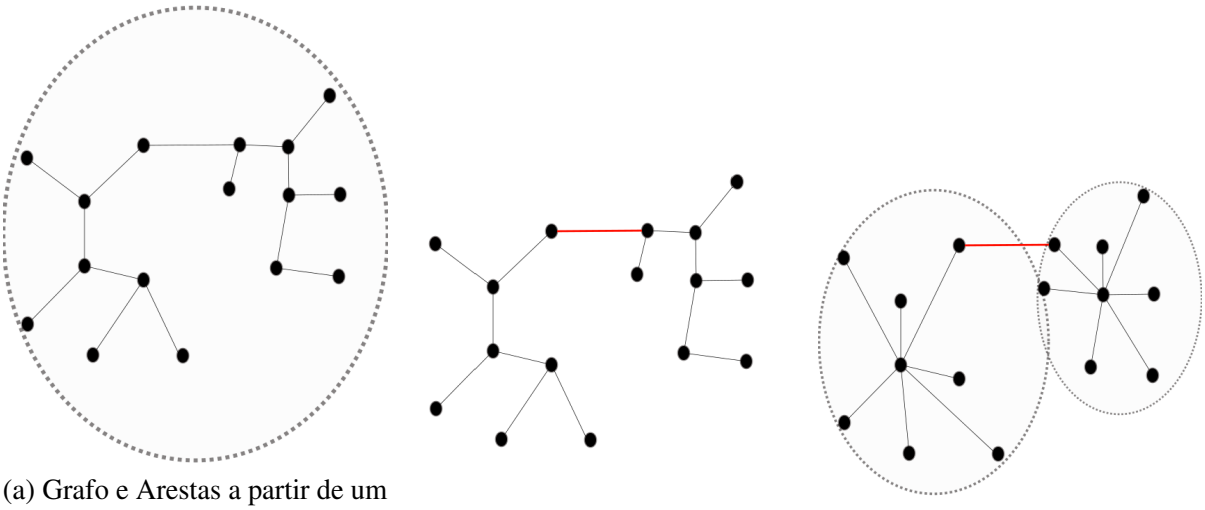
A política de escolha de nós deve ser especificada pelo especialista da aplicação, pois ele deve ser detentor do conhecimento necessário para realizar as escolhas que mais beneficiam a forma como a Slim-Tree irá se comportar.

A política de quebra de nós é outra característica importante de um MAM. Ela permite escolher uma maneira adequada para quebrar um nó quando se deve inserir um novo elemento em um nó que já está cheio, separando-o em dois. A Slim-Tree possui três algoritmos para isso: *Random, MinMax* e *Minimal Spanning Tree* (MST). A quebra de um nó é feita escolhendo-se

dois elementos dentre os existentes, junto com o elemento sendo inserido para serem os novos representantes dos dois nós que serão criados. A seguir, reparte-se os demais elementos entre entre os nós. A política *Random* escolhe aleatoriamente dois elementos para serem os representantes, e reparte os elementos atribuindo cada um ao nó que tem o representante mais próximo a ele. Já a política *minMax* considera todos os pares que minimizam os raios de cobertura, e escolhe aquele que leva ao um par de nós mais compacto. A política MST constrói um grafo completo (todos os nós têm arestas com todos os outros), em que os elementos são os nós, e as arestas têm como peso a distancia entre os dois nós que ela conecta. A seguir, o MST obtém a árvore de cobertura mínima associada (*Minimal Spanning Tree* – MST), e remove dela a aresta com maior peso. Procura-se, então, pelo objeto mais centralizado de cada componente conexa remanescente para ser o representante de cada nó, o qual armazenará também os demais elementos da componente conexa correspondente, buscando diminuir ao máximo o raio de cobertura dos nós resultantes.

Na Figura 5 vemos o processo da política de quebra de nós da política MST. Após inserir um elemento em um nó preenchido (Figura 5a), é construído o MST sobre os objetos e então a maior aresta é excluída (Figura 5b). Os elementos que ainda estão conectados formarão os dois novos nós, e cada grupo vai ter como nó representante aquele que é o mais central do grupo (Figura 5c).

Figura 5 – MST



(a) Grafo e Arestas a partir de um nó superlotado.

(b) Segunda fase.

(c) Terceira fase.

Fonte: Elaborada pelo autor.

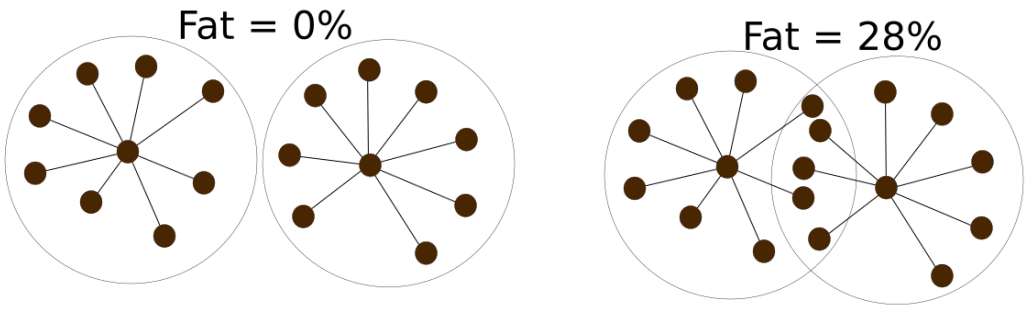
No artigo Traina-Jr. *et al.* (2000) é apresentado junto com a Slim-Tree o algoritmo *Slim-down*, que tem a função de diminuir o *overlap* (sobreposição) que pode ocorrer entre os nós de uma Slim-Tree. A tática desse algoritmo é procurar, em todos os nós, o elemento mais distante do representante, pois esse é o elemento que define o raio de cobertura. Caso ele seja coberto também por um outro nó onde ele não é o mais distante, é possível trocar esse elemento de nó, e assim reduzir o raio do nó onde ele estava, consequentemente reduzindo a sobreposição

entre as árvores.

Para medir a sobreposição que ocorre entre todas as sub-árvores de uma árvore, de maneira que seja possível comparar duas árvores e identificar aquela que apresenta a menor taxa de sobreposição global, utiliza-se o conceito de *fat-factor* (fator de gordura) da árvore (TRAINA-JR. *et al.*, 2000). Essa medida apresenta um valor entre 0 e 1, de maneira que quanto menor o valor, menor a sobreposição. Assim, um *fat-factor* de 0 corresponde a uma árvore ideal, sem sobreposição. Já o valor 1 corresponde à pior árvore possível, em que todos os nós se sobrepõem. A Figura 6 mostra dois exemplos de árvores, medindo seu *fat-factor*. No primeiro exemplo, mostrado na Figura 6a, tem-se o raio dos nós sem formar nenhuma intersecção entre os objetos, gerando um *fat-factor* de 0 (0%): a melhor situação possível em uma estrutura.

No segundo exemplo, mostrado na Figura 6b, observa-se que os nós apresentam intersecção, e 5 objetos são encontrados tanto dentro do raio do nó à esquerda quanto do raio à direita. Nesse exemplo, o *fat-factor* é de 0.28 (28%), o que tende a ocasionar buscas mais lentas em comparação com a busca na árvore do primeiro exemplo, pois, dependendo do caso, os dois nós terão que ser avaliados, obrigando assim a verificação de uma maior quantidade de elementos.

Figura 6 – Calculando o Fat-Factor.



(a) Fat-Factor com 0% de sobreposição. (b) Fat-Factor com 28% de sobreposição.

Fonte: Elaborada pelo autor.

Neste trabalho de mestrado, a Slim-Tree foi o MAM escolhido, para inclusão no SGBD PostgreSQL.

2.7 Considerações Finais

Neste capítulo foram apresentados os principais conceitos relacionados ao funcionamento de um sistema de recuperação de imagens por conteúdo. Dentre tais conceitos, foi explicado o processo de como os dados complexos são manipulados para a extração de características, permitindo representar os dados numericamente. Com a posse do vetor de características, é possível representar os dados no espaço métrico e assim aplicar recursos como as funções de

distância e os métodos de acesso para melhorar a representação e comparação dos dados na recuperação por similaridade.

No contexto deste trabalho de mestrado, os conceitos apresentados neste capítulo foram utilizados no método proposto, o qual permite a inclusão do MAM Slim-Tree no SGBD Relacional PostgreSQL, e que será apresentado com detalhes no Capítulo 4. Como resultado, obteve-se um sistema que permite a realização de consultas por similaridade utilizando os recursos de um MAM.

CAPÍTULO

3

TRABALHOS RELACIONADOS

O foco desta dissertação de mestrado é a inclusão de um Método de Acesso Métrico como estrutura de indexação para consultas por similaridade em um SGBDR. Neste capítulo serão abordados os principais trabalhos relacionados a este projeto. Primeiramente serão detalhados trabalhos que estendem o SGBD Relacional PostgreSQL com novos índices, uma vez que o PostgreSQL é o SGBDR utilizado neste trabalho de mestrado. Em seguida, serão apresentados os trabalhos que dão suporte a dados complexos e permitem realizar consultas por similaridade de diferentes formas.

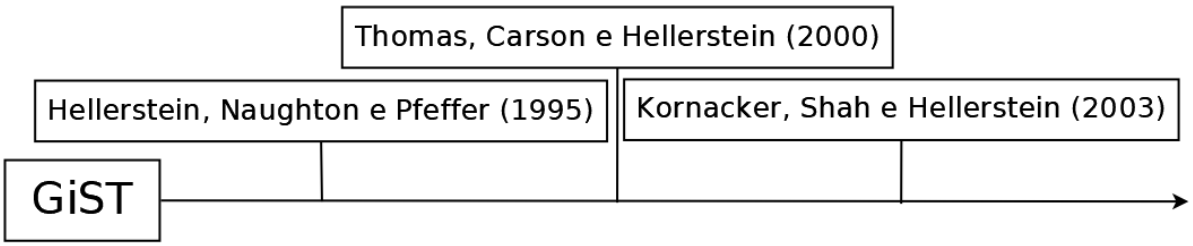
3.1 Trabalhos que utilizam a arquitetura disponibilizada pelo PostgreSQL

No manual da versão 9.6 do PostgreSQL são abordados os recursos que o SGBDR possui e como gerar as modificações necessárias para o usuário. Para o trabalho de mestrado, os capítulos 11 (Indexes), 59 *(Index Access Method Interface Definition)*, 61 *(GiST Indexes)*, e 62 *(SP-GiST Indexes)* foram objetos de estudos aprofundados para compreender o funcionamento de como um índice pode ser incluído dentro do PostgreSQL (GROUP, 2016).

O capítulo 61 do manual do PostgreSQL apresenta o conceito de *Generalized Search Tree (GiST)*, uma plataforma para a criação de métodos de acesso balanceados em estrutura de árvore, como por exemplo a própria *B-tree* e a *R-tree*. Na Figura 7 os principais artigos relacionados ao GiST são colocados em uma linha de tempo. Uma característica importante do GiST é que ele permite o desenvolvimento de estruturas de indexação baseadas em tipos de dados personalizados, criados pelo próprio usuário. Isso pode ser feito por um especialista no domínio do tipo de dados, em vez de um especialista em banco de dados. Cada método de indexação feito em GiST suporta as operações de índice padrão: *SEARCH, INSERT* e *DELETE*, de maneira adaptada para manipular as operações de busca para o formato dos dados em questão.

A distribuição do PostgreSQL inclui alguns índices implementados em GiST, como a *R-Tree* com funcionalidade para tipos de dados geométricos (GROUP, 2016),(KORNACKER; SHAH; HELLERSTEIN, 2003), (HELLERSTEIN; NAUGHTON; PFEFFER, 1995).

Figura 7 – Linha do Tempo GiST.



Fonte: Elaborada pelo autor.

Em Hellerstein, Naughton e Pfeffer (1995) foi apresentado o conceito do GiST (*Generalized index Search Tree*), uma estrutura extensível para dar apoio à criação de métodos de acesso. Nesse trabalho foi apresentada sua estrutura, propriedades e seu comportamento, utilizando como exemplo a construção de estruturas *B-tree* e *R-tree*. O artigo também mostra seus respectivos desempenhos, comparados a uma implementação tradicional. A adaptação de outras estruturas de árvores requer a alteração de alguns métodos no sistema de banco de dados, que encapsulam a estrutura e o comportamento da classe. O GiST pode ser usado para funcionar com muitas outras estruturas de árvores, como: *k-D-B-trees, V-trees e TV-trees*.

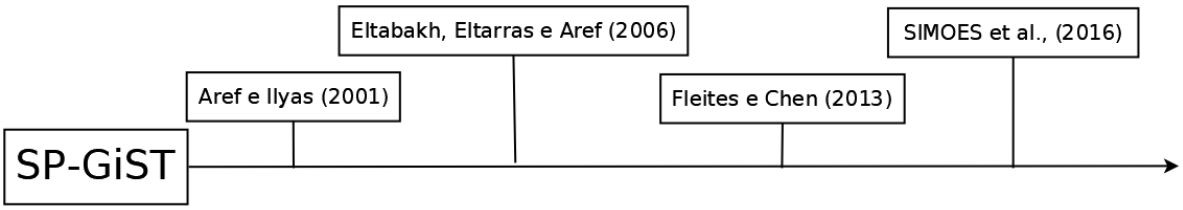
Em Thomas, Carson e Hellerstein (2000) é apresentada uma ferramenta chamada *Blobworld*, a qual utiliza o *amdb* (SHAH; KORNACKER; HELLERSTEIN, 1999), uma ferramenta para analisar métodos de acesso. *Blobworld* trabalha com a recuperação de imagens baseada em conteúdo. O artigo analisa três métodos de acesso multidimensionais que suportam consultas aos vizinhos mais próximos, bem como propõe algumas variantes para a *R-tree*.

O artigo Kornacker, Shah e Hellerstein (2003) apresenta o *amdb*, uma ferramenta para analisar métodos de acesso. Ele define métricas de desempenho para avaliar o acesso a páginas resultantes da execução de uma consulta. Na maioria dos casos, o *amdb* é independente da semântica dos dados, tornando-se universalmente aplicável a qualquer projeto para um Método de Acesso baseado em GiST. Outra contribuição do artigo é o detalhamento de uma ferramenta de visualização e animação, que permite a navegação e inspeção da estrutura da árvore e os dados contidos nos nós da árvore, ferramenta que já havia sido utilizada anteriormente em Thomas, Carson e Hellerstein (2000).

O Capítulo 62 do manual do PostgreSQL apresenta o conceito do *Space-Partitioned Generalized Search Tree* (SP-GiST), uma plataforma para o desenvolvimento de métodos de busca particionados em árvores, possibilitando o desenvolvimento de métodos de busca usando estruturas de dados não balanceadas, como as *Quad-trees*, *k-D*-trees e *Radix trees*. Os principais

artigos relacionados ao SP-GiST são mostrados em uma linha do tempo na Figura 8. O SP-GiST oferece uma interface com um alto nível de abstração, exigindo que o desenvolvedor do método de acesso implemente apenas métodos específicos para um certo tipo de dados.

Figura 8 – Linha do Tempo SP-GiST.



Fonte: Elaborada pelo autor.

O SP-GiST é apresentado no trabalho de Aref e Ilyas (2001). Esse conceito abre uma nova gama de possibilidades para a construção de índices que não poderiam ser introduzidos usando o GiST, uma vez que o SP-GiST possibilita trabalhar com árvores de particionamento de espaço que não tenham estrutura balanceada. Para ilustrar o seu funcionamento foram definidas algumas interfaces baseadas em parâmetros, como o número de partições do espaço que podem ser atendidas pelas estruturas.

Neste contexto, o trabalho Eltabakh, Eltarras e Aref (2006) mostra a eficiência de estruturas desenvolvidas com o SP-GiST em funcionamento dentro do PostgreSQL, e ilustra a importância da adição de outros métodos de acesso, utilizando como exemplo os métodos de acesso espaciais. Com essa nova gama de estruturas, aumenta-se a possibilidade de utilizar os métodos corretos para os domínios mais indicados a uma dada aplicação.

No trabalho Fleites e Chen (2013) é apresentado um sistema baseado no PostgreSQL e na estrutura de indexação *AH-Tree*. Esse sistema suporta Recuperação de Imagens por Conteúdo utilizando a *AH-Tree* para responder a consultas por similaridade. Esse trabalho é baseado na arquitetura GiST para a implementação do índice.

Foi também proposta uma extensão ao PostgreSQL para dados em domínio espaço-temporal, em que foi descrita a implementação de extensões para o Modelo Relacional envolvendo tipos de dados específicos, que não são devidamente atendidos pelo Modelo Relacional padrão. A primeira abordagem foi direcionada a dados georreferenciados, tendo como resultado o desenvolvimento de um protótipo chamado PostGIS-T (SIMOES *et al.*, 2016).

Todos os trabalhos citados nesta seção fazem uso do PostgreSQL e suas funcionalidades. No entanto, existem diversos sistemas que realizam consultas por similaridade em SGBDs. Na próxima seção, serão discutidas outras abordagens relacionadas a este trabalho de mestrado.

3.2 Trabalhos da literatura que dão suporte a dados complexos

Como descrito no capítulo anterior, *Content-Based Retrieval* é uma técnica que permite a seleção/filtragem de dados conforme o grau de dissimilaridade entre os elementos. Nesta técnica é realizado um processamento dos dados, gerando atributos que sumarizam os elementos originais de acordo com uma dada semântica. Os atributos podem ser obtidos por meio da extração de características, utilizando algoritmos especializados para esta tarefa. Este tipo de dado não é totalmente suportado pelos SGBDRs, sendo assim a sua manipulação limitada. Nas próximas seções são apresentados os trabalhos da literatura que suportam a manipulação de dados complexos e estão diretamente relacionados a esse trabalho de mestrado.

3.2.1 PostgreSQL-IE

O PostgreSQL-IE (GULIATO *et al.*, 2009) é uma extensão do PostgreSQL, tendo sido desenvolvida em C e PL/pgSQL. A ferramenta permite a extração automática de características de arquivos do tipo imagem, além de possuir recursos para criar novos extratores de características, novos métodos de acesso e novas estratégias para a elaboração de consultas que permitem combinar predicados baseados em similaridade com predicados tradicionais.

Esta extensão do PostgreSQL permite a inclusão de novas funções para a extração de características. As novas funções a serem desenvolvidas devem ser implementadas em C. Cada função de extração recebe uma ou mais imagens de entrada e retorna os respectivos vetores de características.

Essa abordagem permite o suporte a operações de similaridade sobre imagens. Incluindo funções através de bibliotecas compartilhadas, bem como SQL com palavras-chave relacionadas à similaridade. No entanto, o suporte de similaridade abrange apenas imagens. Em relação aos Métodos de Acesso Métricos, essa extensão não implementa nenhum, mas oferece suporte para uma futura integração.

3.2.2 SIREN

SImilarity **R**etrieval **EN**gine (SIREN) é o protótipo de uma ferramenta proposta por Barioni *et al.* em (BARIONI *et al.*, 2006; BARIONI *et al.*, 2009), a qual implementa a realização de consultas por similaridade em um SGBDR com base em uma extensão da linguagem SQL. A ferramenta foi construída para validar a proposta de extensão da linguagem SQL padrão. Ela é baseada em interceptar todo comando SQL enviado ao PostgreSQL, a fim de manipular as construções sintáticas relacionadas às operações de similaridade.

O SIREN possui basicamente 3 componentes principais. O primeiro componente é responsável por interpretar os comandos da sintaxe SQL, em que todos os comandos são

interceptados e analisados. Caso o comando não tenha operações por similaridade ou referências a um objeto complexo, ele é simplesmente enviado ao SGBDR e a resposta vai direto para a aplicação. O segundo componente é responsável por extrair as características do objeto complexo para sua representação e indexação. Por fim, o terceiro é o componente responsável pela utilização de um método de acesso métrico fora do SGBDR.

A abordagem (SIREN) carece do suporte a múltiplas transações, não podendo assim processar várias informações. Outra desvantagem é que o SIREN não permite algumas otimizações quando há predicados tanto de similaridade quanto tradicionais. Ele executa os predicados de similaridade primeiro, acessando separadamente um MAM que indexa o conjunto de dados e, em seguida, usa o SGBDR para os predicados tradicionais. Dessa forma, este protótipo enrijece o processo de otimização de consultas com predicados mistos.

O SIREN é um mecanismo que trabalha entre a camada de aplicação e o SGBDR, como mostra a Figura 9. Por meio da sua SQL orientada a similaridade, as consultas que usam operadores de similaridade são escritas de maneira semelhante às consultas usando a SQL padrão. Em nível de abstração, o conhecimento dos detalhes e restrições da implementação são removidos da responsabilidade do programador da aplicação. Assim, o SIREN atua como um *middleware* entre a camada do aplicativo e a camada do banco de dados, fornecendo um interpretador de consulta SQL estendida para o banco de dados SQL padrão, além de extratores de características, funções de distância e métodos de acesso métrico. Depois de processar os predicados de similaridade, a consulta é convertida para o formato da SQL padrão e processada pelo SGBDR.

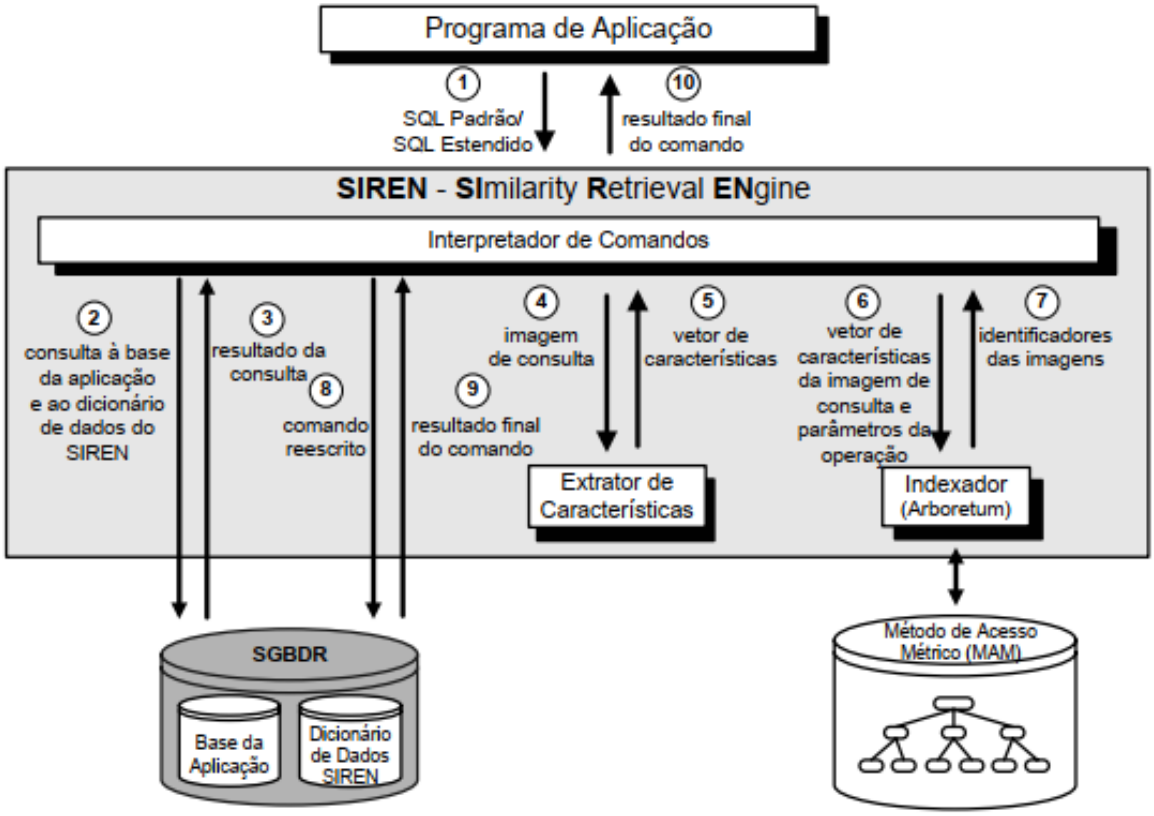
Para realizar consultas por similaridade sobre dados complexos, como imagens, o usuário deve criar uma métrica, a qual combina uma função de distância com um extrator de características, e, por fim associar a métrica recém criada ao atributo complexo.

3.2.3 FMI-SiR

A próxima ferramenta a ser discutida é o User-Defined **F**eatures, **M**etrics and **I**ndexes for **Si**milarity **R**etrieval - características, métricas e índices definidos pelo usuário para recuperação por similaridade (FMI-SiR), proposta por Kaster *et al.* em (KASTER *et al.*, 2010). O foco desta ferramenta é ser um subsistema do SGBDR Oracle para realizar CBIR. FMI-SiR fornece operadores, extratores de características, funções de distância e métodos de acesso. Ao implementar as premissas da arquitetura definida pelo Oracle, o FMI-SiR acaba sendo integrado similarmente às demais funcionalidades do Oracle. A ferramenta busca responder eficientemente a consultas por similaridade.

A manipulação dos dados começa com a extração de características, que gera um vetor de características e armazena a informação em um BLOB. O atributo que contém o vetor de características pode ser indexado pelo MAM Slim-Tree, que, por sua vez, utiliza a função

Figura 9 – Visão Geral do SIREN



Fonte: Barioni *et al.* (2009).

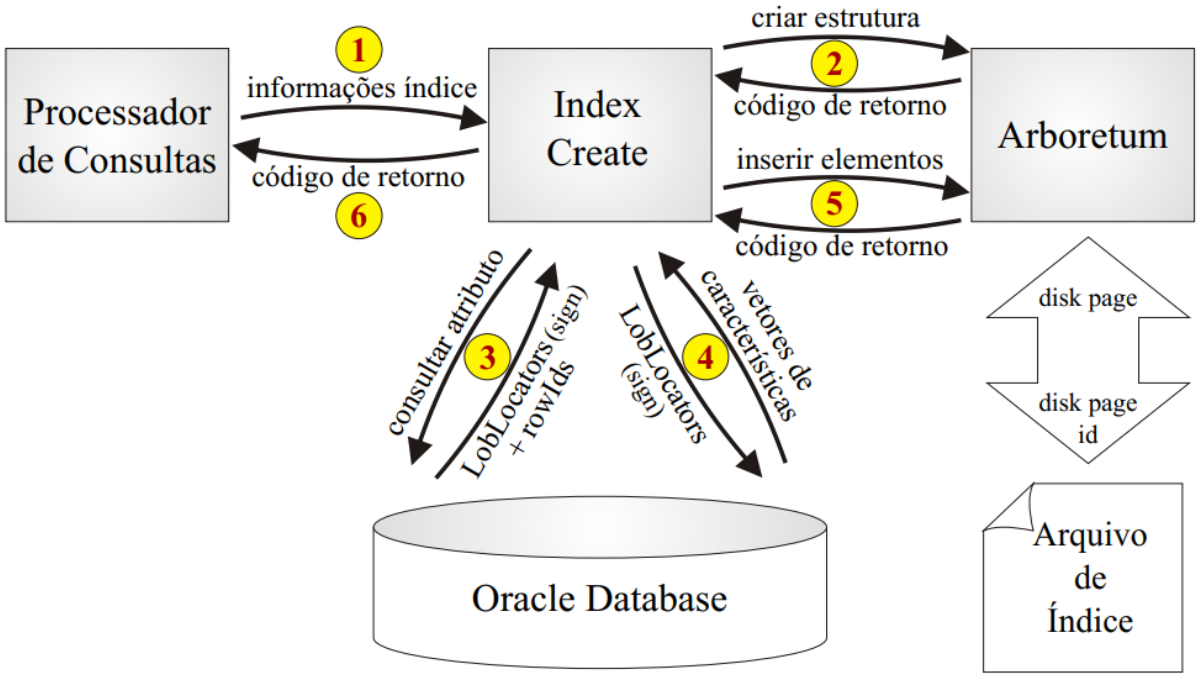
de distância associada. Com esses recursos, é possível realizar as principais consultas por similaridade usando a sintaxe SQL padrão.

Em resumo, FMI-SiR é um *framework* que opera sobre o SGBD Oracle, empregando funções definidas pelo usuário para extrair características e indexar os dados. No entanto, falta nesta ferramenta o suporte para diferentes tipos de dados. Ela não suporta a inclusão dinâmica de extratores de características e funções de distância. Com isso, faz-se necessário estender o código-fonte para implementar novos recursos e recompilar a ferramenta. Também é necessário executar comandos em *Data Definition Language* (DDL) para associar o novo recurso à implementação dentro do módulo. Em relação ao RAFIKI, o FMI-SiR armazena seus índices em arquivos externos ao SGBDR, como é visto na Figura 10, os quais são gerenciados pela biblioteca *Arboretum*, que implementa o acesso paginado aos dados e com controle de cache. No RAFIKI, o arquivo de índice é criado dentro da estrutura do PostgreSQL.

3.2.4 MSQL

MSQL é uma abordagem desenvolvida por pesquisadores da Microsoft na Ásia. No artigo (LU *et al.*, 2017) apresenta uma solução para realizar consultas por similaridade utilizando apenas a SQL. O MSQL permite executar consultas utilizando apenas os comandos *SELECT*-

Figura 10 – Visão Geral da construção de índice no FMI-SiR



Fonte: Kaster *et al.* (2010).

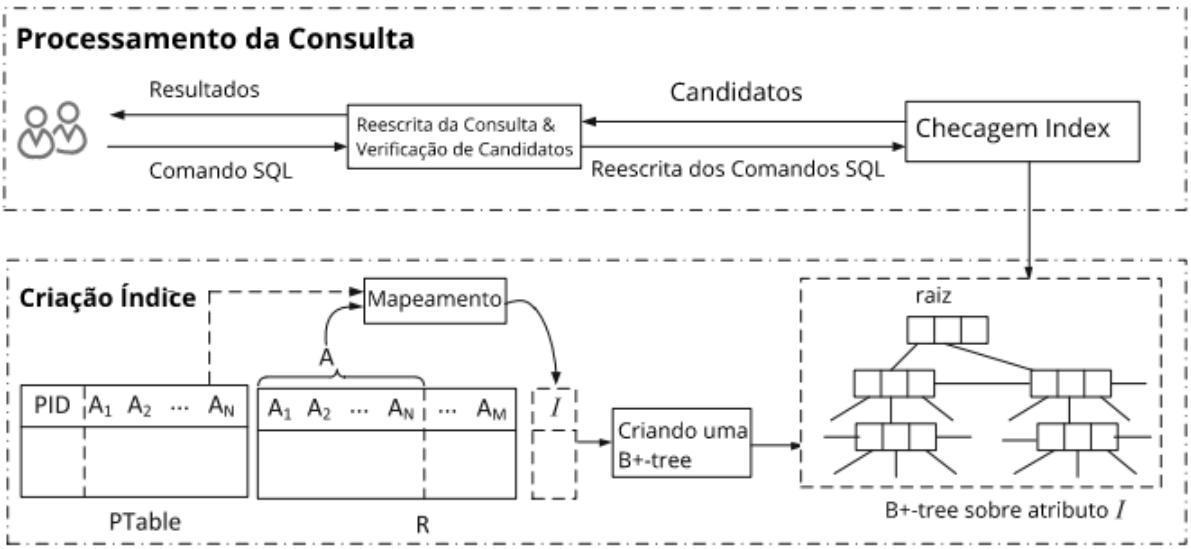
FROM-WHERE. Essa abordagem fornece um esquema de indexação uniforme, baseado no índice B^+-Tree. Isso permite acelerar as consultas por similaridade, usando a busca no índice, como visto na Figura 11. Esta solução armazena as distâncias pré-computadas de um elemento complexo para um elemento pivô, que está representado em uma nova coluna usando um tipo de dado bidimensional, no formato <pivoId, distância>. As consultas por similaridade ocorrem usando esta coluna, pois é possível remover os elementos que não são candidatos da consulta. Somente as tuplas cujos valores da coluna estão entre os limites inferior e superior da consulta são retornados.

A abordagem do MSQL não resolve o problema de inserção de diferentes MAMs em um SGBDR, ou um método de inserção. Ao contrário disso, resolve o problema da indexação por meio de uma nova técnica, na qual utiliza o pré-calculo da distância do elemento para um pivô. Além disso, a técnica apresenta novas heurísticas para a seleção de pivô e regras de poda.

3.2.5 KIARA

O principal trabalho relacionado a este trabalho de mestrado é o KIARA, proposto por Oliveira *et al.* em (OLIVEIRA *et al.*, 2016). Trata-se de uma extensão do SGBD relacional PostgreSQL, com suporte a uma extensão do SQL para realizar consultas por similaridade sobre dados complexos. O KIARA permite o gerenciamento de extratores de características e funções de distância, os quais são dinamicamente inseridos (sem recompilação), após compilar

Figura 11 – Processo de consulta no MSQL



Fonte: Lu *et al.* (2017).

uma vez a *Hermes* e a *Artemis* (bibliotecas em C contendo funções de distância e extratores de características) e não havendo alterações nas mesmas. O KIARA faz o uso de meta-tabelas para manter as informações dos extratores de características e das funções de distância associadas aos atributos dos dados complexos.

Para dar apoio à extensão da SQL, foi criado um *parser* que funciona como um *proxy* no núcleo do KIARA. Ele recebe as consultas e reescreve somente aquelas que contêm predicados de similaridade, expressando-os por meio de UDFs (User-Defined Functions) em SQL padrão. A partir disso, as consultas reescritas são reenviadas para o núcleo do KIARA (OLIVEIRA *et al.*, 2016).

Após a inserção de um novo extrator, os vetor de características são extraídos dos dados complexos e somente aí são armazenados. Consultas por similaridade podem ser incluídas através de funções PL/pgSQL, assim como novos índices. Além disso, essa extensão permite explorar os planos de consultas alternativos, envolvendo predicados tradicionais e por similaridade.

O RAFIKI foi construído estendendo o KIARA, o qual não possuía suporte a MAMs. Sendo assim, a estrutura construída neste trabalho de mestrado se aproveita dos recursos disponibilizados pelo KIARA. O RAFIKI é apresentado, com mais detalhes, no Capítulo 4.

3.3 Considerações Finais

Considerando todas as funcionalidades e opções nas diversas ferramentas descritas nesta seção, neste projeto de mestrado, optou-se por iniciar o método de como inserir um Método de Acesso Métrico em um SGBD Relacional a partir do KIARA. O mesmo dispõe de recursos

interessantes que poderiam ser aproveitados e assim trabalhar em conjunto com a indexação de dados complexos.

Neste capítulo foram apresentados os trabalhos relacionados a este trabalho de mestrado. Como descrito, não há, na literatura, uma proposta de como inserir um Método de Acesso Métrico em um SGBDR que permita usufruir de todos os recursos de otimização de consultas do SGBDR. Dentre os trabalhos apresentados, alguns não permitem a inserção de novos MAMs, enquanto outros o fazem de forma limitada, como por exemplo enrijecendo o processo de otimização de consultas com predicados por similaridade. O RAFIKI, por sua vez, é proposto para preencher essa lacuna.

O trabalho de (GULIATO *et al.*, 2009) não possui nenhum MAM inserido, mas possui suporte para a inserção. Em (BARIONI *et al.*, 2006; BARIONI *et al.*, 2009) a indexação é realizada sobre os dados fora do SGBDR. No trabalho de (KASTER *et al.*, 2010) o arquivo da MAM é mantido fora do SGBDR e não são dinamicamente inseridos. O MSQL (LU *et al.*, 2017) não permite inserir uma MAM em um SGBDR, mas sim utiliza-se uma B^+-tree para indexar os pivôs e assim realizar consultas por similaridade. O trabalho realizado em (OLIVEIRA *et al.*, 2016) cria todo um ambiente para o suporte a CBIR, com exceção do suporte a MAMs. O sistema proposto pelos autores foi usado como base para este projeto de mestrado. No próximo capítulo é apresentado o RAFIKI, que preenche essa lacuna e fornece um método de inserção de uma MAM no SGBDR PostgreSQL.

CAPÍTULO

4

MÉTODO PROPOSTO

O objetivo deste trabalho foi o desenvolvimento de uma técnica para estender um Sistema de Gerenciamento de Banco de Dados Relacional (SGBDR), para que ele permita a representação e a execução de consultas por similaridade. Visualizando a inclusão dos resultados obtidos em um SGBD, os experimentos foram executados em um ambiente controlado, usando em uma versão estável do Sistema Operacional CentOS e do PostgreSQL. Neste capítulo é apresentado o RAFIKI, o método proposto neste trabalho de mestrado para incluir consultas por similaridade entre as funcionalidades de um SGBDR, incluindo um Método de Acesso Métrico para agilizar as buscas por similaridade sobre objetos complexos.

4.1 Pré Requisitos para a inclusão do MAM no SGBDR

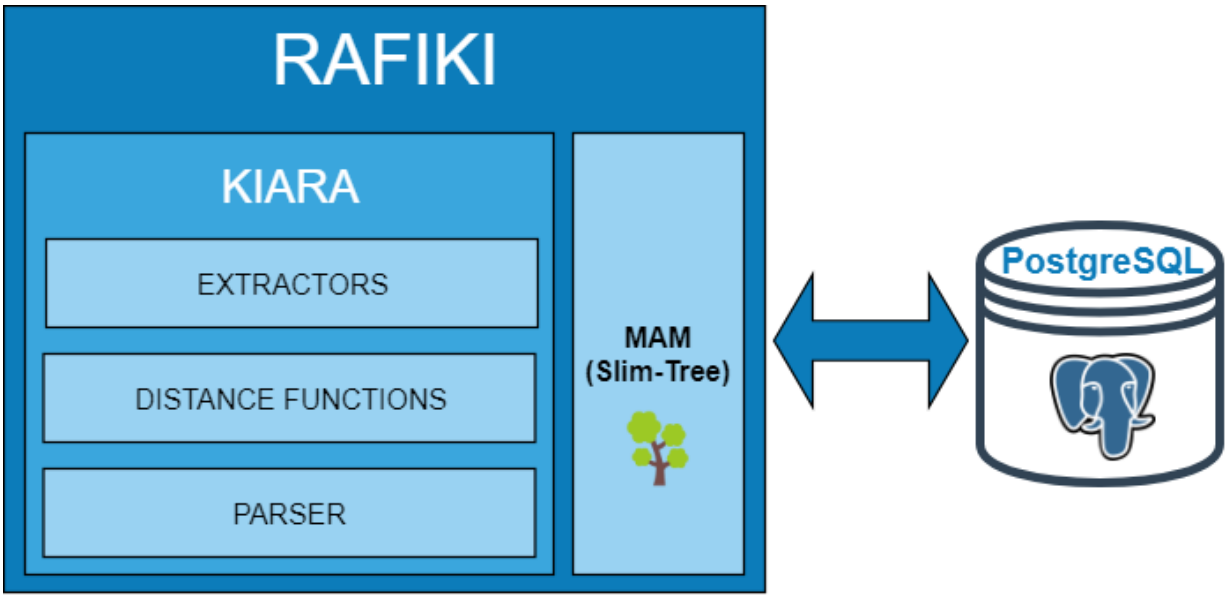
Nesta seção mostra-se como deve ser preparado o esquema de uma aplicação que deve incluir em uma tabela (ao menos) um atributo que armazena dados complexos e permite consultas por similaridade. Consultas por similaridade devem ser executadas preferencialmente usando um Método de Acesso Métrico. Para incluir um Método de Acesso Métrico (MAM) em um SGBDR é necessário criar a arquitetura de representação para dados complexos e também os recursos para armazenar e usar as respectivas características, pois somente assim, uma função de distância pode atuar sobre os mesmos e auxiliar a executar consultas por similaridade. O MAM deve usar as características já extraídas dos elementos complexos, as quais deverão estar armazenadas em tabelas do SGBDR. Além disso, a indexação utiliza uma definição previamente existente de como deve ser a interface para todas as funções de distância que possam ser utilizadas para a construção, manutenção e consulta na estrutura do MAM.

Conforme descrito na seção 3.2.5, um dos trabalhos do nosso grupo de pesquisa (GBdI) é o desenvolvimento da ferramenta KIARA. Essa ferramenta possui os recursos necessários para realizar a extração de características e comparação (por meio das funções de distância) de elementos complexos. KIARA usa a biblioteca Arboretum, que contém funções de distância

e extratores de características (já implementadas em linguagem C++), e que foram integradas ao SGBDR PostgreSQL. Essas funcionalidades se comunicam com o SGBDR utilizando um parser para a sintaxe do SQL estendido e as funções implementadas em C++. Considerando essas características, neste trabalho usou-se a implementação do KIARA como ponto de partida para a integração de um MAM ao PostgreSQL. Como o MAM a ser inserido ao PostgreSQL, foi escolhida a Slim-Tree, pois a mesma também encontra-se na biblioteca Arboretum.

A Figura 12 apresenta como o método proposto, RAFIKI, é composto. Ele utiliza as funcionalidades dos extratores e funções de distância já existentes na KIARA, e adiciona um módulo para o método de acesso métrico (Slim-Tree), usando o SGBDR PostgreSQL.

Figura 12 – Visão Geral do RAFIKI



Fonte: Elaborada pelo autor.

4.1.1 Inclusão da Slim-Tree no PostgreSQL

Esta seção descreve os passos realizados para a inclusão da Slim-Tree no PostgreSQL. Duas características necessárias para a disponibilização desta funcionalidade é que a Slim-Tree seja incluída de maneira a facilitar seu uso, e que permita utilizar os recursos fornecidos pelo SGBDR adequadamente.

4.1.2 Preparação dos dados para o uso do MAM

Antes da criação de um MAM é preciso criar uma tabela contendo os dados que serão trabalhados previamente armazenados. A estrutura básica da tabela deve conter a coluna de identificação e uma coluna para armazenar as assinaturas (cada um dos vetores de características) de cada objeto. Para isso, o RAFIKI utiliza os recursos do SGBDR dos comandos CREATE TABLE e INSERT, além de herdar do KIARA a função *image_import*, a qual transforma cada

imagem para o formato inteiro *bytea* do PostgreSQL, e a função de UPDATE, que consulta a coluna da imagem inserida, faz a extração das características e insere a assinatura obtida na coluna *blob_caracteristicas*. O Código 1 apresenta um exemplo de comando SQL para criação da tabela. O Código 2 mostra um exemplo de inserção de uma imagem na tabela criada, salvando seu *bytea*.

Código-fonte 1 – Criação da Tabela

```
1: CREATE TABLE tab_exemplo(
2:          id serial,
3:          blob_imagem bytea,
4:          blob_caracteristicas bytea,
5:          path text,
6:          CONSTRAINT primary_key UNIQUE(id)
7: );
```

Código-fonte 2 – Inserção na Tabela

```
1: INSERT INTO tab_exemplo(blob_imagem, path)
2:     VALUES(image_import($file), $file);
```

Após inserir a imagem na tabela, o Código 3 chama a função do extrator de característica escolhido pelo usuário, obtendo a assinatura da imagem e inserindo-a na tabela, no campo *blob_caracteristicas*. O exemplo apresentado usa o extrator Color Structure, por meio da função *color_structure_extractor*.

Código-fonte 3 – Atualização da Tabela

```
1: UPDATE tab_exemplo SET blob_caracteristicas =
      color_structure_extractor(blob_imagem);
```

Essa sequência de comandos SQL prepara os dados que serão utilizados na indexação pela Slim-Tree, conforme descrito na seção seguinte. As demais seções apresentam os comandos adicionados ao SGBDR, que são contribuições diretas deste projeto de mestrado.

4.1.3 Inclusão da Slim-Tree no PostgreSQL

Após a etapa de extração de características ter sido concluída, está pronto o ambiente mínimo necessário para a criação da Slim-Tree no PostgreSQL pelo RAFIKI. A partir desta etapa está preparada a infra-estrutura para a execução da principal contribuição deste projeto de mestrado. Para possibilitar a criação da Slim-Tree no PostgreSQL foi implementada uma função

utilizando o PL/pgSQL, criada no módulo de MAM, o qual realiza a interface entre as funções escritas em SQL e em C++. A Figura 13 mostra o módulo MAM, que contém diversas funções. Os detalhes de implementação são apresentados na Seção 4.2.

A função de criação da Slim-Tree, definida pelo RAFIKI, corresponde à estrutura sintática da chamada da função correspondente em SQL mostrada como Código 4. A função *slim_distance_function_create* é responsável pela criação da estrutura do índice, a qual recebe como parâmetros a tabela, a coluna do vetor de características, a função de distância e o extrator de características. Esses parâmetros garantem a unicidade da estrutura, para seu armazenamento no PostgreSQL. Ao executar essa função, um arquivo será criado no gerenciador de páginas, concatenando os valores dos parâmetros com o objetivo de deixar intuitivo o significado da Slim-Tree criada. O Código 5 mostra um exemplo de criação de uma Slim-Tree, utilizando a função de distância Euclidiana e o extrator de características Color Layout.

Código-fonte 4 – Criação da Slim-Tree

```
1: SELECT slim_distance_function_create(
2:     'TABELA', 'COLUNA_VETOR_SIGNATURE', 'FUNCAO_DE_DISTANCIA',
     'EXTRATOR');
```

Código-fonte 5 – Exemplo de Criação Slim-Tree

```
1: SELECT slim_distance_function_create(
2:     'tab_exemplo', 'blob_caracteristicas', 'euclidean_distance'
     , 'color_layout_extractor');
```

Com a estrutura da Slim-Tree criada, o próximo passo consiste em populá-la com os elementos previamente armazenados na tabela do banco, conforme apresentado na próxima seção.

4.1.4 Inserção dos dados na Slim-Tree

A estrutura sintática que deve ser utilizada para adicionar os vetores de características das imagens já inseridas na Slim-Tree é mostrado como Código 6. A função *slim_distance_function_add* se encarrega de adicionar os elementos contidos na tabela. Ela recebe como parâmetros a coluna de *blob_caracteristicas* (vetor de características), a coluna do identificador dos elementos, e, todos os parâmetros textuais, que são usados para recuperar o arquivo salvo da Slim-Tree, criado no gerenciador de páginas. O Código 7 mostra um exemplo para a população de uma Slim-Tree criada com os dados existentes na tabela. Neste exemplo são utilizados a função de distância Euclideana e o extrator Color Layout.

Código-fonte 6 – Adição de Elementos Complexos

```
1: SELECT slim_distance_function_add(
2:     COLUNA_VETOR_SIGNATURE, COLUNA_ID, 'TABELA',
3:     'COLUNA_VETOR_SIGNATURE','FUNCAO_DE_DISTANCIA', 'EXTRATOR'
4: )  from TABELA;
```

Código-fonte 7 – Exemplo de Adição de Elementos Complexos

```
1: SELECT slim_distance_function_add(
2:     blob_caracteristicas, id, 'tab_exemplo',
3:     'blob_caracteristicas', 'euclidean_distance',
4:     'color_layout_extractor'
5: ) from tab_exemplo;
```

Com os elementos inseridos na Slim-Tree, a próxima etapa permite realizar as buscas por abrangência (range) e por vizinhos mais próximos (knn), conforme apresentado na próxima seção.

4.1.5 Consultas por similaridade na Slim-Tree

Nesta etapa, o Código 8 mostra a estrutura sintática de como deve ser feita uma consulta por abrangência para recuperar os elementos mais próximos. A função *slim_distance_function_-range* se encarrega de buscar os elementos a partir de um raio de consulta. Ela recebe como parâmetro a coluna de *blob_caracteristicas*, o raio deve ser especificado como um valor de até dois dígitos, e por fim, todos os parâmetros textuais são usados para recuperar o arquivo salvo na Slim-Tree, criado no gerenciador de páginas.

O Código 9 mostra um exemplo de como solicitar uma busca por abrangência em uma Slim-Tree, retornando um conjunto de elementos complexos. Neste exemplo são utilizados a função de distância Euclideana, o extrator Color Layout e o raio de consulta 0,07. Note que, neste exemplo, um elemento já armazenado na tabela é utilizado como o elemento de consulta, no caso o elemento complexo com o id = 242. Note-se que o vetor de características usado como centro de consulta não precisa estar inserido na tabela. Caso o centro de consulta não esteja armazenado, deve ser indicado um arquivo armazenado no Sistema Operacional, e extraído o vetor de características, cujo exemplo é mostrado no Código 9.

Código-fonte 8 – Consulta por Abrangência

```
1: SELECT slim_distance_function_range(
2:     COLUNA_VETOR_SIGNATURE, RAIO, 'TABELA', 'COLUNA',
3:     'FUNÇÃO_DE_DISTANCIA', 'EXTRATOR'
4: );
```

Código-fonte 9 – Exemplo de Consulta por Abrangência

```
1: SELECT B.id, B.path from images5k_color B JOIN (
2:     SELECT slim_distance_function_range(
3:         blob_caracteristicas, 0.07, 'images5k_color',
4:         'blob_caracteristicas', 'EuclideanDistance',
5:         'color_layout_extractor')
6:     as range from images5k_color where id = 242)
7: A on A.range = B.id;
```

De maneira análoga à consulta por abrangência, para realizar a consulta por vizinhos mais próximos é necessário chamar a função *slim_distance_function_knn*, a qual busca os k elementos mais próximos. A estrutura de uma expressão que realiza essa consulta é mostrada no (Código 10). Essa função recebe os mesmos parâmetros da função de busca por abrangência, com a exceção do parâmetro raio, que é substituído pela quantidade de vizinhos, a serem recuperados (k).

Código-fonte 10 – Consulta por Vizinhos mais Próximos

```
1: SELECT slim_distance_function_knn(
2:     COLUNA_VETOR_SIGNATURE, VIZINHOS, 'TABELA', 'COLUNA',
3:     'FUNÇÃO_DE_DISTANCIA', 'EXTRATOR'
4: );
```

Os resultados das consultas por abrangência e por vizinhos mais próximos são um conjunto de tuplas, que correspondem aos parâmetros de consulta e aos elementos existentes na estrutura.

4.2 Detalhes de Implementação

Os comandos SQL descritos nas seções anteriores são utilizadas pelos usuários para criar, popular e consultar a Slim-Tree no SGBDR PostgreSQL. No entanto, para possibilitar a execução destes recursos, foi necessário implementar funções seguindo o padrão das interfaces de programação (API) em que o SGBDR possa interpretar. Isso corresponde a códigos de funções em C++ e PL/pgSQL, para criar, adicionar e buscar, conforme mostrado na Figura 13.

Um exemplo é o Código 11, que consiste em uma função em PL/pgSQL. Essa função cria uma Slim-Tree usando a distância Euclideana, e a mesma recebe por parâmetro quatro variáveis textuais. Esses parâmetros são utilizados para a manipulação do gerenciador de páginas.

Código-fonte 11 – Código fonte PL/pgSQL para criar uma Slim-Tree

```
1: CREATE OR REPLACE FUNCTION slim_euclidean_distance_create(
2:     text, text, text, text)
3:   RETURNS boolean AS
4: :path, 'slim_euclidean_distance_create'
5:   LANGUAGE c IMMUTABLE STRICT
6:   COST 1;
```

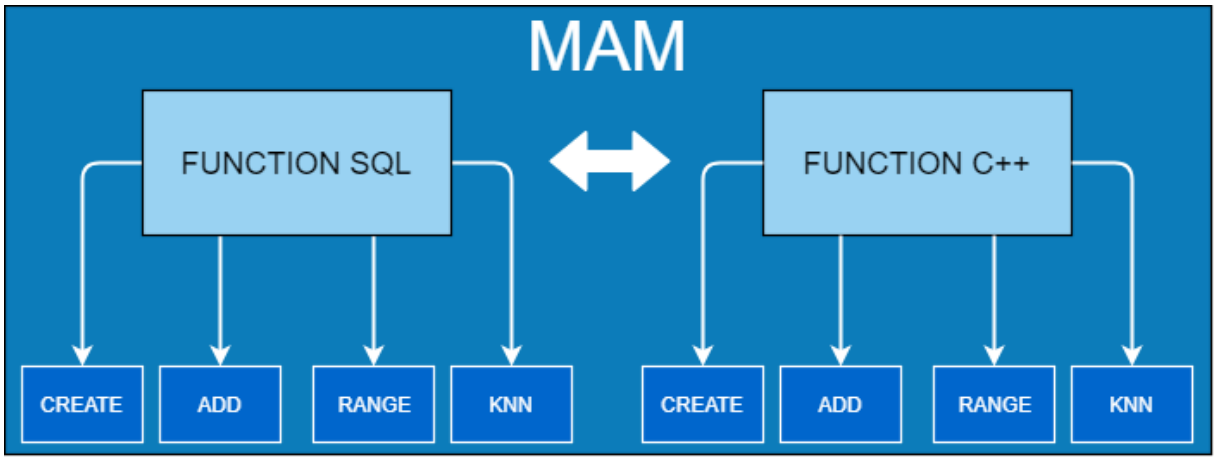
O Código 12 apresenta um trecho em C++ da função referente à criação da Slim-Tree, correspondente ao Código 11, que está em PL/pgSQL.

Código-fonte 12 – Código fonte C++ para criar uma Slim-Tree

```
1: #define SIMILARITY_SUPPORT_SLIM_CREATE(sql_function_name,
     hermes_function_name); \
2: PG_FUNCTION_INFO_V1(sql_function_name); \
3: Datum sql_function_name(PG_FUNCTION_ARGS) { \
4: ...} \
```

Todos os códigos desenvolvidos durante este trabalho de mestrado, referentes ao RAFIKI, estão disponíveis para a comunidade no repositório do grupo de pesquisa GBdI: https://bitbucket.org/gbdi/. O desenvolvimento foi realizado utilizando a versão estável 9.4 do PostgreSQL e a versão 7 do Sistema Operacional CentOS.

Figura 13 – Módulo do Método de Acesso Métrico no RAFIKI.



Fonte: Elaborada pelo autor.

Note-se que o MAM Slim-Tree implementado na Arboretum não permite operações de remoção de dados. Por isso, não foram criados comandos para as operações de atualização e remoção *UPDATE* e *DELETE* de dados complexos.

4.3 Considerações Finais

Neste capítulo foram apresentados como acrescentar o método de indexação da MAM Slim-Tree no SGBDR PostgreSQL. Para isso, foram também definidas as estruturas básicas utilizadas pelo MAM, bem como foi criada a sintaxe básica em SQL para a criação, inserção e busca na estrutura. Desta forma, foi implementado o protótipo do sistema RAFIKI, que utiliza a estrutura do PostgreSQL e herda as funcionalidades de extração de características e comparações por funções de distância do KIARA. No próximo capítulo o protótipo RAFIKI é implementado e validado, por meio de uma análise experimental usando dados do contexto médico.

CAPÍTULO

5

ANÁLISE EXPERIMENTAL

Neste capítulo são apresentados e discutidos os experimentos e resultados obtidos para a avaliação e validação do sistema RAFIKI. Para fins de validação, foram utilizados dados complexos do tipo imagem. Os resultados reportados aqui foram preliminarmente publicados no trabalho (NESSO Jr. *et al.*, 2018). A Seção 5.1 descreve o processo de armazenamento de novas imagens, possibilitando o uso do índices Slim-Tree no PostgreSQL. A Seção 5.2 mostra como o RAFIKI pode ser integrado com outras ferramentas de análise. A Seção 5.3 descreve os dados utilizados para realizar consultas por similaridade. A Seção 5.4 apresenta as motivações e os objetivos dos cenários em que foi utilizado o RAFIKI. A Seção 5.4.1 apresenta o primeiro cenário, onde o usuário deseja analisar os dados obtidos pela consulta. A Seção 5.4.2 apresenta um cenário em que o médico deseja filtrar a consulta, levando em consideração apenas duas classes diferentes. A Seção 5.4.3 apresenta o último cenário em que o foco é uma consulta por similaridade, que retorna dados associados ao elemento complexo de consulta. Por fim, a Seção 5.5 traz uma avaliação de desempenho, comparando os resultados do RAFIKI com seu antecessor, o sistema KIARA.

5.1 Armazenando Imagens com o RAFIKI

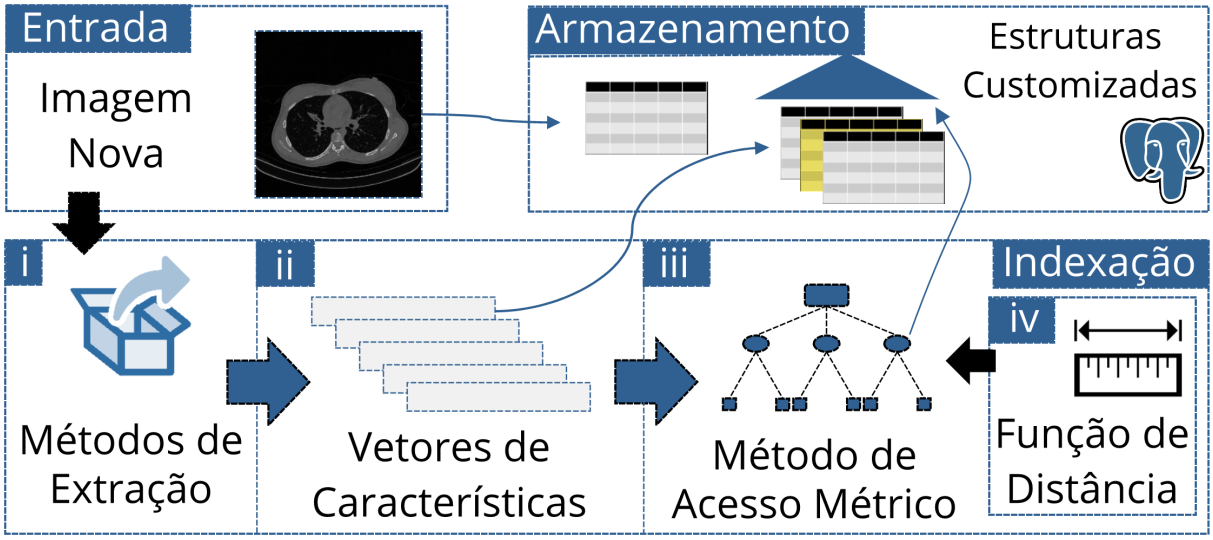
A validação do RAFIKI foi realizada usando dados complexos do tipo imagem do domínio de exames médicos. Para cada imagem de entrada, o fluxo mostrado na Figura 14 foi executado. Dada uma imagem, o RAFIKI (i) extrai suas características utilizando os extratores implementados. A ferramenta tem atualmente disponíveis oito extratores: *Color Layout, Color Structure, Color Temperature, Edge Histogram, Texture Browsing, LBP, Scalable Color* e *Haralick*. Cada um analisa diferentes características (tais como cor, textura e formato) e (ii) as representa em um vetor de características.

A partir do momento que a extração é concluída com sucesso, as características extraídas são armazenadas em uma tabela do banco. Com isso, o SGBDR pode auxiliar na armazenagem,

recuperação, processamento e análise de dados complexos, que podem ser comparados por similaridade. Além disso, o RAFIKI permite que, (iii) os dados sejam indexados pelo Método de Acesso Métrico Slim-Tree. Isso é possível, porque a ferramenta permite que o MAM utilize uma (iv) função de distância para comparar as características considerando os dados no espaço métrico, utilizando para isso os comandos SQL de criação, população e consulta que foram introduzidos no Capítulo 4.

Esses passos são seguidos de forma a armazenar o conteúdo no banco. Uma vez que as informações já tenham sido armazenadas, é possível realizar as consultas, conforme descrito a seguir.

Figura 14 – O RAFIKI armazena diversas informações em uma base de dados PostgreSQL. Para cada imagem de entrada, RAFIKI extrai suas características e as armazena nas tabelas do banco. Os vetores de características são indexados em um MAM usando as funções de distância específicas.



Fonte: NESSO Jr. *et al.* (2018).

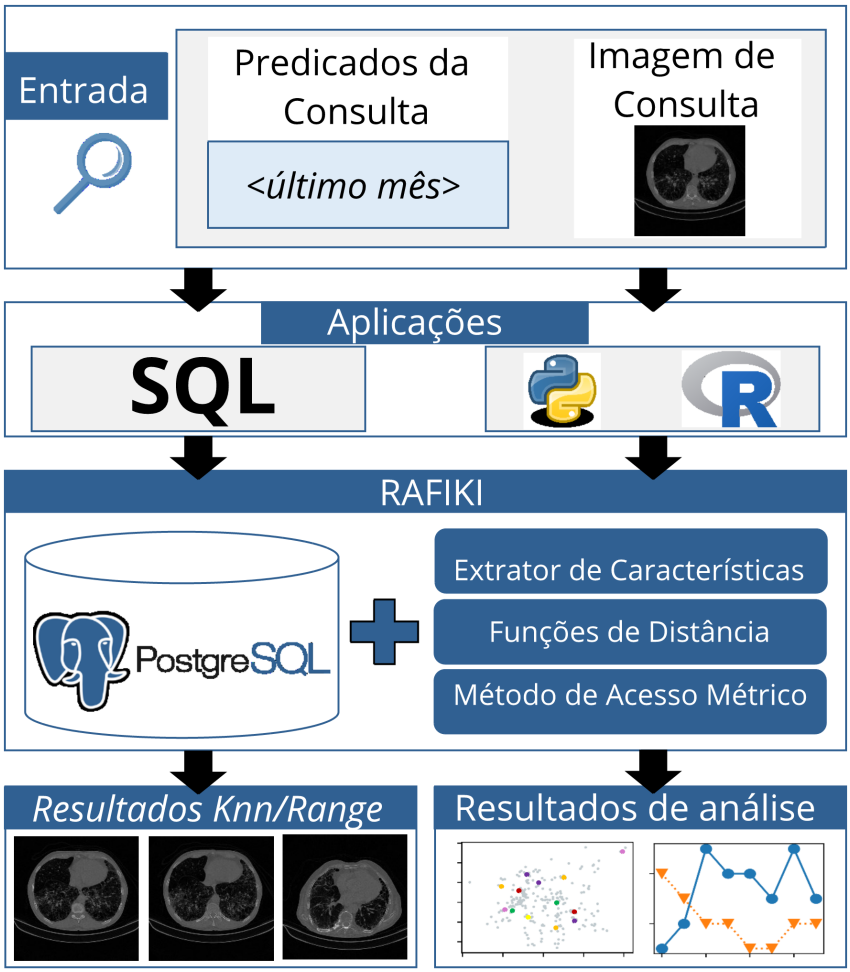
5.2 Integração de aplicativos com o RAFIKI

A Figura 15 mostra a interação entre o RAFIKI e outras aplicações. Uma vez que a informação esteja armazenada, como apresentado na Figura 14, os usuários podem acessar a informação de diferentes maneiras. O usuário pode realizar consultas por similaridades (abrangência ou vizinhos mais próximos) para recuperar as imagens mais similares de acordo com o predicado da consulta e a imagem de entrada. Usando esse recurso, usuários de ferramentas como *R-Statistics*[1] ou *Python*[2] podem realizar outras análises, como calcular os principais componentes (PCA) de um conjunto de imagens, o que permite aos usuários visualizar as imagens

[1] <https://www.r-project.org/>
[2] <https://www.python.org/>

buscadas e projetá-las em um espaço bi ou tri-dimensional, por exemplo. RAFIKI processa a consulta realizada pelo usuário da aplicação combinando a estrutura do PostgreSQL aos recursos de extração de características, funções de distância e ao MAM. Isso proporciona também a integração de outras aplicações e ferramentas, que podem acessar o PostgreSQL, enquanto o RAFIKI fornece toda a estrutura necessária para atender aos sistemas que realizam CBIR, com as vantagens do PostgreSQL. Como resultado, torna-se possível usar as funcionalidades do RAFIKI em um alto nível de abstração, sem que o usuário tenha contado direto com o SQL das consultas realizadas.

Figura 15 – RAFIKI fornece uma infraestrutura que permite a integração entre diferentes linguagens e plataformas. Grande parte do processamento usa os recursos disponibilizados pelo SGBDR. A armazenagem do dado pode ser feita por meio de um predicado ou uma imagem. Por exemplo: usuários da linguagem Python ou R podem usar o RAFIKI para realizar consultas por similaridade, nas quais a ferramenta processa a requisição usando o PostgreSQL junto aos recursos de extração de características, funções de distância e MAM.



Fonte: NESSO Jr. *et al.* (2018).

5.3 Descrição dos Dados e Organização

Para as análises realizadas sobre o RAFIKI, foi utilizado um conjunto de dados proprietário, contendo tomografias computadorizadas (CTs) da região do pulmão. A coleção de imagens foi disponibilizada, para estudos, pelo Hospital Clínico de Ribeirão Preto (HCRP). Todas as 246 imagens foram coletadas de 108 exames diferentes e foram classificadas pelos especialistas em 6 classes: normal, consolidação, espessamento, enfisema, vidro fosco e favo de mel. Cada uma dessas classes são atribuídas a doenças clínicas caracterizadas visualmente no pulmão. As imagens estão igualmente distribuídas, com aproximadamente com 40 imagens por classe. Associada a cada imagem há uma máscara e uma representação em um mapa de calor de anormalidades. As máscaras são imagens pré-processadas representando somente as regiões do tecido pulmonar, excluindo por exemplo a cavidade do corpo e outras regiões. O mapa de calor consiste em imagens com regiões pulmonares, cada uma delas sendo representadas em diferentes intensidades de cor. A intensidade é calculada a partir da probabilidade de ocorrer uma anormalidade pulmonar na região. Essas imagens adicionais foram obtidas utilizando o método *dp-BREATH* (CAZZOLATO *et al.*, 2017; CAZZOLATO *et al.*, 2019). A Figura 17 apresenta exemplos das imagens utilizadas, contendo (i) a imagem original, (ii) a imagem segmentada (máscara), e (iii) o mapa de calor (heatmap) gerado. Essas diferentes informações foram utilizadas para exemplificar a capacidade do RAFIKI em armazenar, organizar e indexar imagens e informações atreladas a elas, de maneira eficiente.

Para as análises experimentais, as imagens de CT estão organizadas em 7 tabelas diferentes, uma para cada classe e mais uma tabela que armazena os vetores de características de todas as 246 imagens. A mesma organização foi aplicada sobre as imagens referentes às máscaras do pulmão. Existem no total 15 tabelas, das quais 7 são vetores de características das imagens de tomografia computadorizadas, outras 7 tabelas com os vetores de características extraídas das máscaras, e por fim, mais uma tabela contendo as imagens do mapa de calor. As imagens representando o mapa de calor são utilizadas apenas como conteúdo adicional às consultas SQL. Juntamente aos mapas de calor, são armazenados as imagens e os ids correspondentes das imagens originais dos CTs. Todas as 15 tabelas são armazenadas no RAFIKI utilizando o PostgreSQL.

5.4 Usando o RAFIKI em Cenários de Aplicação

Para avaliar o sistema proposto em uma aplicação prática, foram definidos três cenários de aplicação, buscando explorar as diferentes funcionalidades e vantagens do RAFIKI e do uso do SGBDR. Esses cenários foram os seguintes:

- **Cenário 1.** Um médico deseja verificar um diagnóstico para uma imagem de CT de pulmão. Neste caso, o profissional realiza uma consulta por similaridade, fornecendo a imagem

de CT como entrada, que servirá como centro de consulta. Como resultado, espera-se recuperar também os metadados associados a imagens semelhantes, possivelmente pertencentes a outros exames.

- **Cenário 2.** O médico analisa a tomografia de um paciente e quer produzir o diagnóstico desse novo exame escolhendo entre duas possíveis classes: enfisema e consolidação. Assim, o profissional checa os dados de casos anteriores, que possuem diferentes vetores de características, extraídos de ambas as classes.

- **Cenário 3.** Um médico deseja analisar os padrões de textura referentes apenas aos tecidos pulmonares. Para isso, o profissional utiliza uma máscara, que apresenta apenas o tecido pulmonar como objeto de consulta. Além disso, o médico está interessado em recuperar os dados referentes a casos históricos. Dessa forma, ele busca as imagens mais semelhantes e suas informações relacionadas, como o mapa de calor para anormalidades pulmonares.

Os cenários mencionados acima são detalhados nas próximas seções, explorando as funcionalidades do RAFIKI e os recursos do SGBDR PostgreSQL.

5.4.1 Cenário 1 - Busca Genérica para Descoberta de Doença

No primeiro cenário de aplicação, o especialista encontra-se examinando uma nova imagem de CT de pulmão, buscando no banco de dados as imagens mais similares, de casos já analisados. O profissional está interessado no diagnóstico associado às imagens mais similares ao elemento de entrada, para auxiliar na escolha de um tratamento adequado para o caso atual. Essa consulta é interessante porque permite ao especialista buscar em casos passados, de forma que o profissional consulte casos já analisados por ele ou por outros profissionais. Isso é uma vantagem, pois o especialista pode utilizar do seu conhecimento, como de outros profissionais.

Para executar a consulta, a imagem de CT é submetida ao RAFIKI, que procura pelas imagens mais similares em cada uma das 6 classes. Cada classe tem uma tabela contendo as imagens correspondentes, e cada tabela está indexada com uma Slim-Tree específica. O Código 13 mostra um exemplo de consulta, dentre as efetuadas para este cenário de aplicação. Nele, compara-se uma imagem da tabela de CT de pulmão (id = 176) com as imagens que contém enfisema, de dados históricos. Nota-se que, para essa consulta, é utilizada uma Slim-Tree criada sobre os dados da tabela Enfisema, usando o extrator de características Color Layout e a função de distância Euclidiana. O raio de consulta usado, para este exemplo, foi de 0,07, e como resultado é retornado o id e o caminho das imagens que atendem ao raio de consulta estabelecido.

Na linha 2 é feita uma consulta usando a Slim-Tree. A linha 3 contém os parâmetros usados na consulta da linha 2, que são: o vetor de características da imagem de consulta (id = 176), o raio de consulta (0,07), a tabela em que será feita a busca (Enfisema), a coluna

de vetores de características da tabela Enfisema (blob_caracteristicas), a função de distância (EuclideanDistance), e o extrator de características (color_layout_extractor). O resultado da consulta na Slim-Tree é armazenado na tabela temporária Imagens, e contém os ids de todas as imagens que atendem à consulta. Depois, na linha 4, é feita uma junção da tabela resultante com a tabela Enfisema, retornando assim os ids e caminhos de todas as imagens que atendem à consulta.

Código-fonte 13 – Exemplo de consulta do cenário 1

```
1: SELECT E.id, E.path from Enfisema E JOIN (
2:     SELECT slim_euclidean_distance_range(
3:         (SELECT blob_caracteristicas FROM ImagensOriginais
     WHERE id = 176), 0.07, 'Enfisema', 'blob_caracteristicas', '
     EuclideanDistance', 'color_layout_extractor') as Imagens)
4:     Resultado on Resultado.Imagens = E.id;
```

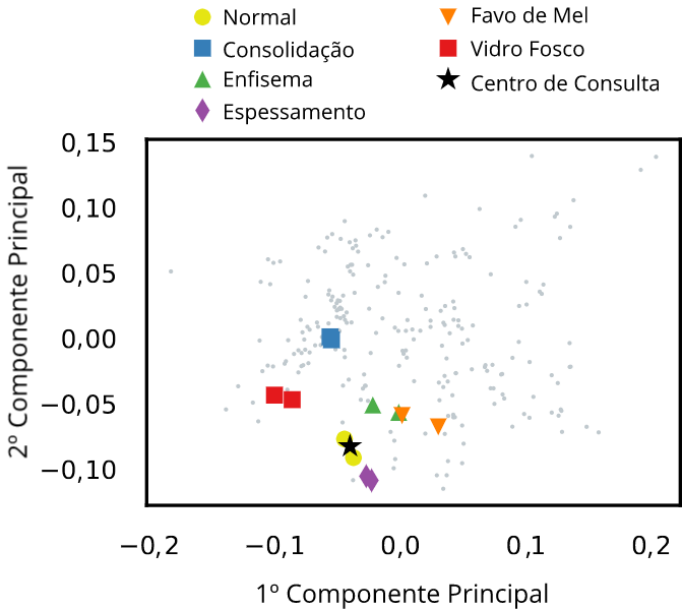
O especialista também pode estar interessado em classificar a nova imagem em relação a uma doença pulmonar. Assim, ele(a) recupera, por exemplo, as duas imagens mais similares de cada classe utilizando o RAFIKI, totalizando 12 tomografias computadorizadas com o diagnóstico associado. As informações adicionais dos casos anteriores podem ser combinadas ao conhecimento do especialista durante o processo de diagnóstico.

A Figura 16 mostra a projeção de uma PCA bidimensional de todas as 246 imagens de CT de pulmão no espaço métrico, em que cada ponto cinza corresponde ao vetor de características de uma imagem. A projeção foi realizada integrando o RAFIKI ao Python, utilizando-o como uma ferramenta analítica. Na figura, o centro de consulta é realçado como uma estrela preta, e as duas imagens de cada classe recuperadas são destacadas com cores de acordo com a legenda. Foi observado que a imagem de CT, submetida para servir de centro de consulta, pode ser classificada como saudável devido a sua proximidade com as imagens recuperadas e distância elevada das imagens mais semelhantes a partir de outras de cada classe com imagens anormais. A integração do RAFIKI com diversas ferramentas permite o uso de diferentes tecnologias, técnicas e algoritmos disponíveis em pacotes de análise de dados da literatura. Especificamente para o campo da Ciência de Computação e áreas afim, duas ferramentas amplamente utilizadas que foram utilizadas nestes experimentos são o Python e a ferramenta R.

5.4.2 Cenário 2 - Decidindo Entre Duas Classes Específicas

O segundo cenário de aplicação ocorre quando um especialista tem uma nova imagem para analisar e precisa decidir entre duas classes diferentes, como consolidação e enfisema. O que é exemplificado nesse cenário é o uso de diferentes extratores, para obter diferentes vetores de características para o mesmo conjunto de dados, mas que representam diferentes

Figura 16 – Cenário 1: Plotagem do PCA no espaço métrico, com todas as imagens de CT de pulmão (pontos em cinza), a imagem correspondente ao centro da consulta (estrela preta), e as duas imagens mais próximas, recuperadas de cada classe.



Fonte: NESSO Jr. *et al.* (2018).

características visuais das imagens. Ao recuperar as imagens mais semelhantes de cada classe, enquanto diversifica-se o tipo do vetor de características utilizado, o profissional pode comparar os resultados das consultas, para cada doença. Cada vetor de características destaca diferentes atributos visuais nas imagens, daí a importância de um cenário onde se utilizam diferentes extratores. Neste cenário, o especialista está interessado em analisar as propriedades de cor e forma das imagens de CT de pulmão. Assim, duas consultas são realizadas para cada doença, uma usando o *Color Layout* (para cor) e uma usando o *Edge Histogram* (para textura).

O Código 14 mostra um exemplo da consulta, dentre as efetuadas para o cenário da aplicação 2. Nele, compara-se uma imagem da tabela de CT de pulmões originais (id = 165) com as imagens que contém Consolidação, de dados históricos. Nota-se que, para essa consulta, é utilizada uma Slim-Tree criada sobre os dados da tabela Consolidação, usando o extrator de características Edge Histogram e a função de distância Euclidiana. Para este exemplo, é utilizada uma consulta aos vizinhos mais próximos, com $k = 3$, e como resultado é retornado o id e o caminho das imagens resultantes.

Na linha 2 é feita uma consulta usando a Slim-Tree correspondente aos dados de consolidação. A linha 3 contém os parâmetros usados na consulta da linha 2, que são: o vetor de características da imagem de consulta (id = 165), o valor de k (3), a tabela em que será feita a busca (Consolidacao), a coluna de vetores de características da tabela Consolidacao (blob_caracteristicas), a função de distância (EuclideanDistance), e o extrator de características (edge_histogram_extractor). O resultado da consulta na Slim-Tree é armazenado na tabela tem-

porária Imagens, e contém os ids de todas as imagens que atendem à consulta. Depois, na linha 4, é feita uma junção da tabela resultante com a tabela "Consolidacao", retornando assim os ids e caminhos de todas as imagens que atendem à consulta.

Código-fonte 14 – Exemplo de consulta do cenário 2

```
1: SELECT C.id, C.path from Consolidacao C JOIN (
2:     SELECT slim_euclidean_distance_knn(
3:         (SELECT blob_caracteristicas FROM ImagensOriginais
     WHERE id = 165), 3, 'Consolidacao', 'blob_caracteristicas',
     'EuclideanDistance', 'edge_histogram_extractor') as Imagens)
4:     Resultado on Resultado.Imagens = C.id;
```

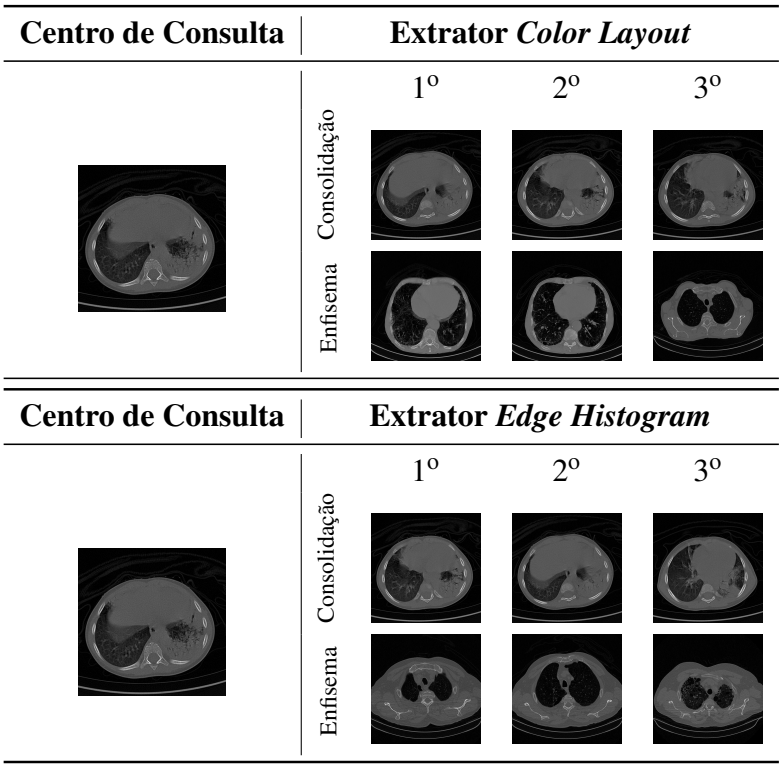
A partir da imagem de consulta, o especialista recupera as três imagens mais semelhantes de cada classe. A quantidade de imagens recuperadas pelo RAFIKI é variável de acordo com definição do usuário, que pode escolher se basear em uma quantidade maior ou menor de casos. A Figura 17 mostra os resultados dessas consultas, alternando o extrator empregado. Foram observadas algumas repetições nas imagens recuperadas. A 1ª e a 2º imagens de consolidação aparecem novamente, quando se varia o extrator empregado (somente em ordem diferente). Por outro lado, as imagens recuperadas da classe enfisema não apresentam repetição. Esse tipo de "descoberta" exemplificado aqui, que pode ser de grande interesse do usuário e é dependente da semântica das aplicações que usam os SGBDs, passa a poder ser feita com grande flexibilidade e facilidade graças aos recursos desenvolvidos neste trabalho.

Com base nos resultados, o RAFIKI pode ajudar o especialista na tomada de decisão, fornecendo casos semelhantes de doenças específicas. Além disso, se os resultados obtidos não forem suficientes, e a suspeita sobre qual doença a imagem apresenta persiste, o profissional pode realizar diferentes consultas. Por exemplo, o especialista pode recuperar as imagens mais similares de outras doenças ou alternar os padrões visuais, aplicando outros extratores de características. Neste cenário, o especialista é capaz de concluir que a imagem de consulta apresenta consolidação de acordo com o extrator *Color Layout*. Para uma análise mais abrangente, neste exemplo o extrator *Edge Histogram* pode ser usado apra reforçar a primeira conclusão, fornecendo evidências além da utilização do *Color Layout*.

5.4.3 Cenário 3 - Recuperando Informações Adicionais

Observado o poder de organização de um SGBDR, o terceiro cenário de aplicação explora a recuperação de informações adicionais associadas aos resultados da consulta. Isso é possível porque o RAFIKI pode integrar várias tabelas. Neste cenário, o especialista realiza uma consulta por similaridade usando uma máscara, que contém apenas as regiões do tecido pulmonar de uma imagem de CT. Aqui, o profissional não está interessado apenas nas imagens mais semelhantes,

Figura 17 – Cenário 2: Com a mesma imagem de consulta, o especialista analisa os padrões de cor e forma. Assim o RAFIKI retorna as imagens mais similares das classes consolidação e enfisema, que apresentam diferentes características visuais.



Fonte: NESSO Jr. *et al.* (2018).

mas também em outras imagens relacionadas às imagens recuperadas. Neste caso, o profissional está interessado nas imagens relacionadas às máscaras do pulmão, nas imagens de CT completas, e em seus respectivos mapas de calor.

O Código 15 mostra um exemplo de consulta, dentre as efetuadas para o cenário de aplicação 3. Nele, compara-se uma imagem da tabela de CT de máscaras de pulmões (id = 21), que é uma imagem previamente processada e segmentada, com as demais imagens de máscara, de dados históricos. Nota-se que, para essa consulta, é utilizada uma Slim-Tree criada sobre os dados da tabela Máscaras, usando o extrator de características Texture Browsing e a função de distância Euclidiana. Para este exemplo, também é utilizada uma consulta aos vizinhos mais próximos, tal que $k = 3$, e como resultado é retornado o id e o caminho das imagens de máscara.

Na linha 2 é feita uma consulta usando a Slim-Tree correspondente às imagens de máscara dos CTs de pulmão. A linha 3 contém os parâmetros usados na consulta da linha 2, que são: o vetor de características da imagem de consulta (id = 21), o valor de k (3), a tabela em que será feita a busca (Mascaras), a coluna de vetores de características da tabela Mascaras (blob_caracteristicas), a função de distância (EuclideanDistance), e o extrator de características (texture_browsing_extractor). O resultado da consulta na Slim-Tree é armazenado na tabela temporária Imagens, e contém os ids de todas as imagens que atendem à consulta. Depois, na

linha 4, é feita uma junção da tabela resultante com a tabela “Mascaras”, retornando assim os ids e caminhos de todas as imagens que atendem à consulta.

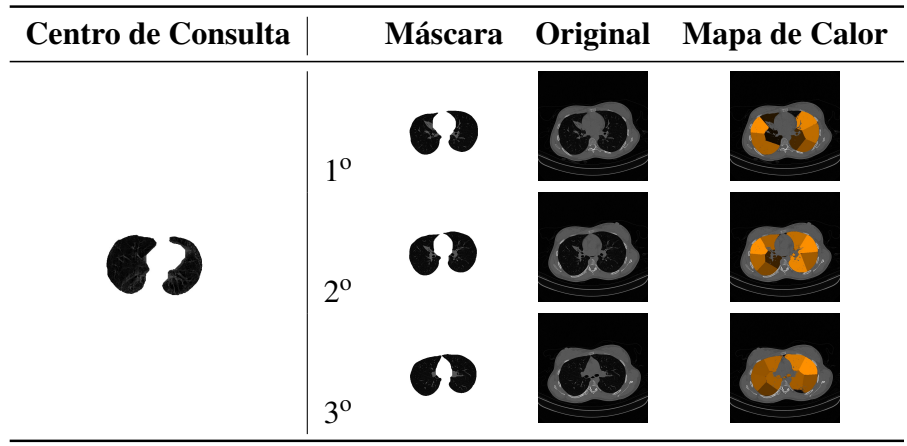
Código-fonte 15 – Exemplo de consulta do cenário 3

```
1: SELECT M.id, M.path from Mascaras M JOIN (
2:     SELECT slim_euclidean_distance_knn(
3:         (SELECT blob_caracteristicas FROM Mascaras WHERE id =
      21), 3, 'Mascaras', 'blob_caracteristicas', '
      EuclideanDistance', 'texture_browsing_extractor') as Imagens
      )
4:     Resultado on Resultado.Imagens = M.id;
```

A Figura 18 mostra um exemplo de tal consulta, recuperando as três imagens mais similares, associando-as com suas respectivas imagens de CT e do mapa de calor.

Neste cenário foi abordado que o RAFIKI estende o SGBDR e herda suas características, permitindo a integração de informações de várias tabelas. Isso acarreta a vários benefícios, como relacionar várias imagens pré-processadas da mesma imagem de CT original para análise posterior. Além disso, o RAFIKI permite separar o armazenamento, classificar e organizar os diferentes tipos de imagens, por exemplo, tomografias cerebrais, tomografias computadorizadas da coluna e mamografias. Neste último cenário, fica mais claro para o analista quais são as regiões mais críticas da imagem, um conhecimento que leva em consideração os diagnósticos anteriores.

Figura 18 – Cenário 3: RAFIKI integra várias tabelas, fornecendo informações adicionais sobre os resultados recuperados. Dada uma imagem de consulta, o sistema retorna as máscaras mais semelhantes, a imagem original associada, e sua representação por mapa de calor.



Fonte: NESSO Jr. *et al.* (2018).

5.4.4 Considerações Finais da Aplicação do RAFIKI

Todos os cenários descritos até aqui exemplificam a utilidade do RAFIKI para aplicações práticas, considerando sua organização (com suporte do SGBDR), sua integração com ferramentas de análise, e a possibilidade de combinar diferentes informações, em diferentes tabelas, e recuperá-las quando relevante. Com a contribuição do presente trabalho de mestrado, as consultas realizadas no RAFIKI utilizaram a sintaxe de consulta SQL, com os comandos para utilização do método de acesso métrico Slim-Tree (definidos no Capítulo 4). A seguir é mostrado o ganho do uso do método de acesso métrico Slim-Tree para realizar as consultas dentro do SGBDR, usando o RAFIKI.

5.5 Análise de Desempenho

Nesta seção, avaliamos o desempenho do RAFIKI em relação à recuperação de imagens. Todos os experimentos foram realizados em uma máquina virtual, com o sistema operacional CentOS 7, 2 GB de memória RAM e com os parâmetros padrão do PostgreSQL. Para a análise de desempenho foi usado um conjunto de dados maior, chamado *MAMMOSET* (OLIVEIRA *et al.*, 2017) para avaliar melhor o ganho de desempenho das capacidades de indexação do RAFIKI. O conjunto contém um total de 3.457 regiões de interesse (ROIs) de imagens de mama, em que cada ROI é padronizada como uma imagem de 256x256 pixels. Os resultados do RAFIKI foram comparados com o KIARA, que foi descrito no Capítulo 3, e que não emprega um MAM para indexar as imagens armazenadas.

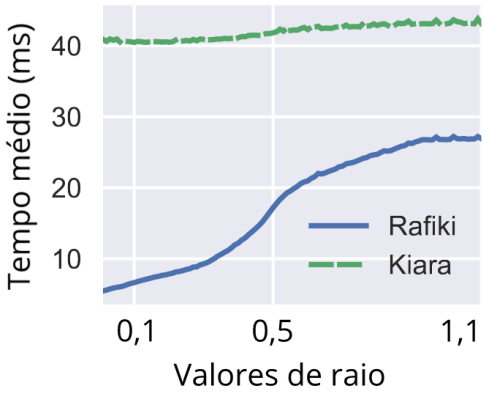
Na Figura 19 são apresentados os tempos de execução de uma consulta, em ambos os métodos. A mesma consulta foi executada em ambas as abordagens: recuperar todas as imagens semelhantes a uma determinada imagem de consulta até um valor de intervalo específico. Os valores empregados para o range variaram entre 0,01 a 1,1 em incrementos de 0,01. Cada consulta foi executada 20 vezes, os 10% dos melhores e piores tempos foram removidos e os resultados médios são apresentados no gráfico.

O Código 16 mostra um exemplo de consulta realizada no KIARA, para cálculo do tempo de execução. O Código 17 mostra o exemplo correspondente, para o RAFIKI. Nessas consultas, foi utilizado o raio de abrangência de 0,01, a função de distância Euclidiana, o extrator de características Color Layout, a imagem de consulta com id = 2, sobre a tabela Mammoset.

Código-fonte 16 – Consulta realizada no KIARA para cálculo de tempo de execução

```
1: SELECT id, path
2: FROM Mammoset
3: WHERE dist('Mammoset', 'image', 'CL_EU', image_id, (SELECT
     fem_extraction('Mammoset', 'image', 'CL_EU', (SELECT image
     FROM Mammoset WHERE id = 2)))) <= 0.01;
```

Figura 19 – Comparação de tempo do método KIARA *versus* RAFIKI variando o valor do intervalo.



Fonte: NESSO Jr. *et al.* (2018).

Código-fonte 17 – Consulta realizada no RAFIKI para cálculo de tempo de execução

```
1: SELECT M.id, M.path FROM Mammoset M  JOIN
2:     (SELECT slim_euclidean_distance_range((SELECT
      blob_caracteristicas FROM Mammoset WHERE id = 2), 0.01, '
      Mammoset', 'blob_caracteristicas', 'EuclideanDistance', '
      color_layout_extractor') as Imagens)
3:     Resultado on Resultado.Imagens = M.id;
```

Como esperado, o RAFIKI apresentou um desempenho mais rápido que o KIARA, devido à capacidade do MAM de realizar podas no espaço de busca. Com o intervalo de 0,1 (recuperando apenas uma imagem), o KIARA obteve um desempenho de 40ms enquanto o RAFIKI executou a mesma consulta em 6ms (mais de 6 vezes mais rápido). Mesmo para um intervalo grande, o RAFIKI é significantemente mais rápido: com o intervalo de 0,5 (recuperando 1.535 imagens), com o RAFIKI a consulta foi executada em 17ms enquanto que com o KIARA foi executada em 42ms (60% mais rápido).

Vale a pena notar que quando o valor do intervalo era de 1,1, englobando todos os elementos do banco de dados, o tempo de execução do RAFIKI era 38% mais rápido que o KIARA, com 27ms e 43ms, respectivamente. Isso ocorre porque o KIARA calcula a distância do objeto de consulta para cada elemento em uma varredura sequencial. Diferentemente, uma consulta com o índice no RAFIKI calcula apenas a distância do elemento da consulta ao representante do nó. O raio de cobertura do nó limita a maior distância entre o representante e seus elementos. Assim, ao executar uma consulta com um intervalo maior ou igual ao raio de abrangência, todos os elementos no nó estão dentro do intervalo e serão incluídos no conjunto de resposta, evitando assim comparar cada imagem.

5.6 Considerações Finais

Neste capítulo foi detalhado um estudo experimental exemplificando o uso do RAFIKI em uma aplicação para a área médica, abordando os aspectos de armazenamento, desempenho e integração com outras ferramentas de análise. Para isso, o sistema foi aplicado em três cenários hipotéticos, no contexto médico. A recuperação de imagens baseada em conteúdo pode ajudar o especialista durante um determinado diagnóstico, recuperando casos anteriores, que podem possuir informações adicionais. Os cenários foram realizados com o intuito de beneficiar-se dos recursos oferecidos por um SGBDR atuando em conjunto com o RAFIKI, explorando um MAM para o retorno das consultas por similaridade. Verificou-se que o RAFIKI tanto facilita a construção de consultas e aplicativos, que facilitam explorar a semântica do processamento envolvido, quanto agiliza a execução das consultas.

Neste trabalho foi montado toda uma estrutura para que o RAFIKI pudesse receber uma imagem de entrada, e dessa forma extrair automaticamente seus vetores de características, permitindo assim indexar e organizar as informações em um SGBDR. Além disso, as consultas foram realizadas utilizando-se o SQL proposto, para a chamada da Slim-Tree. Utilizando essa estrutura, foi possível analisar a eficiência do RAFIKI em comparação com o KIARA. Ao avaliar o desempenho do RAFIKI, utilizando um conjunto de dados de mamografias, o sistema foi até seis vezes mais rápido que o KIARA, seu concorrente direto. Mesmo nos piores casos, a ferramenta proposta foi mais rápido do que o KIARA em 38%.

CAPÍTULO

6

CONCLUSÕES

Neste trabalho de mestrado foi desenvolvida uma técnica para estender o SGBDR (PostgreSQL) que permite tanto a representação, quanto a execução de consultas por similaridade utilizando um MAM (Slim-Tree). Com o intuito de avaliar a técnica proposta, foi desenvolvido o RAFIKI, uma ferramenta que é baseada na KIARA, criando um novo módulo para a inclusão de um MAM. Com o RAFIKI, uma série de testes foram desenvolvidos, avaliando tanto a aplicação do método em diferentes cenários, quanto a respeito de seu desempenho em comparação com a KIARA, sua antecessora. A análise apresentada no Capítulo 5 mostra que o RAFIKI possuiu desempenho superior à KIARA, no contexto de aplicações médicas. O presente trabalho abre portas para novas pesquisas, para utilizar um MAM para indexação de dados em um SGBDR.

6.1 Contribuições

Neste trabalho foi abordado o problema de inclusão do MAM *Slim-Tree* no SGBDR PostgreSQL. Como passo inicial, foi utilizada a ferramenta KIARA, que já estendia o PostgreSQL, e continha extratores de características, funções de distância, e as consultas SQL necessárias para se utilizar de tais recursos. A KIARA foi estendida, adicionando nela um módulo para o MAM, resultando assim no RAFIKI. Essa extensão incluiu um módulo intermediário que permite indexar os vetores de características previamente extraídos, armazenados em relações do banco de dados. A indexação utiliza funções de distância previamente definidas no banco. Além disso, foram definidas as funções em SQL, necessárias para a criação, população e consulta na Slim-Tree, que utilizam funções de distância e extratores de características.

Como contribuição para o presente trabalho foi definido um método que descreve, passo-a-passo, como deve ser realizada a inclusão de um MAM em um SGBDR. Esse método poderá ser utilizado em trabalhos futuros, para incluir novos MAMs no PostgreSQL ou em outros SGBDRs, com modificações específicas que se façam necessárias. O código fonte resultante das implementações realizadas durante este projeto de Mestrado estão disponíveis no repositório do

grupo de pesquisa GBdI: https://bitbucket.org/gbdi/.

6.2 Publicações

A seguir são apresentados os artigos publicados durante o desenvolvimento deste trabalho de Mestrado. Na publicação principal foi proposto o RAFIKI, que foi validado em uma aplicação do cenário médico. Além disso, foram publicados dois trabalhos resultantes de colaboração com o grupo de pesquisa.

Publicação principal:

- CBMS 2018 - Qualis-CC Capes B1 (NESSO-JR et al. , 2018): NESSO-JR, M. R.; CAZZOLATO, M. T.; SCABORA, L. C.; OLIVEIRA, P. H.; SPADON, G.; SOUZA, J. A.; OLIVEIRA, WILLIAN; RODRIGUES-Jr., J. F.; TRAINA, A. J. M.; TRAINA-Jr, C.. RAFIKI: Retrieval-Based Application for Imaging and Knowledge Investigation. In: 31st IEEE Internatinal Symposium on Computer-Based Medical Systems (CBMS). Karlstad, Sweeden. June 18-21, 2018. p. 71–76.

Contribuições resultantes de colaborações com o grupo de pesquisa.

- Journal CMPB 2019 - Qualis-CC Capes A2 (CAZZOLATO et al. , 2019): CAZZOLATO, M. T.; SCABORA, L. C.; NESSO-JR., M. R.; MILANO-OLIVEIRA, L. F.; COSTA, A. F.; KASTER, D. S.; KOENIGKAN-SANTOS, M.; AZEVEDO-MARQUES, P. M.; TRAINA-Jr, C.; TRAINA, A. J. M.. dp -BREATH: Heat maps and probabilistic classification assisting the analysis of abnormal lung regions. Journal of Computer Methods and Programs in Biomedicine (CMPB), May, 2019. p. 27–34.

- CBMS 2017 - Qualis-CC Capes B1 (CAZZOLATO et al. , 2017): CAZZOLATO, M. T.; SCABORA, L. C.; COSTA, A. F.; NESSO-JR., M. R.; MILANO-OLIVEIRA, L. F.; KASTER, D. S.; TRAINA-Jr, C.; TRAINA, A. J. M.. BREATH: Heat Maps Assisting the Detection of Abnormal Lung Regions in CT Scans. In: 2017 IEEE 30th International Symposium on Computer-Based Medical Systems (CBMS). Thessaloniki, Greece. June 22-24, 2017. p. 248–253.

6.3 Trabalhos futuros

Dentre os trabalhos futuros que podem ser desenvolvidos, a partir das contribuições deste projeto de mestrado, podem ser citados os seguintes:

- **Adicionar novos MAMs em SGBDR usando o método proposto:** Incluir diferentes MAMs, para que assim possa-se usufruir do melhor que cada um deles oferecem para a realização de consultas por similaridade.
- **Refinar o método proposto:** Modularizar os componentes, de forma que o trabalho se torne genérico o suficiente para ser aplicado em diferentes SGBDRs.
- **Testar o método em outros SGBDRs:** A partir do momento que for possível deixar o método aplicável a outros SGBDRs, comparar as vantagens e desvantagens de se utilizar outros MAMs.

REFERÊNCIAS

AREF, W. G.; ILYAS, I. F. An extensible index for spatial databases. In: **Proceedings of the 13th International Conference on Scientific and Statistical Database Management, July 18-20, 2001, George Mason University, Fairfax, Virginia, USA**. [s.n.], 2001. p. 49–58. Disponível em: <https://doi.org/10.1109/SSDM.2001.938537>. Citado na página 37.

ATHANASIOU, N.; VASSILAKOPOULOS, M.; CORRAL, A.; MANOLOPOULOS, Y. Use-based optimization of spatial access methods. In: **Proceedings of the 9th International Conference on Management of Digital EcoSystems, MEDES 2017, Bangkok, Thailand, November 07-10, 2017**. [s.n.], 2017. p. 65–72. Disponível em: <https://doi.org/10.1145/3167020.3167030>. Citado na página 28.

BARIONI, M. C. N.; KASTER, D. S.; RAZENTE, H. L.; TRAINA, A. J. M.; TRAINA-JR., C. Querying Multimedia Data by Similarity in Relational DBMS. In: YAN, L.; MA, Z. (Ed.). **Advanced Database Query Systems: Techniques, Applications and Technologies**. Hershey, PA, USA: IGI Global, 2011. p. 323–359. Citado na página 28.

BARIONI, M. C. N.; RAZENTE, H.; TRAINA, A.; TRAINA JR., C. Siren: A similarity retrieval engine for complex data. In: **Proceedings of the 32Nd International Conference on Very Large Data Bases**. [S.l.]: VLDB Endowment, 2006. (VLDB '06), p. 1155–1158. Citado nas páginas 38 e 43.

BARIONI, M. C. N.; RAZENTE, H. L.; TRAINA JR, A. J. M.; TRAINA, C. Seamlessly integrating similarity queries in sql. **Softw. Pract. Exper.**, John Wiley & Sons, Inc., New York, NY, USA, v. 39, n. 4, p. 355–384, mar. 2009. ISSN 0038-0644. Citado nas páginas 38, 40 e 43.

BBDO, A. 2017. Disponível em: <http://cerebrocriativo.blogspot.com.br/2006/02/to-remember-cesar.html>. Acesso em: 07/01/2017. Citado na página 25.

BLADE, S. 2017. Disponível em: <https://socialblade.com/>. Acesso em: 07/01/2017. Citado na página 17.

BLANCO, G.; BEDO, M. V. N.; CAZZOLATO, M. T.; SANTOS, L. F. D.; JORGE, A. E. S.; JR., C. T.; AZEVEDO-MARQUES, P. M.; TRAINA, A. J. M. A label-scaled similarity measure for content-based image retrieval. In: **IEEE International Symposium on Multimedia, ISM 2016, San Jose, CA, USA, December 11-13, 2016**. [s.n.], 2016. p. 20–25. Disponível em: <https://doi.org/10.1109/ISM.2016.0014>. Citado na página 22.

BUGATTI, P. H.; KASTER, D. S.; PONCIANO-SILVA, M.; JR., C. T.; AZEVEDO-MARQUES, P. M.; TRAINA, A. J. M. Prosper: Perceptual similarity queries in medical CBIR systems through user profiles. **Comp. in Bio. and Med.**, v. 45, p. 8–19, 2014. Disponível em: <https://doi.org/10.1016/j.compbiomed.2013.11.015>. Citado na página 24.

CAZZOLATO, M. T.; SCABORA, L. C.; COSTA, A. F.; NESSO-JR, M. R.; MILANO-OLIVEIRA, L. F.; KASTER, D. S.; TRAINA-JR, C.; TRAINA, A. J. M. Breath: Heat maps assisting the detection of abnormal lung regions in ct scans. In: **International Symposium on**

Computer-Based Medical Systems (CBMS). Thessaloniki, Greece: [s.n.], 2017. Citado nas páginas 25 e 56.

CAZZOLATO, M. T.; SCABORA, L. C.; JR., M. R. N.; OLIVEIRA, L. F. M.; COSTA, A. F.; KASTER, D. S.; KOENIGKAM-SANTOS, M.; MARQUES, P. M. de A.; JR., C. T.; TRAINA, A. J. M. *dp*-breath: Heat maps and probabilistic classification assisting the analysis of abnormal lung regions. **Computer Methods and Programs in Biomedicine**, v. 173, p. 27–34, 2019. Disponível em: <https://doi.org/10.1016/j.cmpb.2019.01.014>. Citado na página 56.

CIACCIA, P.; PATELLA, M. Bulk loading the m-tree. In: **ADC Australasian Database Conference**. [S.l.: s.n.], 1998. p. 15–26. Citado na página 29.

DATTA, R.; JOSHI, D.; LI, J.; WANG, J. Z. Image retrieval: Ideas, influences, and trends of the new age. **ACM Comput. Surv.**, v. 40, n. 2, p. 5:1–5:60, 2008. Disponível em: <http://doi.acm.org/10.1145/1348246.1348248>. Citado na página 25.

DOMENICONI, C.; GUNOPULOS, D.; MA, S.; YAN, B.; AL-RAZGAN, M.; PAPADOPOULOS, D. Locally adaptive metrics for clustering high dimensional data. **Data Mining and Knowledge Discovery**, v. 14, n. 1, p. 63–97, 2007. Citado na página 27.

ELTABAKH, M. Y.; ELTARRAS, R.; AREF, W. G. Space-partitioning trees in postgresql: Realization and performance. In: **Proceedings of the 22nd International Conference on Data Engineering, ICDE 2006, 3-8 April 2006, Atlanta, GA, USA**. [s.n.], 2006. p. 100. Disponível em: <https://doi.org/10.1109/ICDE.2006.146>. Citado na página 37.

FLEITES, F.; CHEN, S. Efficient content-based multimedia retrieval using novel indexing structure in postgresql. In: **2013 IEEE International Symposium on Multimedia, ISM 2013, Anaheim, CA, USA, December 9-11, 2013**. [s.n.], 2013. p. 500–501. Disponível em: <https://doi.org/10.1109/ISM.2013.96>. Citado na página 37.

GROUP, T. P. G. D. **PostgreSQL 9.6.3 Documentation**. [S.l.], 2016. Citado nas páginas 35 e 36.

GULIATO, D.; MELO, E. V. de; RANGAYYAN, R. M.; SOARES, R. C. POSTGRESQL-IE: an image-handling extension for postgresql. **J. Digital Imaging**, v. 22, n. 2, p. 149–165, 2009. Disponível em: <https://doi.org/10.1007/s10278-007-9097-5>. Citado nas páginas 38 e 43.

HELLERSTEIN, J. M.; NAUGHTON, J. F.; PFEFFER, A. Generalized search trees for database systems. In: **VLDB'95, Proceedings of 21th International Conference on Very Large Data Bases, September 11-15, 1995, Zurich, Switzerland.** [s.n.], 1995. p. 562–573. Disponível em: <http://www.vldb.org/conf/1995/P562.PDF>. Citado na página 36.

JAGADISH, H. V.; MENDELZON, A. O.; MILO, T. Similarity-based queries. In: **ACM Symp. on Principles of Database Systems (PODS)**. San Jose, CA: ACM Press, 1995. p. 36–45. Citado na página 26.

JTC1/SC29/WG11, I. **Introduction to MPEG-7**. [S.l.], 2000. Citado na página 26.

KASTER, D. S.; BUGATTI, P. H.; TRAINA, A. J. M.; JR., C. T. Fmi-sir: A flexible and efficient module for similarity searching on oracle database. **JIDM**, v. 1, n. 2, p. 229–244, 2010. Citado nas páginas 39, 41 e 43.

KEOGH, E. J. Exact indexing of dynamic time warping. In: **VLDB 2002, Proceedings of 28th International Conference on Very Large Data Bases, August 20-23, 2002, Hong Kong, China**. [s.n.], 2002. p. 406–417. Disponível em: <http://www.vldb.org/conf/2002/S12P01.pdf>. Citado na página 26.

KORNACKER, M.; SHAH, M. A.; HELLERSTEIN, J. M. Amdb: A design tool for access methods. **IEEE Data Eng. Bull.**, v. 26, n. 2, p. 3–11, 2003. Disponível em: <http://sites.computer.org/debull/A03june/hellersteinF.ps>. Citado na página 36.

LIMA, E. L. **Espacos Metricos**. [S.l.]: Instituto de Matemetica Pura e Aplicada, 1993. Citado na página 26.

LU, W.; HOU, J.; YAN, Y.; ZHANG, M.; DU, X.; MOSCIBRODA, T. MSQL: efficient similarity search in metric spaces using SQL. **VLDB J.**, v. 26, n. 6, p. 829–854, 2017. Disponível em: <https://doi.org/10.1007/s00778-017-0481-6>. Citado nas páginas 40, 42 e 43.

MALIK, R.; KIM, S.; JIN, X.; RAMACHANDRAN, C.; HAN, J.; GUPTA, I.; NAHRSTEDT, K. Mlr-index: An index structure for fast and scalable similarity search in high dimensions. In: WINSLETT, M. (Ed.). **21st International Conference on Scientific and Statistical Database Management, SSDBM 2009**. New Orleans, LA: Springer, 2009. (Lecture Notes in Computer Science, v. 5566), p. 167–184. Citado na página 26.

NESSO Jr., M. R.; CAZZOLATO, M. T.; SCABORA, L. C.; OLIVEIRA, P. H.; SPADON, G.; SOUZA, J. A. de; OLIVEIRA, W. D.; CHINO, D. Y. T.; RODRIGUES-JR., J. F.; TRAINA, A. J. M.; TRAINA-JR., C. RAFIKI: Retrieval-based application for imaging and knowledge investigation. In: **2018 IEEE 31st International Symposium on Computer-Based Medical Systems (CBMS)**. [S.l.: s.n.], 2018. p. 71–76. ISSN 2372-9198. Citado nas páginas 53, 54, 55, 59, 61, 62 e 64.

OLIVEIRA, P. H.; FRAIDEINBERZE, A. C.; LAVERDE, N. A.; GUALDRON, H.; GONZAGA, A. S.; FERREIRA, L. D.; OLIVEIRA, W. D.; JR., J. F. R.; CORDEIRO, R. L. F.; JR., C. T.; TRAINA, A. J. M.; SOUSA, E. P. M. de. On the support of a similarity-enabled relational database management system in civilian crisis situations. In: **ICEIS 2016 - Proceedings of the 18th International Conference on Enterprise Information Systems, Volume 1, Rome, Italy, April 25-28, 2016**. [s.n.], 2016. p. 119–126. Disponível em: <https://doi.org/10.5220/0005816701190126>. Citado nas páginas 18, 41, 42 e 43.

OLIVEIRA, P. H.; JR., C. T.; KASTER, D. S. Clap, ACIR and SCOOP: novel techniques for improving the performance of dynamic metric access methods. **Inf. Syst.**, v. 72, p. 117–135, 2017. Disponível em: <https://doi.org/10.1016/j.is.2017.10.003>. Citado na página 29.

OLIVEIRA, P. H.; SCABORA, L. C.; CAZZOLATO, M. T.; BEDO, M. V. N.; TRAINA, A. J. M.; TRAINA-JR., C. MAMMOSET: An Enhanced Dataset of Mammograms. In: **Proceedings of the Satellite Events of the 32nd Brazilian Symposium on Databases**. [S.l.]: SBC, 2017. p. 256–266. Citado na página 63.

PAPAPETROU, P.; ATHITSOS, V.; POTAMIAS, M.; KOLLIOS, G.; GUNOPULOS, D. Embedding-based subsequence matching in time-series databases. **ACM Transactions on Database Systems (TODS)**, v. 36, n. 3, p. 1–39, 2011. Citado na página 26.

POLA, I. R. V.; TRAINA, A. J. M.; TRAINA CAETANO, J. Easing the dimensionality curse by stretching metric spaces. In: WINSLETT, M. (Ed.). **21st International Conference on**

Scientific and Statistical Database Management, SSDBM 2009. New Orleans, LA: Springer, 2009. (Lecture Notes in Computer Science, v. 5566), p. 417–434. Citado na página 26.

SALEMBIER, P.; SIKORA, T. **Introduction to MPEG-7: Multimedia Content Description Interface**. New York, NY, USA: John Wiley & Sons, Inc., 2002. ISBN 0471486787. Citado na página 25.

SAMET, H. **Foundations of Multidimensional and metric Data Structures**. San Francisco, CA: Morgan Kaufmann Publishers, 2006. Citado na página 26.

SANTOS, L. F. D.; DIAS, R. L.; RIBEIRO, M. X.; TRAINA, A. J. M.; JR., C. T. Combining diversity queries and visual mining to improve content-based image retrieval systems: The divi method. In: **2015 IEEE International Symposium on Multimedia, ISM 2015, Miami, FL, USA, December 14-16, 2015**. [s.n.], 2015. p. 357–362. Disponível em: <https://doi.org/10.1109/ISM.2015.115>. Citado nas páginas 24 e 25.

SHAH, M. A.; KORNACKER, M.; HELLERSTEIN, J. M. Amdb: A visual access method development tool. In: **UIDIS**. [s.n.], 1999. p. 130–140. Disponível em: <https://doi.org/10.1109/UIDIS.1999.791469>. Citado na página 36.

SHARIF, U.; MEHMOOD, Z.; MAHMOOD, T.; JAVID, M. A.; REHMAN, A.; SABA, T. Scene analysis and search using local features and support vector machine for effective content-based image retrieval. **Artif. Intell. Rev.**, v. 52, n. 2, p. 901–925, 2019. Disponível em: <https://doi.org/10.1007/s10462-018-9636-0>. Citado na página 24.

SIMOES, R. E. O.; QUEIROZ, G. R. de; FERREIRA, K. R.; VINHAS, L.; CÂMARA, G. Postgis-t: towards a spatiotemporal postgresql database extension. In: **XVII Brazilian Symposium on Geoinformatics - GeoInfo 2016, Campos do Jordão, SP, Brazil, November 27-30, 2016.** [s.n.], 2016. p. 252–262. Disponível em: <http://urlib.net/8JMKD3MGP3W34P/3N2UAT8>. Citado na página 37.

SKOPAL, T. Pivoting m-tree: A metric access method for efficient similarity search. In: **Proceedings of the Dateso 2004 Annual International Workshop on DAtabases, TExts, Specifications and Objects, Desna, Czech Republic, April 14-16, 2004**. [s.n.], 2004. p. 27–37. Disponível em: <http://ceur-ws.org/Vol-98/paper3.pdf>. Citado na página 30.

SKOPAL, T.; POKORNÝ, J.; KRÁTKÝ, M.; SNÁSEL, V. Revisiting m-tree building principles. In: KALINICHENKO, L. A.; MANTHEY, R.; THALHEIM, B.; WLOKA, U. (Ed.). **Advances in Databases and Information Systems, 7th East European Conference, ADBIS 2003**. Dresden, Germany: Springer, 2003. (Lecture Notes in Computer Science, v. 2798), p. 148–162. Citado nas páginas 29 e 30.

TEKLI, J.; CHBEIR, R.; TRAINA, A. J. M.; TRAINA CAETANO, J. Xml document-grammar comparison: Related problems and applications. **Central European Journal of Computer Science**, v. 1, n. 1, p. 117–136, 2011. Citado na página 26.

THOMAS, M.; CARSON, C.; HELLERSTEIN, J. M. Creating a customized access method for blobworld. In: **Proceedings of the 16th International Conference on Data Engineering, San Diego, California, USA, February 28 - March 3, 2000**. [s.n.], 2000. p. 82. Disponível em: <https://doi.org/10.1109/ICDE.2000.839390>. Citado na página 36.

TRAINA, A. J. M.; TRAINA CAETANO, J.; BALAN, A. G. R.; RIBEIRO, M. X.; BUGATTI, P. H.; WATANABE, C. Y. V.; AZEVEDO-MARQUES, P. M. d. Feature extraction and selection for decision making over medical images. In: DESERNO, T. M. (Ed.). **Biomedical Image Processing - Methods and Applications**. [S.l.]: Springer-Verlag, 2010. p. 181–209. Citado na página 26.

TRAINA-JR., C.; TRAINA, A. J. M.; SEEGER, B.; FALOUTSOS, C. Slim-trees: High performance metric trees minimizing overlap between nodes. In: **Advances in Database Technology - EDBT 2000, 7th International Conference on Extending Database Technology, Konstanz, Germany, March 27-31, 2000, Proceedings**. [s.n.], 2000. p. 51–65. Disponível em: <https://doi.org/10.1007/3-540-46439-5_4>. Citado nas páginas 30, 32 e 33.

WILSON, D. R.; MARTINEZ, T. R. Improved heterogeneous distance functions. **Journal of Artificial Intelligence Research**, v. 6, p. 1–34, 1997. Citado na página 26.

ZEZULA, P.; AMATO, G.; DOHNAL, V.; BATKO, M. **Similarity Search - The Metric Space Approach**. Kluwer, 2006. v. 32. (Advances in Database Systems, v. 32). ISBN 978-0-387-29146-8. Disponível em: <https://doi.org/10.1007/0-387-29151-2>. Citado nas páginas 29 e 30.

ZHOU, X.; WANG, G.; YU, J. X.; YU, G. M+-tree : A new dynamical multidimensional index for metric spaces. In: **Database Technologies 2003, Proceedings of the 14th Australasian Database Conference, ADC 2003, Adelaide, South Australia, February 2003**. [s.n.], 2003. p. 161–168. Disponível em: <http://crpit.com/confpapers/CRPITV17Zhou.pdf>. Citado na página 30.


ICMC
USP
SÃO CARLOS