UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DAINF - DEPARTAMENTO ACADÊMICO DE INFORMÁTICA
CURSO DE ENGENHARIA DE COMPUTAÇÃO

CRISTIANO JOSÉ MENDES MATSUI

CONSULTAS POR SIMILARIDADE EM BASES DE DADOS COMPLEXOS UTILIZANDO TÉCNICA OMNI EM SGBDR

TRABALHO DE CONCLUSÃO DE CURSO

PATO BRANCO
2018

CRISTIANO JOSÉ MENDES MATSUI

CONSULTAS POR SIMILARIDADE EM BASES DE DADOS COMPLEXOS UTILIZANDO TÉCNICA OMNI EM SGBDR

Trabalho de Conclusão de Curso apresentado ao Curso de Engenharia de Computação da Universidade Tecnológica Federal do Paraná, como requisito parcial para a obtenção do título de Bacharel em Engenharia de Computação.

Orientador: Dr. Ives Renê Venturini Pola

Coorientadora: Dra. Fernanda Paula Barbosa Pola

PATO BRANCO
2018

RESUMO

MATSUI, Cristiano. Consultas por similaridade em bases de dados complexos utilizando técnica OMNI em SGBDR. 2018. 26 f. Trabalho de Conclusão de Curso – Curso de Engenharia de Computação, Universidade Tecnológica Federal do Paraná. Pato Branco, 2018.

A necessidade de armazenamento de mídias cada vez maiores em termos de tamanho de armazenamento e complexas é uma tendência que aumentou consideravelmente com os avanços da tecnologia e comunicação. Estes dados conhecidos como dados complexos exigem uma complexidade estrutural de armazenamento e análise maior do que dados simples como palavras ou números, além de requererem operadores especiais de consultas, como a consulta por abrangência - *range query* (Rq) e a consulta aos k-vizinhos mais próximos - *k-nearest neighbors query* (kNNq). Dentre o conjunto de dados complexos destacam-se as imagens, que precisam ser comparadas de acordo com características extraídas como cor, forma ou textura. Esta comparação é realizada na forma de um cálculo de distância entre o valor da característica da imagem central da consulta em relação a todas as outras imagens da base de dados. O tempo de consulta aumenta significativamente com o aumento da base de dados e da complexidade destes. Para contornar o problema da maldição da cardinalidade, este trabalho tem como proposta a aplicação da técnica OMNI utilizada para promover uma etapa de filtragem do número de imagens a terem as suas distâncias calculadas, evitando a comparação com toda a base de dados. Esta técnica possui custos adicionais de espaço de armazenamento para as suas estruturas, mas consegue acelerar consultas por similaridade, reduzindo o tempo de execução necessário.

Palavras-chave: Dados Complexos. OMNI. CBIR. Rq. kNNq. Otimização.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

GBDI	Grupo de Bases de Dados e Imagens
IOid(s_i)	Identificador do objeto s_i
kNNq	Consulta aos k-vizinhos mais próximos
mbOr	Região Omni de delimitação mínima
Rq	Consulta por abrangência
SGBDR	Sistema Gerenciador de Banco de Dados Relacional
SQL	*Structured Query Language*

LISTA DE SÍMBOLOS

$\in$	Pertence
$\forall$	Para todo
s_q	Elemento central da consulta
ξ	Raio da consulta por abrangência
$\mathbb{S}$	Domínio de dados
S	Conjunto de elementos pertencentes ao domínio $\mathbb{S}$
k	Número de vizinhos desejados em uma *kNNq*
s_i	Elemento pertencente ao conjunto S
$d : \mathbb{S} \times \mathbb{S} \rightarrow \mathbb{R}^+$	Métrica ou função de distância
$M<S,d>$	Espaço métrico
$\mathscr{F}$	Base de focos Omni
f_k	Foco pertencente a base focal
D	Dimensão intrínseca da base de dados

LISTA DE ALGORITMOS

SUMÁRIO

1 INTRODUÇÃO

Neste capítulo será apresentada uma contextualização do problema, assim como o estado da arte abordado por este trabalho. Também será introduzido um tipo de consulta não-nativo ao SGBDR, a consulta por similaridade, assim como os tipos de consultas por similaridade que serão utilizados neste trabalho como a consulta por abrangência e a consulta por k-vizinhos mais próximos. Finalmente, será apresentada a organização deste documento.

1.1 CONSIDERAÇÕES INICIAIS

Nos anos recentes foi notado um grande aumento no tráfego e armazenamento de diferentes aplicações e dados multimídias, como imagens, áudio, vídeo, impressões digitais, séries temporais, sequências de proteínas, etc. Estes tipos de dados, que apresentam muito mais atributos do que simples numerais ou pequenas cadeias de caracteres, são conhecidos como dados complexos (ZIGHED et al., 2008).

Quando tratados por um Sistema Gerenciador de Banco de Dados Relacional (SGBDR), não suportam comparações com os operadores conhecidos como "big six" da linguagem SQL: $=$, $\neq$, $<$, $>$, $\leq$, $\geq$. Esse fato limita muito as comparações entre dados complexos inseridos em um SGBDR, ocasionando um grande problema no contexto de base de dados, uma vez que os principais sistemas de gerenciamento de base de dados são relacionais (DB-ENGINES, 2017). Com isso, tornou-se necessária a concepção de novos tipos de comparadores, como buscas por similaridade.

Estas consultas por similaridade se aplicam de maneira geral a muitos dos tipos de dados complexos (BARIONI et al., 2009). Exemplos na área médica são encontrados nos trabalhos de (MARCHIORI et al., 2001), (BUGATTI et al., 2014) e (LEHMANN et al., 1999). Também é possível encontrar trabalhos no campo de reconhecimento facial (GUTTA; WECHSLER, 1997) e sistemas de identificação por biometria (CHORAS, 2007).

1.2 TIPOS DE CONSULTAS

Dentre os operadores de consulta por similaridade os mais comuns são as consultas por abrangência (*range query: Rq*) e consulta aos k-vizinhos mais próximos (*k-nearest neighbor querry: kNNq*).

As consultas por abrangência *Rq*(s_q, ξ) recebem como parâmetro um elemento s_q do domínio de dados (elemento central da consulta) e um limite máximo de dissimilaridade ξ. O resultado é o conjunto de elementos da base que diferem do elemento central da consulta por no máximo a dissimilaridade indicada.

Definição 1. *Seja S um atributo complexo de um domínio $\mathbb{S}$ ($S \subset \mathbb{S}$) sobre o qual a condição de similaridade é expressada, seja d uma função de distância, seja ξ o limiar de dissimilaridade e seja $s_q \in \mathbb{S}$ o elemento central de consulta. A consulta Rq(s_q , ξ) retorna todos os elementos $s_i \in \mathbb{S}$ que possuem o valor do atributo S distantes até um máximo de ξ deste atributo referente ao elemento central da consulta:*

$$Rq(s_q, \xi) : S = \{s_i \in S \mid d(s_i, s_q) \leq \xi\} \tag{1}$$

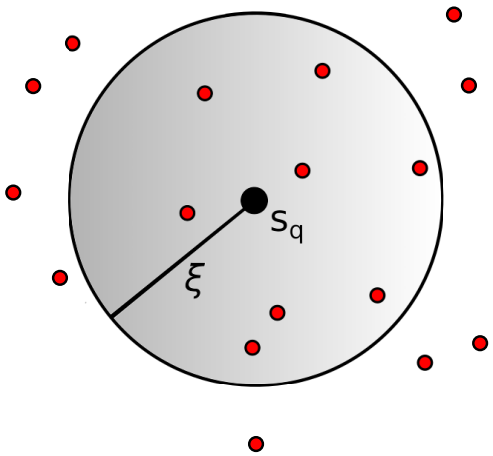


Figura 1 – Exemplo de consulta por abrangência

Fonte: Autoria Própria

A Figura 1 exemplifica este tipo de consulta. Tomando s_q e ξ como parâmetros da consulta, todos os elementos contidos pelo raio de abrangência fazem parte do conjunto resposta da consulta. O elemento s_q não necessariamente precisa pertencer ao conjunto de dados de pesquisa, devendo apenas pertencer ao mesmo domínio de dados.

Uma consulta aos k-vizinhos mais próximos *kNNq(s_q, k)* também recebe como um de seus parâmetros um elemento central da consulta s_q, e um número inteiro k de vizinhos desejados, e retorna como resultado o conjunto dos k elementos com a menor dissimilaridade em relação ao elemento central da consulta s_q (POLA, 2010).

Definição 2. *Seja S um atributo complexo de um domínio $\mathbb{S}$ sobre o qual a condição de similaridade é expressada, seja d uma função de distância, seja $k \in \mathbb{N}^*$ a quantidade de elementos desejados e seja $s_q \in \mathbb{S}$ o elemento central de consulta. A consulta kNNq(s_q , k) retorna k elementos $s_i \in \mathbb{S}$ que possuem o valor do atributo S menos distantes do valor deste atributo referente ao elemento central da consulta(FERREIRA et al., 2009):*

$$kNNq(s_q, k) : S = \{s_i \in S \mid \forall\ s_j \in S - S', d(s_i, s_q) \leq d(s_j, s_q)\}, \tag{2}$$

onde $S' = 0$, se $i = 1$ e $S' = \{s_1, ..., s_{(i-1)}\}$, se $1 < i \leq k$.

A Figura 2 exemplifica este tipo de operação. Nesta consulta, os parâmetros foram o elemento central da consulta s_q e o número de elementos k a serem encontrados igual a 4. Os elementos ligados ao elemento central foram os mais próximos a este, portanto apenas eles fazem parte do conjunto resposta da consulta. Note que o elemento central da consulta não precisa pertencer ao conjunto de dados, como é o caso exemplificado.

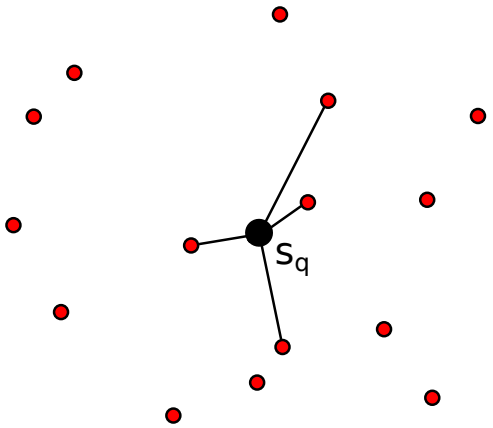


Figura 2 – Exemplo de consulta por k-vizinhos mais próximos

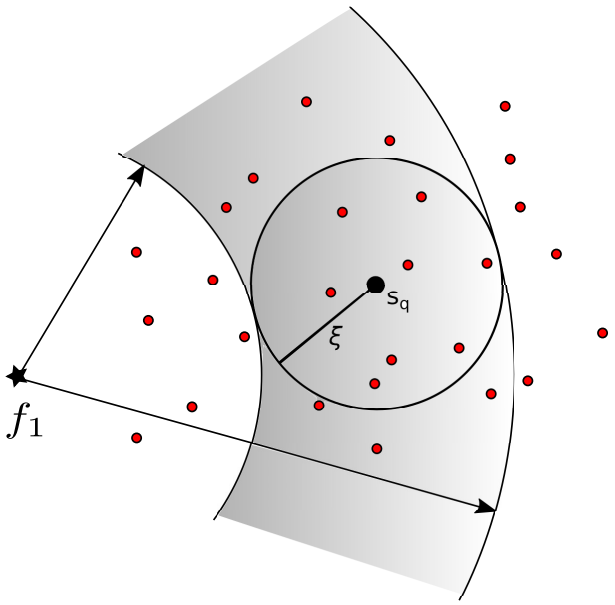
Fonte: Autoria Própria

O SGBDR não possui suporte nativo a estes tipos de consulta, mas é possível construir estas consultas utilizando ferramentas existentes em um banco de dados relacional (como a estrutura de dados *B-tree*).

A solução abordada por esta proposta é a do uso de técnica OMNI, presente no trabalho de (SANTOS FILHO et al., 2001). Um número calculado de elementos do conjunto de dados são selecionados como "focos", e utilizados para evitar cálculos desnecessários de distâncias, fazendo uso da desigualdade triangular.

A técnica aqui aplicada tem base em calcular previamente as distâncias de todos os elementos para todos os focos selecionados, armazenando estas distâncias no banco. Quando uma consulta por similaridade (como uma consulta por abrangência) é realizada, são conhecidas as distâncias entre o elemento central da consulta s_q e o raio de abrangência ξ. Considerando um foco f_1 e utilizando a desigualdade triangular, elementos que possuem uma distância entre o foco escolhido menor do que a distância de s_q até o foco menos o valor ξ serão descartados do conjunto de elementos necessários para os cálculos de distância com o elemento central. Simetricamente, elementos cuja distância até o foco seja maior do que a distância de s_q até o foco mais o valor do raio de abrangência ξ também serão descartados. Com isso, ocorre uma grande redução do número de cálculos necessários para fornecer o conjunto resposta. Essa poda também pode ser realizada por mais de um foco.

Figura 3 – Consulta por abrangência pela técnica OMNI utilizando um foco



Fonte: Autoria Própria

A Figura 3 ilustra a poda no número de cálculos. Apenas os elementos na área sombreada terão as suas distâncias em relação ao centro da consulta calculadas, pois estão no conjunto de elementos que não foram descartados utilizando a desigualdade triangular com as distâncias previamente calculadas em relação ao foco. O armazenamento das distâncias de cada foco f_i para cada outro elemento s_k será feito utilizando uma estrutura de indexação que implementa os conceitos da técnica OMNI com a estrutura da B-tree, originando uma nova estrutura chamada de OMNIB-Tree. As distâncias serão armazenadas em l OMNIB-Trees, sendo l o número de focos criados para a base de dados (SANTOS FILHO et al., 2001).

1.3 OBJETIVOS

Os objetivos encontram-se divididos em objetivos gerais, referentes ao resultado obtido com a conclusão deste trabalho e objetivos específicos, ilustrando etapas intermediárias necessárias para alcançar o objetivo geral.

1.3.1 OBJETIVOS GERAIS

O principal foco deste trabalho foi a construção de um sistema de consultas em SGBDR por similaridade em uma base de imagens utilizando a técnica OMNI para reduzir o custo computacional das operações de consulta.

1.3.2 OBJETIVOS ESPECÍFICOS

- Modelar o banco de dados para atender a problemática apresentada;

- Aplicar os extratores de características das imagens utilizadas;
- Inserir no banco de dados as imagens e os valores de suas características;
- Criar a estrutura OMNI necessária para a filtragem dos cálculos;
- Analisar e comparar os resultados obtidos.

1.4 ORGANIZAÇÃO DO TRABALHO

No Capítulo 2 é feito a explanação de alguns conceitos necessários para o entendimento e funcionamento da técnica. O Capítulo 3 descreve detalhadamente as minúncias da técnica OMNI, assim com as estruturas necessárias para a sua implementação. No Capítulo 4 é feito a ilustração e discussão dos resultados obtidos e no Capítulo 5 são expostas as conclusões, considerações finais deste trabalho e a indicação de possíveis trabalhos futuros.

2 CONCEITOS

O presente capítulo aborda os conceitos utilizados para tratar o problema apresentado no Capítulo 1, assim como uma explanação sobre os extratores de características de imagens que foram utilizados e o estado da arte atual.

2.1 ESPAÇOS MÉTRICOS

Uma métrica em um domínio $\mathbb{S}$ é uma função $d : \mathbb{S} \times \mathbb{S} \rightarrow \mathbb{R}^+$ que associa a cada par ordenado de elementos (s_1, s_2) um número real $d(s_1,s_2)$ chamado de distância de s_1 a s_2, e que atenda as propriedades definidas no espaço métrico(LIMA, 1977).

Um espaço métrico M é um par $<S, d>$ no qual S é um conjunto de elementos e d é uma métrica (ou função de distância). Esta distância $d(s_1,s_2)$ pode ser compreendida como uma medida de dissimilaridade entre dois elementos. Quanto menor esta distância entre dois elementos, mais semelhantes eles são.

Definição 3. *Seja S um conjunto não-vazio de elementos e $d(s_1,s_2)$ uma métrica definida sobre $\mathbb{S} \times \mathbb{S}$. O par $<S, d>$ é chamado de espaço métrico desde que d satisfaça as seguintes propriedades:*

1. *$d(s_1,s_1) = 0$ (identidade);*
2. *Se $s_1 \neq s_2$ então $d(s_1,s_2) > 0$ (não-negatividade);*
3. *$d(s_1,s_2) = d(s_2,s_1)$ (simetria);*
4. *$d(s_1,s_3) \leq d(s_1,s_2) + d(s_2,s_3)$ (desigualdade triangular)*

onde s_1, s_2, $s_3 \in \mathbb{S}$.

O conjunto de dados complexos a ser utilizado geralmente apresenta dimensões elevadas, mas também podem não ter dimensão fixa. Para a utilização computacional destes conceitos torna-se necessário o conceito de bola, empregado em espaços métricos. A seguinte definição é baseada no livro de (SHIRALI; VASUDEVA, 2005).

Definição 4. *Seja $<\mathbb{S}, d>$ um espaço métrico. O conjunto*

$$S(s_0,r) = x \in \mathbb{S} : d(s_0, x) \leq r, onde\ r > 0\ e\ s_0 \in \mathbb{S}, \tag{3}$$

é chamado de bola fechada de raio r e centro em s_0.

A aplicação direta das propriedades do espaço métrico neste trabalho reside fortemente na quarta propriedade apresentada pela Definição 3. A desigualdade triangular é fundamentada na geometria euclidiana, na qual a soma do comprimento de dois lados de um triângulo não pode ser superior ao comprimento do terceiro lado, como ilustra a Figura 4.

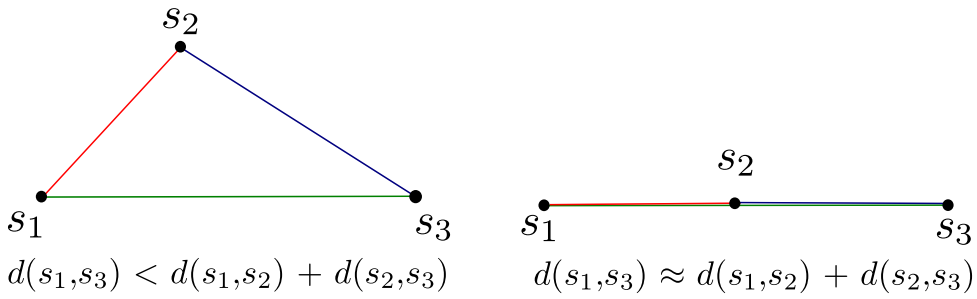


Figura 4 – Exemplo geométrico da desigualdade triangular

Fonte: Autoria Própria

O emprego da desigualdade triangular neste trabalho será detalhado no Capítulo 3.

2.2 FUNÇÕES DE DISTÂNCIA

Para verificar a similaridade entre dois elementos de um domínio, é utilizada uma função de distância. Esta função recebe como parâmetro um par de elementos do conjunto e retorna o valor da dissimilaridade entre eles. Quanto mais próximo de zero, mais similares os elementos são.

A importância do uso de uma função de distância para este trabalho é fornecer uma métrica de comparação entre os elementos complexos. Além de necessária para a consulta por similaridade, sua propriedade de desigualdade triangular é utilizada para evitar cálculos desnecessários, fornecendo uma métrica empregada pela técnica OMNI.

2.2.1 DISTÂNCIA MINKOWSKI

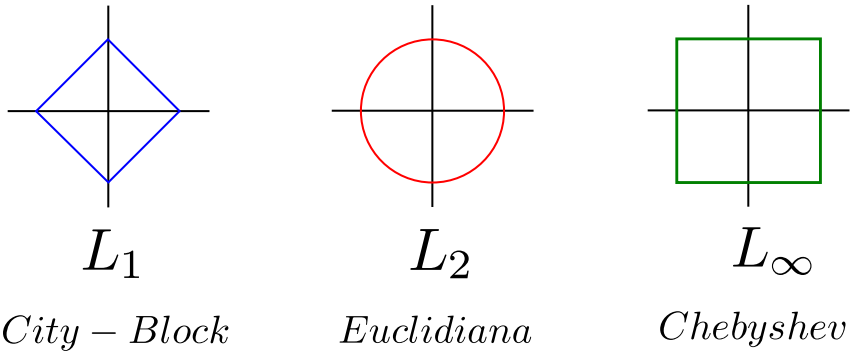
Dentre as funções de distância, será abordada a métrica Minkowski. Esta métrica é a mais utilizada para cálculos de índice de similaridade pois é independente da origem do espaço do conjunto de dados, e tem como resultado o valor da dissimilaridade entre dois elementos (JAIN; DUBES, 1988).

Definição 5. *Sejam* $s_1 = \{s_{11},s_{12},...,s_{1n}\}$ *e* $s_2 = \{s_{21},s_{22},...,s_{2n}\}$ *dois vetores de dimensionalidade* n *pertencentes ao conjunto de elementos* $\mathbb{S}$*, a distância Minkowski entre esses dois elementos é dada por:*

$$d(s_1,s_2) = \sqrt[p]{\sum_{i=1}^{n} |s_{1i} - s_{2i}|^p} \tag{4}$$

A Figura 5 ilustra as diferentes formas geométricas geradas de acordo com a métrica L_p utilizada. Para estes determinados valores de p, a distância Minkowski recebe nomes próprios (*distância City-Block* para $p = 1$, *distância Euclidiana* para $p = 2$ e *distância Chebyshev* para $p = \infty$).

Figura 5 – Círculos unitários em espaços bi-dimensionais para diferentes valores de p



Fonte: Autoria Própria

2.3 ESTRUTURAS DE INDEXAÇÃO

Dados complexos costumam apresentar tamanho físico muito elevado quando comparados com dados numéricos ou pequenas cadeias de caracteres, que são os tipos de dados mais comuns em bancos de dados tradicionais. Para responder consultas que envolvam dados complexos, são necessários mais acessos a disco em relação a uma consulta que envolva tipos de dados mais simples, como os previamente mencionados. Estes acessos são custosos e precisam ser reduzidos para um melhor desempenho do banco.

Uma maneira de tornar o acesso a disco mais eficiente é evitar movimentar grandes porções do banco de dados do disco para a memória, fazendo o uso de índices dentro do banco de dados. Um índice em um SGBDR funciona de maneira semelhante a um índice de um livro. Para procurar um tópico específico, é possível consultar o índice no fim do livro e descobrir qual o número da página correspondente ao tópico, contornando a necessidade da leitura sequencial do livro até o tópico procurado. Os índices são armazenados em ordem e apresentam um tamanho muito menor do que um capítulo do livro, reduzindo o esforço necessário para a sua consulta (SILBERSCHATZ; KORTH; SUDARSHAN, 2011).

Mas índices também podem ser empregados para uso em memória RAM. Enquanto as estruturas de indexação orientadas a disco são armazenadas em disco e apresentam um custo elevado para serem acessadas, índices orientados a memória estão contidos na memória RAM, assim não existem acessos a disco para serem minimizados. Assim, uma estrutura de indexação em memória busca reduzir o tempo de computação geral enquanto usa o mínimo de memória possível. Estas estruturas armazenam ponteiros para as tuplas, que são menos custosos de serem manipulados do que as tuplas (LEHMAN; CAREY, 1986).

Dentre as estruturas de indexação em disco para dados tradicionais, destacam-se a B^+-Tree e índices bitmap. Para indexação em memória, podem ser utilizadas estruturas como B-Trees e arrays para preservar a ordenação natural dos dados, enquanto estruturas de hashing (*Chained Bucket Hashing, Linear Hashing e Extendible Hashing*) aleatorizam os dados dentro do índice.

2.4 EXTRATORES DE CARACTERÍSTICAS

O principal foco deste trabalho é o emprego desta técnica para bases de dados constituídas por imagens. Para isto, torna-se necessário o uso de uma miríade de extratores de características das imagens, para um maior refinamento do uso de consultas por similaridade. Estas características podem se referir a: atributos visuais (cor, forma, textura), atributos lógicos (identificação de elementos) e atributos semânticos (identificação de emoções humanas).

As características visuais podem ser utilizadas como histogramas de cores para a análise de cor, matrizes de co-ocorrência para a análise de textura e métodos baseados em contorno para a análise de forma. Geralmente, consultas são feitas utilizando uma combinação destas características, e não apenas uma delas.

Dado uma palheta discreta de cores definida por alguns eixos de cor, o histograma de cores é obtido através da discretização das cores da imagem e contagem do número de vezes que cada cor discreta ocorre na matriz da imagem (SWAIN; BALLARD, 1991). As vantagens do uso do histograma de cores é a sua simplicidade computacional e pouca sensibilidade a alterações na imagem (rotação e translação), particularmente útil para a representação de objetos tridimensionais. Entretanto, duas imagens completamente diferentes podem apresentar o mesmo histograma de cores.

Para a análise de textura, o objetivo e conseguir distinguir regiões que apresentam cores similares (como folhagem e grama), analisando o padrão de variação dessas cores. A técnica mais utilizada analisa conjunto de pares de pixels da imagem em tons de cinza e monta estruturas com informações características. A principal estrutura utilizada nesta técnica é a "Matriz de co-ocorrência", e a sua utilização é a identificação de padrões em uma imagem, sendo considerada crucial para pesquisas por similaridade em imagens médicas (GLATARD; MONTAGNAT; MAGNIN, 2004).

Diversas medidas que podem ser extraídas pela análise de textura estão presentes no trabalho de (HARALICK; SHANMUGAM; DINSTEIN, 1973). Algumas das métricas extraídas da análise das matrizes de co-ocorrência são relacionadas com características específicas da textura da imagem como homogeneidade, contraste e a presença de estruturas organizadas dentro da imagem. Outras métricas caracterizam a complexidade e a natureza das transições dos tons de cinza presentes na imagem, como a entropia.

Os extratores de características das imagens utilizados neste trabalho são do framework Arboretum, desenvolvido pelo Grupo de Bases de Dados e Imagens (GBDI) da Universidade de São Paulo - campus São Carlos.

3 TÉCNICA OMNI

Neste capítulo será explanada a técnica OMNI, mencionada previamente nos Capítulos 1 e 2. A técnica OMNI, sua concepção, construção e uso é retirada do trabalho de (SANTOS FILHO et al., 2001) e (TRAINA JUNIOR et al., 2007).

3.1 CONCEITOS E DEFINIÇÕES OMNI

Como apresentado anteriormente, a técnica OMNI baseia-se em em uma série de elementos chamada de OMNI-focos. Estes focos são elementos presentes na base de dados previamente escolhidos, e possuem como função servirem de marco para o cálculo das coordenadas Omni de cada elemento presente no banco.

Definição 6. *Seja um espaço métrico* $M =< S,d >$*, uma base de focos OMNI é um conjunto* $\mathscr{F} = \{f_1, f_2, ..., f_l | f_k \in S, f_k \neq f_j, l \leq N\}$ *onde cada* f_k *é um foco (ou ponto focal) de* S*,* l *é o número de focos da base de focos e* N *é o número de elementos da base de dados.*

Definição 7. *Dado um objeto* $s_i \in S$ *e a base de focos OMNI* $\mathscr{F}$*, as coordenadas OMNI* C_i *do objeto é o conjunto de distâncias de* s_i *para cada foco em* $\mathscr{F}$*:*

$$C_i = \{< f_k, d(f_k, s_i) >, \forall f_k \in \mathscr{F}\} \tag{5}$$

Para distinguir a distância $d(f_k, s_i)$ como uma coordenada, é utilizada a notação $df_k(s_i) = d(f_k, s_i)$.

Quando um novo objeto é inserido na base de dados, as suas coordenadas OMNI são calculadas e armazenadas. Em uma pesquisa por similaridade, estas coordenadas são utilizadas para podar o número de cálculos de distância, como ilustrado pela Figura 3.

O custo associado a utilização da técnica OMNI provém de duas fontes: o tempo para calcular as coordenadas OMNI (cálculo da distância entre o elemento inserido e cada um dos focos) e o espaço físico necessário para armazenar a estrutura OMNI, geralmente armazenada em disco. Como o número de focos necessários usualmente é baixo, estes custos atrelados a técnica OMNI são menores do que o ganho de desempenho nas consultas.

3.2 USO DA BASE DE FOCOS

Os problemas entre a escolha da base de focos $\mathscr{F}$ e a sua cardinalidade l são intimamente relacionados. Quanto mais focos, mais espaço e tempo para processamento é necessário. Por isso, é necessário maximizar o ganho de desempenho com a menor quantidade possível de focos.

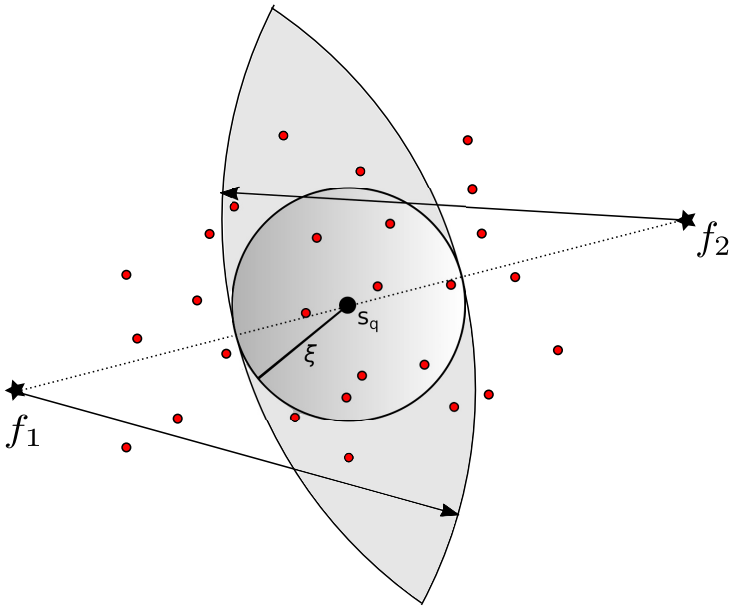
Definição 8. *Dado uma base de focos OMNI* $\mathscr{F} = \{f_1, f_2, ..., f_l\}$ *e uma coleção de objetos* $A = \{x_1,x_2,...,x_n\} \subset \mathbb{S}$*, a região OMNI de delimitação mínima (minimum bounding OMNI*

region - mbOr) de A é definido como a interseção dos intervalos métricos $R_A = |\,{}_1^l I_i$, onde $I_i = [min(d(f_i,x_j)), max(d(f_i,x_j))]$, $1 \leq i \leq l$, $1 \leq j \leq n$.

A ideia gráfica de uma *mbOr* foi previamente apresentada na Figura 3 para o caso de uma consulta utilizando um único foco. É possível ver que a *mbOr* sempre inclui todos os elementos do conjunto-resposta, mas pode incluir elementos que não são pertinentes ao conjunto-resposta (alarmes falsos). Embora seja necessária mais uma etapa de cálculo de distâncias (etapa de refinamento), o número de cálculos é reduzido drasticamente com o uso da *mbOr*.

Uma *mbOr* pode ser reduzida ainda mais com o uso de múltiplos focos, como ilustra a Figura 6.

Figura 6 – Consulta por abrangência pela técnica OMNI utilizando dois focos



Fonte: Autoria Própria

Considerando a família Minkowski de distâncias (métricas L_p), um número de focos correspondente ao valor da dimensão intrínseca da base de dados acrescido de um ($\lceil D \rceil + 1$) seria o suficiente para maximizar a performance da técnica OMNI. O uso de mais focos do que o necessário traria pouca ou nenhuma redução à mbOr. Este valor da dimensão intrínseca pode ser obtido através da técnica de box-counting (BLOCK; BLOH; SCHELLNHUBER, 1990).

3.3 ESCOLHA DOS FOCOS

Para a escolha dos focos OMNI, o elemento candidato a foco deve pertencer a base de dados. Isso se deve ao fato de que algumas vezes é impossível sintetizar um objeto de uma base de dados, como uma impressão digital. O algoritmo utilizado para o encontro dos focos é o algoritmo HF. Esse algoritmo procura aleatoriamente um elemento s_1, encontra o elemento

f_1 mais distante daquele e o seleciona como o primeiro foco. Após isso, procura pelo elemento f_2 mais longe de f_1 e o seleciona como o segundo foco, e armazenando a distância entre eles como $borda$.

O próximo foco é o elemento com as distâncias mais similares aos outros focos previamente escolhidos. Para cada objeto s_i não escolhido como foco ainda, o erro da distância em relação à borda é:

$$erro_i = \sum_{k}^{kfoco} |borda - d(f_k, s_i)| \tag{6}$$

Algoritmo 1: Algoritmo HF

Input: a base de dados $\mathbb{S}$ e o número de focos l
Output: base de focos OMNI $\mathscr{F}$
Início
1.Selecione aleatoriamente um objeto $s_i \in \mathbb{S}$.
2.Encontre o objeto f_1 mais longe de s_i. Insira f_1 em $\mathscr{F}$.
3.Encontre o objeto f_2 mais longe de f_1. Insira f_2 em $\mathscr{F}$.
4.Defina $borda = d(f_1,f_2)$, usada para calcular $erro_i$.
while *existirem focos a serem determinados* **do**
  5.Para cada $s_i \in \mathbb{S}$, $s_i \notin \mathscr{F}$: calcule $erro_i$.
  6.Selecione $s_i \in \mathbb{S}$ de tal modo que $erro_i$ seja mínimo.
  7.Insira s_i em $\mathscr{F}$.
end

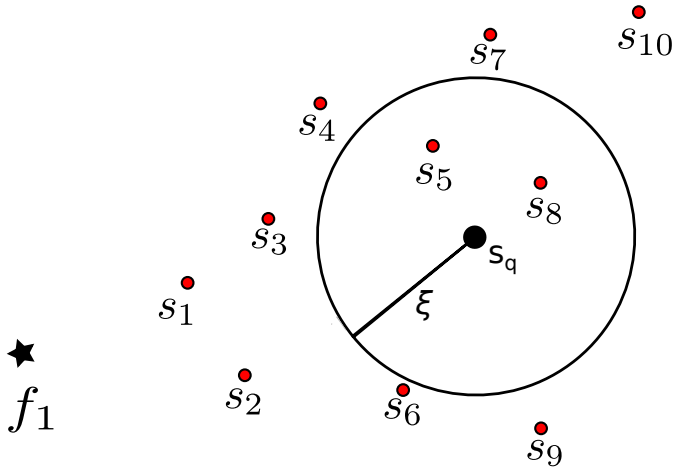
O algoritmo HF requer $l * N$ cálculos de distância. O Algoritmo 1 mostra que as etapas 3 e 5 são cálculos necessários para a determinação das coordenadas OMNI de cada objeto. Isso mostra que o algoritmo HF também prepara as coordenadas OMNI da base de dados. Outro fato importante é o da base de focos OMNI ser invariável a operações de inserção ou remoção na base.

3.4 INDEXAÇÃO DAS COORDENADAS OMNI

Objetos em um espaço métrico não são ordenáveis, portanto métodos de acesso baseados na propriedade de ordenação total como B-Trees não podem ser utilizados diretamente para a sua indexação. No entanto, as distâncias $df_k(s_i)$ de cada foco f para a base de objetos pode ser ordenada e indexada por estruturas B-Tree. Assim, é possível armazenar as coordenadas OMNI em um conjunto de h B-Trees, sendo h o número de focos da base. Cada nó da *k-ésima* B-Tree é composto pela distância $df_k(s_i)$ e o identificador interno do objeto *IOid*(s_i). Este conjunto de B-Trees é chamado de OMNIB-Forest, e fornece um suporte efetivo para bases de dados imersas em um espaço métrico, utilizando recursos nativos a SGBDRs.

Um exemplo do uso das coordenadas OMNI para a filtragem inicial do cálculo de distância dos objetos da base em relação ao elemento central da consulta no caso de uma consulta por abrangência pode ser visto abaixo.

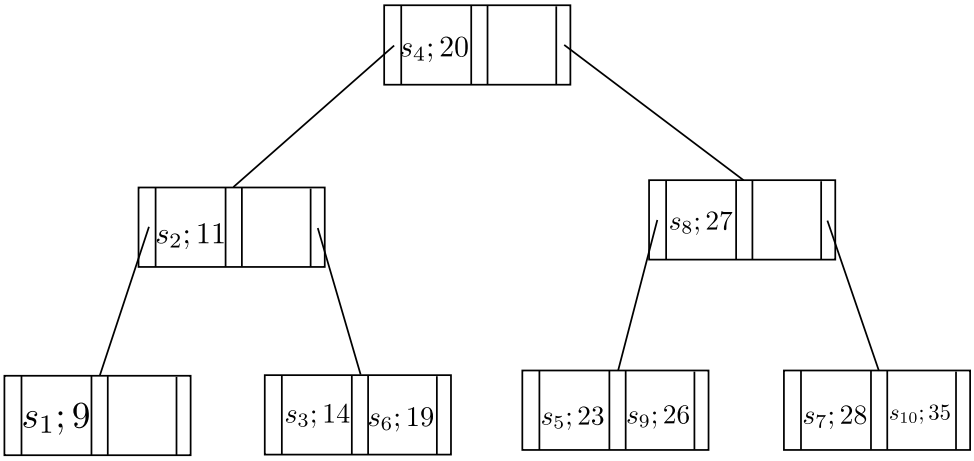
Figura 7 – Base de elementos de exemplo para uma consulta por abrangência utilizando um foco



Fonte: Autoria Própria

A Figura 8 representa a OMNIB-Tree utilizada para armazenar as coordenadas OMNI dos objetos da base em relação ao foco f_1. Nela foram armazenadas a distância do objeto s_i até o foco, e um identificador do objeto.

Figura 8 – OMNIB-Tree utilizada para o exemplo

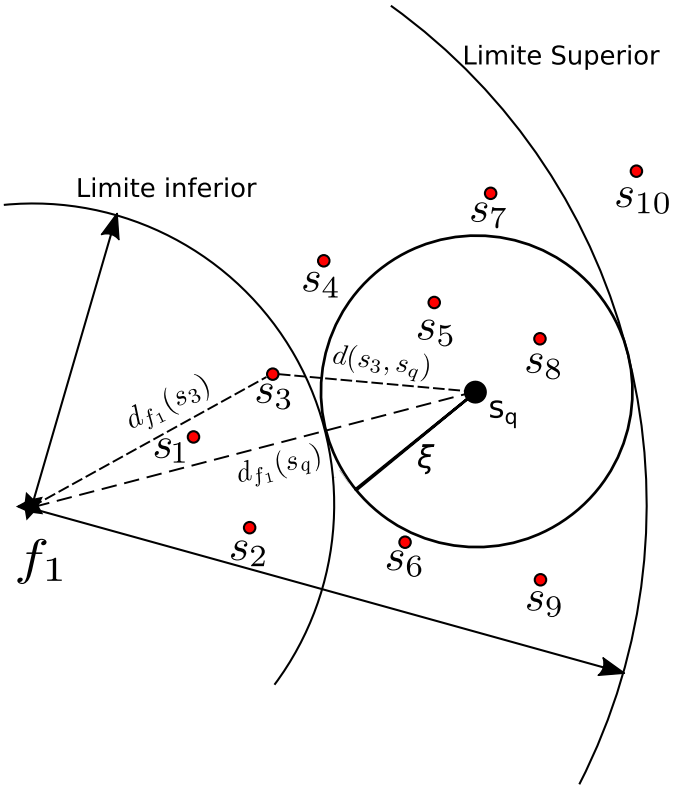


Fonte: Autoria Própria

A partir de uma análise geométrica, é possível perceber que um elemento candidato a estar presente no conjunto de resposta deve ter a sua distância em relação ao foco maior ou igual a distância do elemento central de consulta menos o valor do raio da consulta, formando, assim, um limite inferior. Analogamente, o limite superior pode ser definido utilizando a distância do elemento central da consulta até o foco mais o valor do raio da consulta.

$$|d(f_k, s_q) - \xi| \leq d(f_k, s_i) \leq |d(f_k, s_q) + \xi| \tag{7}$$

Figura 9 – Análise geométrica dos limites dos elementos candidatos



Fonte: Autoria Própria

A Figura 9 ilustra a utilização da propriedade da desigualdade triangular para a etapa de filtragem. Tomando o elemento s_3 como referência e a equação da desigualdade triangular apresentada na Definição 3, é possível definir que:

$$d_{f_1}(s_3) \leq d_{f_1}(s_q) + d(s_3, s_q) \tag{8}$$

$$d(s_3, s_q) \geq |d_{f_1}(s_3) - d_{f_1}(s_q)| \tag{9}$$

Utilizando o conceito de bola apresentado pela Definição 4, se o elemento s_3 só pode pertencer a bola de consulta se a distância $d(s_3, s_q)$ for menor do que o raio de consulta ξ. Com a Equação 9, é possível definir que o elemento está fora do conjunto de elementos candidatos se:

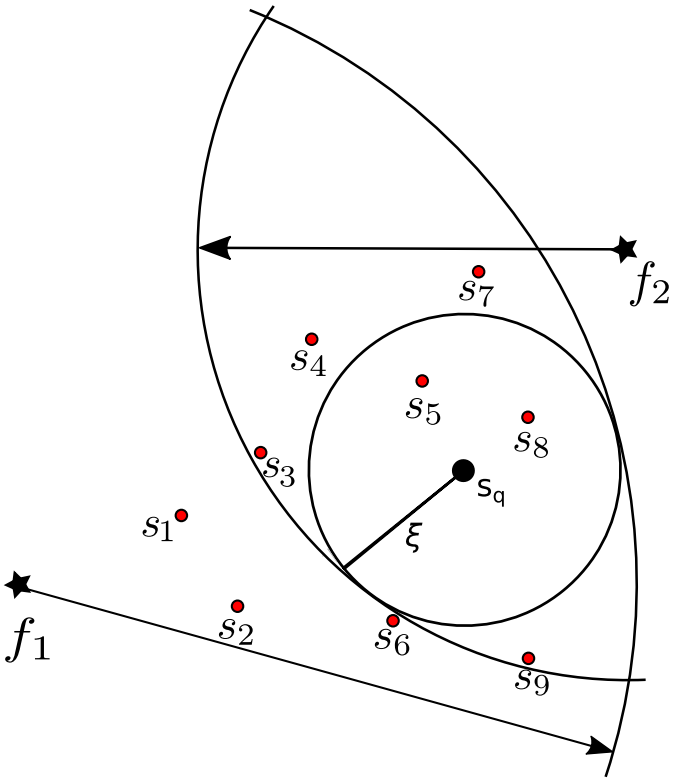
$$|d_{f_1}(s_3) - d_{f_1}(s_q)| \geq \xi \tag{10}$$

Sabendo que a distância $df_1(s_q)$ é de 23 unidades e o raio da consulta por abrangência é de 8 unidades, utilizando a Equação 7 é possível determinar que a distância $d(f_k, s_i)$ do elemento candidato precisa ser ≥ 15 e ≤ 31. Utilizando a OMNIB-Tree criada para o foco f_1, é possível encontrar os elementos candidatos que prosseguirão para a etapa de refinamento, sendo eles: $s_4, s_5, s_6, s_7, s_8, s_9$. É possível notar que estes candidatos estão em conformidade com o que foi previsto utilizando uma análise geométrica fornecida pela Figura 9. Também é

possível verificar a validade da Equação 10, pois esta fornece o mesmo conjunto de elementos encontrados com a análise geométrica.

É importante notar que nem todos os elementos selecionados durante a etapa de filtragem da técnica OMNI pertencem ao conjunto resposta. Na *mbOr*, pode ocorrer a presença de elementos que não estão dentro da bola da consulta por abrangência (alarmes falsos). Para minimizar o número de alarmes falsos, múltiplos focos podem ser utilizados, como demonstrado pela Figura 10. Para este exemplo, o elemento s_{10} é escolhido como um novo foco f_2. O elemento s_6 é descartado antes da etapa de refinamento, o que não acontece para uma filtragem utilizando um único foco.

Figura 10 – Etapa de filtragem com dois focos



Fonte: Autoria Própria

4 RESULTADOS

Este capítulo ilustra como foram feitas as etapas de implementação, detalhando alguns fatores importantes nos códigos utilizados, métodos para a descoberta ótima do número de focos para cada base, assim como uma breve explicação sobre as bases de dados utilizadas e algunas das métricas empregadas. Também apresenta os resultados obtidos, e análises pertinentes a estes resultados, estabelecendo um comparativo com as técnicas convencionais de pesquisa por similaridade. Todos os resultados foram extraídos utilizando o SGBD PostgreSQL 10.3 em conjunto com o aplicativo pgAdmin 4 versão 2.1, responsável pela interface com o banco de dados. O hardware utilizado para as pesquisas é um Intel® Core i7™ 7700HQ 2.8 GHz possuindo 16 GB de memória RAM. O sistema operacional utilizado foi o Windows 10 Home Edition.

4.1 BASES DE DADOS

Foram utilizadas duas bases de dados complexas distintas para a verificação deste trabalho: uma base de imagens de cães e gatos, e uma base de dados extraídos de exames médicos de imagem.

4.1.1 BASE CAT_DOG

Esta base de dados é dividida entre conjunto de teste e conjunto de treinamento, e encontra-se disponível em <https://www.kaggle.com/c/dogs-vs-cats>. A base originalmente foi criada para uma competição promovida pela comunidade de cientistas de dados conhecida como Kaggle, aonde o objetivo era criar o algoritmo que conseguisse classificar imagens entre fotografias ou desenhos de gatos ou cachorros com o maior índice de acerto.

Para a verificação da técnica implementada, foi utilizado o conjunto de imagens de treino, com 25000 imagens, divididas igualmente entre cães e gatos. Tornou-se necessário a remoção de 3 imagens que eram pequenas demais e impossibilitavam a segmentação destas na etapa de processamento de imagens. As imagens foram processadas para a extração das características desejadas e inseridas no banco de dados. As características escolhidas para cor foram a média e a variância do valor dos pixels em cada um dos três canais, para textura foram dissimilaridade, contraste e correlação, e para a característica de forma foram utilizadas métricas como área, excentricidade e a razão área sobre área convexa. Todas as características foram extraídas utilizando um script feito na linguagem Python versão 3.5, com o auxílio da biblioteca de algoritmos de manipulação de imagens *scikit-image*.

(a) cat.325

(b) cat.6243

(c) dog.2032

(d) dog.10641

Figura 11 – Exemplos de imagens da base CAT_DOG

4.1.2 BASE HC

A base HC foi fornecida pelo Hospital das Clínicas de Ribeirão Preto, e ela consiste em 500000 imagens com as suas características já extraídas de uma base de imagens no formato DICOM. Embora ela seja uma base não-sintética, não foram fornecidos as imagens correspondentes aos vetores de características, tornando-se assim impossível realizar uma análise qualitativa sobre a taxa de acerto nas buscas por similaridade mas ainda é possível verificar o tempo de consulta nesta base, fator este que é o interesse deste trabalho.

Os dados se encontram na forma de vetores, com a primeira posição sendo um número inteiro utilizado como um identificador da coluna seguido por um vetor de 256 números decimais, representando as características extraídas de 256 tons de cinza de cada imagem. Embora esta base não possua as imagens correspondentes aos dados extraídos, o seu tamanho e a sua complexidade são maiores do que a base apresentada em 4.1.1, tornando mais evidente os ganhos de performance nas consultas por similaridade utilizando a técnica proposta.

4.2 MÉTRICAS DOS TESTES

As métricas utilizadas nos testes foram os tempos gastos pelo pgAdmin para planejar e executar as consultas. Estas métricas foram adquiridas através dos comandos *EXPLAIN ANALYZE* seguido da consulta a ser realizada, como por exemplo abaixo.

Código 1 – Exemplo de comando utilizado para a visualização das métricas de performance

```
1 EXPLAIN ANALYZE SELECT * FROM rangeOmniHCF2(12345, 30)
```

Os testes foram feitos escolhendo um centro de pesquisa aleatoriamente, e variando os parâmetros de raio para o caso da consulta por abrangência, e os valores de k para uma consulta do tipo k-vizinhos mais próximos. Também foi utilizado o número de podas de cálculos desnecessários de distância, conseguidos pela técnica aplicada neste trabalho.

4.2.1 EXPLAIN ANALYZE

O comando *EXPLAIN ANALYZE* utilizado para a verificação dos tempos gastos pelo SGBD para a realização das consultas é nativo exclusivamente do PostgreSQL. O comando é utilizado para mostrar o plano de execução gerado para uma declaração, sem executá-la. As tabelas referenciadas, assim como algoritmos de junção ou índices utilizados são explicitados pelo *EXPLAIN* (POSTGRESQL, 2017). Em associação com o comando *ANALYZE*, a declaração também é executada e as estatísticas relacionadas ao tempo de execução atuais são mostradas para o usuário, divididas entre templo de planejamento e tempo de execução.

4.2.1.1 TEMPO DE PLANEJAMENTO

O tempo de planejamento (*planning time*) é o tempo gasto pelo SGBD para planejar a maneira mais rápida de executar a consulta. O plano escolhido é aquele que possui a execução estimada menos custosa dentre diversos outros planos testados pelo planejador. No contexto deste trabalho, o tempo de planejamento é pequeno o bastante para ser desprezado, representando um pouco mais de 1% do tempo gasto no caso da consulta mais rápida, e menos de 0,001% no caso mais lento. Os tempos de planejamento ficaram em uma média de 0.023 ms para pesquisas na base cat_dog e 0.027 ms para pesquisas na base HC.

4.2.1.2 TEMPO DE EXECUÇÃO

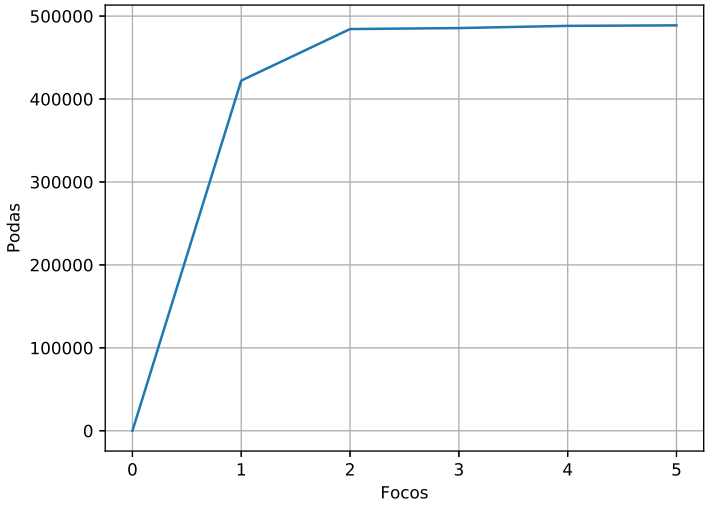
O tempo de execução (*execution time*) representa o tempo em que o sistema leva para para executar o plano de consulta escolhido, além do tempo gasto para retornar a resposta da consulta para o usuário. Para o mérito de análise do ganho de performance utilizando a técnica OMNI, esta será a métrica utilizada. Este tempo é influenciado pela complexidade da consulta, tamanho da base de dados, presença ou não de índices e hardware utilizado para a sua execução.

4.3 NÚMERO DE FOCOS

Como mencionado no Capítulo 3, a escolha do número de focos impacta diretamente na performance da técnica OMNI. Se o número for menor que o ótimo, a *mbOr* torna-se muito grande, e o número de podas de cálculos diminui, aumentando assim o tempo de execução. Se o número de focos for maior que o ótimo, existe pouca variação na *mbOr* mas o número de cálculos necessários para descobrir a pertinência na região aumenta, aumentando o tempo de execução. O cálculo da dimensão intrínseca da base de dados pela técnica de box-counting não é realizável para bases clusterizadas (MO; HUANG, 2012) e normalizar as coordenadas de uma base de dados deste tipo seria alterar as informações originais salvas nas tabelas, alterando assim o dado armazenado. Uma solução proposta por este trabalho provém da contagem do número de podas.

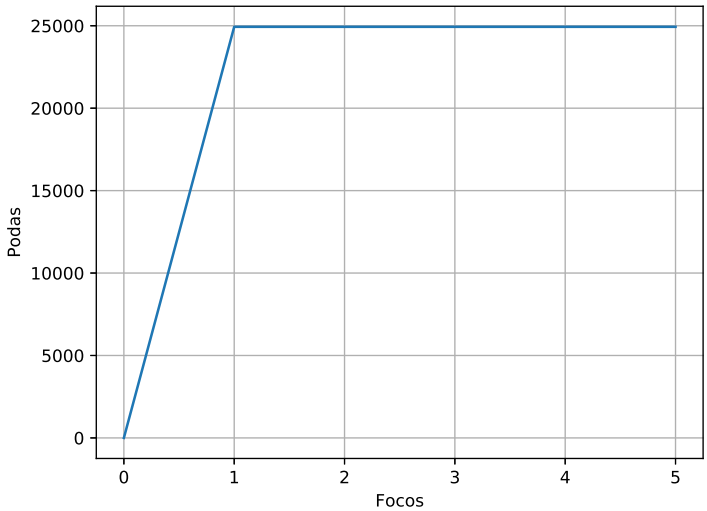
O número ótimo de focos é obtido analisando um gráfico do número de podas em relação ao número de focos utilizados. Um número ótimo de focos é aquele que produz um número suficiente de podas sem afetar o desempenho para a aplicação delas. Como ilustrado pelas Figuras 12 e 13, os números ótimos de focos para as bases HC e cat_dog são respectivamente 2 e 1.

Figura 12 – Número de podas por focos utilizados para a base HC



Fonte: Autoria Própria

Figura 13 – Número de podas por focos utilizados para a base cat_dog



Fonte: Autoria Própria

4.4 SCRIPTS SQL

Os scripts necessários para a parte de manipulação do banco de dados foram todos escritos em PLPGSQL, uma linguagem estrutural estendida da SQL, desenvolvida para o uso

no SGBD PostgreSQL. Também foi utilizada a extensão *cube* presente no PostgreSQL, para o cálculo das distâncias. Aqui serão discutidos apenas alguns scripts mais importantes para a análise de resultados. Os demais códigos encontram-se integralmente nos Apêndices A. Para fins de otimização dos scripts (tanto sequenciais ou OMNI), tornou-se necessário a criação de uma função para cada par (atributo, distância) envolvendo a base de dados cat_dog.

4.4.1 SCRIPTS SEQUENCIAIS

Os scripts sequenciais das consultas realizam a pesquisa vasculhando toda a base de dados, e retornando os elementos que fazem parte do critério de seleção, seja dentro do raio de abrangência indicado, ou dentre os k elementos mais próximos do elemento central. Abaixo, códigos utilizados para consultas sequenciais na base cat_dog.

Código 2 – Consulta por abrangência sequencial utilizando forma e distância euclidiana

```
CREATE OR REPLACE FUNCTION rangeQueryShapeL2 (center_id integer, radius FLOAT8)
RETURNS SETOF genericQuery AS $$
BEGIN
    RETURN QUERY SELECT DISTINCT * FROM
    (SELECT T2.id_ft, (cube(T1.feature) <-> cube(T2.feature)) AS dist -- <-> indica calculo de
    FROM SHAPE T1, SHAPE T2 WHERE T1.id_ft = center_id)        -- distancia L2
    AS quer WHERE dist <= radius ORDER BY dist;
END;$$
LANGUAGE PLPGSQL;
```

Código 3 – Consulta k-vizinhos mais próximos sequencial utilizando cor e distância city-block

```
CREATE OR REPLACE FUNCTION kNNColorL1 (center_id integer, k_neigh integer)
RETURNS SETOF genericQuery AS $$
BEGIN
  RETURN QUERY SELECT * FROM
  (SELECT T2.ID_FT, (cube(T1.feature) <#> cube(T2.feature)) AS dist  -- <#> indica calculo de
  FROM COLOR T1, COLOR T2 WHERE T1.id_ft = center_id       -- distancia L1
  AND T1.ID_FT <> T2.ID_FT) as quer
  ORDER BY dist LIMIT k_neigh;
END;$$
LANGUAGE PLPGSQL;
```

Como a base HC apresenta apenas um grande vetor como característica, foram necessárias apenas 3 funções, uma para cada distância Minkowski utilizada.

Código 4 – Consulta por abrangência sequencial utilizando distância Chebyshev para a base HC

```
CREATE OR REPLACE FUNCTION rq_HC_LInf (center_id integer, radius FLOAT8)
RETURNS SETOF genericQuery AS $$
BEGIN
  RETURN QUERY SELECT DISTINCT * FROM
  (SELECT T2.COD, (cube(T1.feature) <=> cube(T2.feature)) AS dist   -- <-> indica calculo de
  FROM HC_TABLE T1, HC_TABLE T2 WHERE T1.cod = center_id) as quer   -- distancia Chebyshev
  WHERE dist <= radius ORDER BY dist;
END;$$
LANGUAGE PLPGSQL;
```

4.4.2 SCRIPTS OMNI

Para um melhor uso do ganho de desempenho proposto pela técnica OMNI, observou-se a necessidade de criação de uma função para cada tripla (atributo, distância, número de focos). Isto se deve ao fato de que funções em SQL que possuem o nível de flexibilidade necessário para a passagem de parâmetros como atributo, distância e número de focos utilizados, apresentam uma performance muito inferior quando comparadas com as suas versões *hard-coded*. Além disso, a técnica OMNI exige uma preparação inicial do banco e a criação das suas estruturas, como mencionado na Seção 1.2. Os scripts necessários para essa preparação encontram-se no Apêndice A. Abaixo, códigos utilizados para as consultas OMNI.

Código 5 – rangeOmni utilizando 1 foco para a base cat_dog

```
1  CREATE OR REPLACE FUNCTION rangeOmniShapeL2F1 (center_id integer, radius FLOAT8)
2  RETURNS SETOF genericQuery AS $$
3  DECLARE rec_id RECORD; feature_aux FLOAT8[]; distance FLOAT8; dist_fc FLOAT8;
4  BEGIN
5    SELECT T1.feature INTO feature_aux FROM Shape T1 WHERE T1.id_ft = center_id;
6    SELECT dist_l2 INTO dist_fc FROM SHAPE_F_BASE where id_feature = center_id;
7    FOR rec_id IN SELECT id_feature FROM SHAPE_F_BASE
8    WHERE ((dist_l2 < (radius + dist_fc)) AND (dist_l2 > dist_fc  - radius)) LOOP
9      SELECT (cube(feature_aux) <-> cube(feature))
10     INTO distance FROM SHAPE T1
11     WHERE rec_id.id_feature = T1.id_ft  ;
12     IF  (distance  < radius) THEN
13       RETURN NEXT (rec_id.id_feature, distance);
14     END IF;
15   END LOOP;
16 END;$$
17 LANGUAGE PLPGSQL;
```

Alguns elementos importantes do Código 5 requerem uma explanação. Na linhas 3 nota-se a criação de variáveis para o armazenamento de alguns dados lidos. Isto é necessário para armazenar informações importantes para a consulta em memória, evitando assim o acesso desnecessário a disco que é mais custoso que o acesso à memória. A linha 8 é a responsável pela checagem da pertinência à *mbOr*, dada pela desigualdade triangular indicada pela Equação 7. Da linha 9 até a linha 13 nota-se a etapa de refinamento da técnica, necessária para eliminar elementos que não fazem parte do conjunto-resposta.

Códigos que utilizam mais de um foco apresentam a mesma estrutura, sendo a única diferença a interseção das múltiplas regiões, necessitando l comparações de desigualdade triangular, sendo l o número de focos da base. Essa diferença é ilustrada na linha 10 do Código 6.

Código 6 – rangeOmni utilizando 2 focos para a base HC

```
CREATE OR REPLACE FUNCTION rangeOmniHCL2F2 (center_id integer, radius FLOAT8)
RETURNS SETOF genericQuery AS $$
DECLARE rec_id RECORD; feature_aux FLOAT8[]; distance FLOAT8; dist_fc FLOAT8[];
BEGIN
      SELECT T1.feature INTO feature_aux FROM HC_TABLE T1 WHERE T1.cod = center_id;
       dist_fc  = ARRAY(SELECT DISTINCT dist_l2 FROM HC_F_BASE WHERE id_feature = center_id);
      FOR rec_id IN SELECT id_feature FROM HC_F_BASE
      WHERE ((dist_l2 < (radius + dist_fc [1]) )
      AND (dist_l2 > dist_fc [1]  − radius))
      INTERSECT (SELECT id_feature FROM HC_F_BASE
      WHERE ((dist_l2 < (radius + dist_fc [2]) )
      AND (dist_l2 > dist_fc [2]  − radius))) LOOP
        SELECT (cube(feature_aux) <−> cube(feature)) INTO distance
        FROM HC_TABLE T1 WHERE rec_id.id_feature = T1.cod ;
        IF (distance  < radius) THEN
          RETURN NEXT (rec_id.id_feature, distance);
        END IF;
      END LOOP;
END;$$
LANGUAGE PLPGSQL;
```

4.5 TEMPOS DE CONSULTAS

Os tempos de consultas foram extraídos como explicados na Seção 4.2. Os resultados foram obtidos através de uma média de 10 medições distintas, os valores dos centros de pesquisa foram escolhidos aleatoriamente antes do início dos testes, e o mesmo valor de centro foi utilizado para melhor comparação dos resultados. É importante ressaltar que diferentes características como textura, cor e forma não apresentam alterações significativas nos tempos de consulta. A Tabela 1 mostra os resultados obtidos para a base CAT_DOG, com $S_q = 42412$, $\xi = 100$ e utilizando a característica de forma.

Tabela 1 – Busca por abrangência - CAT_DOG - L1 - FORMA.

	Tempo de Execução(ms)	N° Podas de Cálculos
Sequencial	8,721	-
OMNI 1 Foco	1,747	24935
OMNI 2 Focos	5,201	24935
OMNI 3 Focos	7,606	24935
OMNI 4 Focos	10,107	24935
OMNI 5 Focos	12,271	24935

É possível notar que a melhor performance foi obtida com o número ótimo de focos ilustrado pela Figura 13. E isto também pode ser observado nas Tabelas 2 e 3, que contém os resultados do mesmo experimento anterior, mas para as distâncias L_2 e L_∞.

Tabela 2 – Busca por abrangência - CAT_DOG - L2 - FORMA.

	Tempo de Execução(ms)	N° Podas de Cálculos
Sequencial	8,728	-
OMNI 1 Foco	1,825	24935
OMNI 2 Focos	4,988	24935
OMNI 3 Focos	7,525	24935
OMNI 4 Focos	8,994	24935
OMNI 5 Focos	11,591	24935

Tabela 3 – Busca por abrangência - CAT_DOG - L_∞ - FORMA.

	Tempo de Execução(ms)	N° Podas de Cálculos
Sequencial	8,805	-
OMNI 1 Foco	2,051	24935
OMNI 2 Focos	4,638	24935
OMNI 3 Focos	8,563	24935
OMNI 4 Focos	9,651	24935
OMNI 5 Focos	11,626	24935

Outro fator a ser notado é o número de podas permaneceu constante independente do tipo de distância utilizada. Isto se deve ao fato do valor das distâncias não variarem muito com diferentes distâncias Minkowski para esta base de dados. As Tabelas 4 e 5 exemplificam o fato de que diferentes características utilizadas não alteram significativamente a performance das consultas, como dito no início da Seção 4.5. Para essa consulta, foram utilizados $S_q =$ 42413 e 42414 (característica de textura e cor referente a mesma imagem da característica de forma com identificador $=$ 42412) e $\xi = 100$.

Tabela 4 – Busca por abrangência - CAT_DOG - L1 - TEXTURA.

	Tempo de Execução(ms)	N° Podas de Cálculos
Sequencial	8,710	-
OMNI 1 Foco	1,531	24950
OMNI 2 Focos	4,512	24950
OMNI 3 Focos	6,838	24950
OMNI 4 Focos	9,447	24950
OMNI 5 Focos	11,182	24950

Tabela 5 – Busca por abrangência - CAT_DOG - L1 - COR.

	Tempo de Execução(ms)	Nº Podas de Cálculos
Sequencial	8,334	-
OMNI 1 Foco	1,675	24923
OMNI 2 Focos	4,819	24923
OMNI 3 Focos	8,073	24923
OMNI 4 Focos	9,628	24923
OMNI 5 Focos	12,925	24923

Para a base HC, também é possível verificar o ganho de performance obtido pela técnica OMNI. Como a base é significativamente maior e os dados mais complexos, o ganho de performance é mais notável, como ilustram as Tabelas 6 e 7. Estas tabelas foram construídas utilizando consultas com $S_q = 49628$ e $\xi = 20$. Novamente, a melhor performance pode ser observada com o número ótimo de focos indicado pela Figura 12.

Tabela 6 – Busca por abrangência - HC - L1.

	Tempo de Execução(ms)	Nº Podas de Cálculos
Sequencial	4566,851	-
OMNI 1 Foco	448,105	484381
OMNI 2 Focos	292,709	490599
OMNI 3 Focos	361,712	499406
OMNI 4 Focos	462,795	499845
OMNI 5 Focos	882,159	499937

Tabela 7 – Busca por abrangência - HC - L2.

	Tempo de Execução(ms)	Nº Podas de Cálculos
Sequencial	4310,409	-
OMNI 1 Foco	1215,435	422246
OMNI 2 Focos	307,757	484381
OMNI 3 Focos	556,683	485574
OMNI 4 Focos	598,664	488229
OMNI 5 Focos	952,008	488841

A técnica OMNI aplicada com a consulta por k-vizinhos mais próximos não conseguiu trazer resultados mais rápidos do que a consulta sequencial. Isto se deve ao fato de que esta consulta efetua uma busca por abrangência em torno do elemento central de pesquisa, e retorna os k valores mais próximos. Para isto, o valor do raio utilizado deve ser grande o bastante para retornar um número razoável de elementos. O Código 7 ilustra como foram feitas as consultas OMNI-*kNN*.

Código 7 – kNN-OMNI utilizando 2 focos para a base HC

```
CREATE OR REPLACE FUNCTION kNNOmniHCL2F2 (center_id INTEGER, k_value INTEGER)
RETURNS SETOF genericQuery AS $$
DECLARE radius FLOAT8 := 100;
BEGIN
  RETURN QUERY SELECT *
  FROM rangeOMNIHCL2F2(center_id,radius)
  ORDER BY distance LIMIT k_value;
END;$$
LANGUAGE PLPGSQL;
```

Outra forma implementada foi a expansão do raio em tempo de execução da consulta, caso seja pequeno demais para retornar os k elementos necessários, mas este método provou ser ainda menos eficiente que o previamente mencionado. A Tabela 8 contém os valores dos tempos de execução de uma consulta OMNI-*kNN* com $s_q = 75003$, $k = 30$. O valor de ξ utilizado para efetuar a consulta foi de 100.

Tabela 8 – Busca OMNI-*kNN* - HC - L2.

	Tempo de Execução(ms)
Sequencial	4733,579
OMNI 1 Foco	6519,995
OMNI 2 Focos	6341,987
OMNI 3 Focos	7490,256
OMNI 4 Focos	7548,550
OMNI 5 Focos	10348,541

O uso da distância L_∞ na base de dados HC provou ser inviável. A distância Minkowski com $p = \infty$ essencialmente mede a maior diferença de valores entre duas dimensões quaisquer de dois pontos. Como a base possui os valores dos seus atributos entre 0 e 256, esta distância geralmente convergia para 256, tornando boa parte da base igualmente espaçada entre si, e entre os focos. Isto dificulta a poda de elementos, tornando a técnica ineficaz nesta situação.

4.6 ANÁLISE QUALITATIVA DAS CONSULTAS

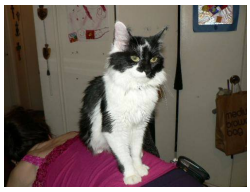
As consultas tanto sequenciais como OMNI retornam o mesmo resultado, sendo a diferença apenas a velocidade de execução da consulta. O foco deste trabalho mantém-se apenas nos tempos das consultas, mas para ilustrar como seriam o retorno dessas consultas, seguem abaixo alguns exemplos de consultas nas três características da base CAT_DOG.

(a) cat.50 - S_q

(b) cat.9211

(c) dog.10089

(d) dog.3125



(e) cat.5010

(f) cat.4258

Figura 14 – Exemplo de consulta por abrangência utilizando Forma

(a) dog.6431 - S_q

(b) cat.3864

(c) cat.6114

(d) cat.8027

(e) dog.12048

(f) dog.4639

Figura 15 – Exemplo de consulta por abrangência utilizando Textura

(a) dog.11868 - S_q

(b) cat.12219

(c) dog.9322

(d) cat.9185

(e) cat.4439

(f) dog.10028

Figura 16 – Exemplo de consulta por abrangência utilizando Cor

4.7 LIMITAÇÕES DA TÉCNICA

Embora a técnica OMNI mostrou-se eficaz para consultas por abrangência mesmo em bases pequenas e com baixo nível de complexidade, algumas limitações são inerentes à natureza da técnica. A principal limitação é em relação ao tamanho do raio ξ empregado em consultas OMNI. Quanto maior o raio, menos elementos são descartados pela etapa de poda, sendo necessário mais cálculos de pertinência à *mbOr* e mais cálculos de distância. As Tabelas 9 e 10 mostram um comparativo do tempo de execução de consultas por abrangência sequenciais e OMNI conforme aumenta-se o raio de pesquisa para as bases CAT_DOG e HC, juntamente com o número de elementos do conjunto resposta.

Tabela 9 – Performance em função do aumento do raio - CAT_DOG.

Raio	Sequencial (ms)	OMNI (ms)	Nº Elementos
100	8,598	1,938	62
500	8,580	4,582	324
1000	8,951	7,91	633
1500	8,760	11,119	973
2000	9,164	12,735	1331
2500	9,518	16,379	1654
3000	10,307	20,153	2000
3500	10,466	25,228	2303
4000	10,464	28,088	2597
4500	10,609	29,088	2928

Tabela 10 – Performance em função do aumento do raio - HC.

Raio	Sequencial (ms)	OMNI (ms)	Nº Elementos
20	4742,696	254,221	11
50	4917,345	1122,165	553
80	4626,832	1890,589	2001
110	4762,850	2683,298	4256
140	4825,782	3531,885	7992
170	4708,758	8280,403	13903
200	4927,518	8251,041	22708
230	4893,118	8393,683	36753
260	5157,097	8800,501	59719
290	5415,949	9045,798	95710

5 CONCLUSÃO

No presente capítulo será abordada a conclusão do trabalho aqui escrito. Pontos fracos e pontos fortes da técnica utilizada, assim como cenários de melhor empregabilidade da técnica. Também serão discutidos possíveis aprimoramentos deste trabalho, na forma de trabalhos ou pesquisas futuras.

5.1 PERFORMANCE DA TÉCNICA

Para as consultas por abrangência, a técnica OMNI conseguiu um ganho de performance significativo quando comparada com o método sequencial. A Tabela 11 mostra os ganhos de performance da consulta mais rápida em relação ao método sequencial. O cálculo de ganho de performance (GP) percentual se dá por: $GP_{\%} = (T_o/T_s - 1) * 100$, sendo T_o o tempo da consulta OMNI mais rápida (quantidade de focos ótima) e T_s o tempo da consulta sequencial.

Tabela 11 – Ganho de Performance - CAT_DOG.

	GP (%)
L_1 - FORMA	399,20
L_2 - FORMA	378,24
L_∞ - FORMA	329,30
L_1 - TEXTURA	468,91
L_1 - COR	397,55

O ganho de performance para a base CAT_DOG foi extremamente satisfatório, dado ao tamanho reduzido e baixa complexidade dos dados. Inicialmente estimou-se que a técnica só teria resultados significativos com bases massivas e com complexidade elevada, cenários aonde a poda de cálculos desnecessários traria um maior benefício em questões de tempo de execução.

A base HC apresentou um ganho de performance ainda maior com a técnica OMNI, devido a sua elevada complexidade de dados e quantidade de tuplas a serem analisadas em cada consulta. Isto ilustra o potencial da técnica para bases ainda maiores, ou até mesmo para a manipulação de *Big Data*. A Tabela 12 contém os ganhos de performance da técnica na base HC.

Tabela 12 – Ganho de Performance - HC.

	GP (%)
L_1	1460,20
L_2	1304,04
L_∞	-

Porém, a técnica tem as suas limitações, como visto na Seção 4.7. Enquanto consultas sequenciais apresentam pouca perda de desempenho com o aumento do raio da consulta por abrangência, as consultas OMNI são muito mais sensíveis às variações de raio. Isso se deve ao fato de raios muito elevados causam um crescimento na *mbOr*, o que acarreta em um número maior de alarmes falsos, aumentando a quantidade de cálculos de distância necessários na etapa de refinamento. Sob a luz dessas informações, é seguro dizer que a técnica OMNI só pode ser empregada viavelmente para consultas que retornem até no máximo cerca de 2% do número de tuplas presentes na base, como ilustrado pelas Tabelas 9 e 10.

Outro ponto interessante a ser notado é que a técnica apresenta uma certa tolerância em relação ao número de focos. Como visto na Seção 4.5, técnica OMNI ainda consegue um desempenho melhor que o do método sequencial mesmo com um número não ótimo de focos.

5.2 CUSTOS DA TÉCNICA

A técnica OMNI prevê que as melhorias de performance em tempo de execução são possíveis através de um custo adicional de espaço em disco, e o tempo necessários para a criação das estruturas OMNI utilizadas, como mencionado previamente na Seção 3.1. A Tabela 13 contém o espaço em disco necessário para armazenar as tabelas utilizadas pelo banco, assim como as estruturas OMNI criadas. O espaço necessário está dividido entre espaço em disco para armazenar os dados, e o espaço utilizado pelas estruturas de indexação.

Tabela 13 – Custos de armazenamento.

	Índices (MB)	Dados (MB)
SHAPE	1,62	4,86
COLOR	0,55	2,83
TEXTURE	0,55	2,83
SHAPE_F_BASE	36	1,47
HC_TABLE	53	1058
COLOR_F_BASE	12	13
TEXTURE_F_BASE	24	1,47
HC_F_BASE	263	1229

O custo de tempo de criação das estruturas OMNI variam entre 30 segundos até 36 minutos, dependendo do tamanho da base e número de focos utilizados. Mas este custo pode ser ignorado devido ao fato de que o processo de criação destas estruturas é realizado apenas uma vez, e que a base focal é invariante a operações de inserções.

5.3 TRABALHOS FUTUROS

Como visto neste trabalho, a técnica OMNI possui um grande potencial em acelerar consultas por similaridade, mas as suas limitações restringem um pouco a sua usuabilidade.

Dentre os aprimoramentos futuros para este trabalho, destacam-se estudos mais específicos sobre os limites da técnica e como expandí-los. Outra melhoria importante seria um algoritmo mais eficiente de consultas por k-vizinhos mais próximos que possa ser utilizado com a técnica OMNI.

Outro trabalho interessante a nível de um comparativo qualitativo seria o de aprimorar a segmentação e extração de características de imagens, de modo a produzir consultas com um maior índice de acerto, assim como consultas que consigam operar com mais de uma característica ao mesmo tempo, mantendo o ganho de performance proposto pela OMNI.

Referências

BARIONI, M. C. N. et al. Seamlessly integrating similarity queries in sql. **Software: Practice and Experience**, John Wiley & Sons, Ltd., v. 39, n. 4, p. 355–384, 2009. ISSN 1097-024X. Disponível em: <http://dx.doi.org/10.1002/spe.898>.

BLOCK, A.; BLOH, W. von; SCHELLNHUBER, H. J. Efficient box-counting determination of generalized fractal dimensions. **Phys. Rev. A**, American Physical Society, v. 42, p. 1869–1874, Aug 1990. Disponível em: <https://link.aps.org/doi/10.1103/PhysRevA.42.1869>.

BUGATTI, P. H. et al. Prosper: Perceptual similarity queries in medical cbir systems through user profiles. **Computers in biology and medicine**, Elsevier, v. 45, p. 8–19, 2014.

CHORAS, R. S. Image feature extraction techniques and their applications for cbir and biometrics systems. **International journal of biology and biomedical engineering**, v. 1, n. 1, p. 6–16, 2007.

DB-ENGINES. **DB-Engines Ranking**. 2017. Disponível em: <https://db-engines.com/en/ranking>. Acesso em: 30 de agosto de 2017.

FERREIRA, M. R. P. et al. Identifying algebraic properties to support optimization of unary similarity queries. v. 450, 05 2009.

GLATARD, T.; MONTAGNAT, J.; MAGNIN, I. E. Texture based medical image indexing and retrieval: Application to cardiac imaging. In: **Proceedings of the 6th ACM SIGMM International Workshop on Multimedia Information Retrieval**. New York, NY, USA: ACM, 2004. (MIR '04), p. 135–142. ISBN 1-58113-940-3. Disponível em: <http://doi.acm.org/10.1145/1026711.1026734>.

GUTTA, S.; WECHSLER, H. Face recognition using hybrid classifiers. **Pattern Recognition**, Elsevier, v. 30, n. 4, p. 539–553, 1997.

HARALICK, R. M.; SHANMUGAM, K.; DINSTEIN, I. Textural features for image classification. **IEEE Transactions on Systems, Man, and Cybernetics**, SMC-3, n. 6, p. 610–621, Nov 1973. ISSN 0018-9472.

JAIN, A. K.; DUBES, R. C. **Algorithms for Clustering Data**. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1988. ISBN 0-13-022278-X.

LEHMAN, T. J.; CAREY, M. J. A study of index structures for main memory database management systems. In: **Proceedings of the 12th International Conference on Very Large Data Bases**. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1986. (VLDB '86), p. 294–303. ISBN 0-934613-18-4. Disponível em: <http://dl.acm.org/citation.cfm?id=645913.671312>.

LEHMANN, T. M. et al. Content-based image retrieval in medical applications: a novel multistep approach. In: INTERNATIONAL SOCIETY FOR OPTICS AND PHOTONICS. **Storage and Retrieval for Media Databases 2000**. [S.l.], 1999. v. 3972, p. 312–321.

LIMA, E. **Espaços métricos**. [S.l.]: Instituto de Matemática Pura e Aplicada, CNPq, 1977. (Projeto Euclides).

MARCHIORI, A. et al. Cbir for medical images - an evaluation trial. In: **Proceedings IEEE Workshop on Content-Based Access of Image and Video Libraries (CBAIVL 2001)**. [S.l.: s.n.], 2001. p. 89–93.

MO, D.; HUANG, S. H. Fractal-based intrinsic dimension estimation and its application in dimensionality reduction. **IEEE Transactions on Knowledge and Data Engineering**, v. 24, n. 1, p. 59–71, Jan 2012. ISSN 1041-4347.

POLA, I. R. V. **Explorando conceitos da teoria de espaços métricos em consultas por similaridade sobre dados complexos**. Agosto 2010. Tese (Doutorado em Ciência da Computação) — Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, 2010.

POSTGRESQL. **PostgreSQL 10 Documentation**. 2017. Disponível em: <https://www.postgresql.org/docs/10/>. Acesso em: 25 de outubro de 2018.

SANTOS FILHO, R. F. et al. Similarity search without tears: The omni family of all-purpose access methods. In: **Proceedings of the 17th International Conference on Data Engineering**. Washington, DC, USA: IEEE Computer Society, 2001. p. 623–630. ISBN 0-7695-1001-9. Disponível em: <http://dl.acm.org/citation.cfm?id=645484.656543>.

SHIRALI, S.; VASUDEVA, H. **Metric Spaces**. Springer London, 2005. ISBN 9781846282447. Disponível em: <https://books.google.com.br/books?id=MXUbkAhMjLQC>.

SILBERSCHATZ, A.; KORTH, H. F.; SUDARSHAN, S. **Database system concepts**. 6. ed. [S.l.]: McGraw-Hill, 2011.

SWAIN, M. J.; BALLARD, D. H. Color indexing. **International Journal of Computer Vision**, v. 7, n. 1, p. 11–32, Nov 1991. ISSN 1573-1405. Disponível em: <https://doi.org/10.1007/BF00130487>.

TRAINA JUNIOR, C. et al. The omni-family of all-purpose access methods: a simple and effective way to make similarity search more efficient. v. 16, p. 483–505, 08 2007.

ZIGHED, D. A. et al. **Mining Complex Data**. 1st. ed. [S.l.]: Springer Publishing Company, Incorporated, 2008. ISBN 3540880666, 9783540880660.

Apêndices

APÊNDICE A – SCRIPTS OMNI

Neste apêndice, estão alguns dos demais scripts e códigos utilizados para a criação das estruturas OMNI, assim como os scripts SQL utilizados para a criação das tabelas utilizadas pelo SGBD. Foram descritos aqui apenas scripts referentes à base CAT_DOG. Os scripts OMNI para a base HC são virtualmente os mesmos, alterando apenas o nome de algumas tabelas para as tabelas análogas da outra base.

Código 8 – Criação das tabelas primárias do banco

```
CREATE TABLE COMPLEX_DATA (
ID_CD VARCHAR(15) PRIMARY KEY);

CREATE TABLE SHAPE (
ID_FT INTEGER PRIMARY KEY,
CD_REF VARCHAR(15),
IS_FOCUS BOOLEAN DEFAULT 'FALSE',
FEATURE FLOAT8[],
CONSTRAINT CD_FK FOREIGN KEY (CD_REF) REFERENCES COMPLEX_DATA (ID_CD)
ON DELETE CASCADE ON UPDATE CASCADE);

CREATE TABLE COLOR (
ID_FT INTEGER PRIMARY KEY,
CD_REF VARCHAR(15),
IS_FOCUS BOOLEAN DEFAULT 'FALSE',
FEATURE FLOAT8[],
CONSTRAINT CD_FK FOREIGN KEY (CD_REF) REFERENCES COMPLEX_DATA (ID_CD)
ON DELETE CASCADE ON UPDATE CASCADE);

CREATE TABLE TEXTURE (
ID_FT INTEGER PRIMARY KEY,
CD_REF VARCHAR(15),
IS_FOCUS BOOLEAN DEFAULT 'FALSE',
FEATURE FLOAT8[],
CONSTRAINT CD_FK FOREIGN KEY (CD_REF) REFERENCES COMPLEX_DATA (ID_CD)
ON DELETE CASCADE ON UPDATE CASCADE);
```

Código 9 – Criação das tabelas OMNI

```
CREATE TABLE SHAPE_F_BASE(
ID_FOCUS INTEGER,
ID_FEATURE INTEGER,
DIST_L1 FLOAT8,
DIST_L2 FLOAT8,
DIST_LINF FLOAT8,
PRIMARY KEY (ID_FOCUS, ID_FEATURE),
CONSTRAINT SHAPEFOCUS_FK FOREIGN KEY (ID_FOCUS) REFERENCES SHAPE (ID_FT)
ON DELETE CASCADE ON UPDATE CASCADE,
CONSTRAINT SHAPEFEATURE_FK FOREIGN KEY (ID_FEATURE) REFERENCES SHAPE(ID_FT)
ON DELETE CASCADE ON UPDATE CASCADE);

CREATE TABLE COLOR_F_BASE(
ID_FOCUS INTEGER,
ID_FEATURE INTEGER,
DIST_L1 FLOAT8,
```

```
DIST_L2 FLOAT8,
DIST_LINF FLOAT8,
PRIMARY KEY (ID_FOCUS, ID_FEATURE),
CONSTRAINT COLORFOCUS_FK FOREIGN KEY (ID_FOCUS) REFERENCES COLOR (ID_FT)
ON DELETE CASCADE ON UPDATE CASCADE,
CONSTRAINT COLORFEATURE_FK FOREIGN KEY (ID_FEATURE) REFERENCES COLOR(ID_FT)
ON DELETE CASCADE ON UPDATE CASCADE);

CREATE TABLE TEXTURE_F_BASE(
ID_FEATURE INTEGER,
ID_FOCUS INTEGER,
DIST_L1 FLOAT8,
DIST_L2 FLOAT8,
DIST_LINF FLOAT8,
PRIMARY KEY (ID_FOCUS, ID_FEATURE),
CONSTRAINT TEXTUREFOCUS_FK FOREIGN KEY (ID_FOCUS) REFERENCES TEXTURE (ID_FT)
ON DELETE CASCADE ON UPDATE CASCADE,
CONSTRAINT TEXTUREFEATURE_FK FOREIGN KEY (ID_FEATURE) REFERENCES TEXTURE(ID_FT)
ON DELETE CASCADE ON UPDATE CASCADE);
```

Código 10 – Inserção das distâncias na tabela de focos

```
CREATE OR REPLACE FUNCTION insertShapeDistance()
RETURNS VOID AS $$
BEGIN
  INSERT INTO shape_f_base (id_focus, id_feature, dist_l1, dist_l2, dist_linf )(
   select S1.id_ft, S2.id_ft, cube(S1.feature) <#> cube(S2.feature),
  cube(S1.feature) <-> cube(S2.feature), cube(S1.feature) <=> cube(S2.feature)
  from shape S1, shape S2
  where S1.is_focus = TRUE AND S1 <> S2) ;
END;
$$ LANGUAGE plpgsql;
```

Código 11 – Exclusão da base focal

```
CREATE OR REPLACE FUNCTION wipe_shape_focus_base ()
RETURNS VOID AS $$
BEGIN
  update shape set is_focus = false where is_focus = true;
   delete from shape_f_base;
  RAISE NOTICE 'Shape Focus Base wiped';
END;$$
LANGUAGE PLPGSQL;
```

Código 12 – Criação da base focal OMNI

```
CREATE OR REPLACE FUNCTION createShapeFocusBase(num integer)
RETURNS VOID AS $$
DECLARE fprev_id integer; fnext_id integer; distance float8;
n_inserted integer := 0; border float8;
BEGIN
  PERFORM wipe_shape_focus_base();
   select id_ft into fprev_id from shape -- SELECIONA UM DADO RANDOM
  order by random()
  LIMIT 1;
  LOOP
     select S2.id_ft, cube(S1.feature) <-> cube(S2.feature) as dist
     into fnext_id, distance from shape S1, shape S2
     where S1.id_ft = fprev_id AND S2.is_focus = 'False'
```

```
    order by dist DESC
    LIMIT 1;

    update shape
    set is_focus = 'True'
    where id_ft = fnext_id;

    n_inserted = n_inserted + 1;
    num = num - 1;
    RAISE NOTICE 'Num = %, n_inserted = %', num, n_inserted;
    EXIT WHEN num = 0;
    IF n_inserted = 2 THEN
       select cube(S1.feature) <-> cube(S2.feature) into border
       from shape S1, shape S2 --CALCULA O VALOR DA BORDA
       where S1.id_ft = fprev_id and S2.id_ft = fnext_id;
       fprev_id = fnext_id;

       LOOP
  select S2.id_ft, abs(border - (cube(S1.feature) <-> cube(S2.feature)))
  as err into fnext_id, distance from shape S1, shape S2
  where S1.id_ft = fprev_id AND S2.is_focus = 'False'
  order by err ASC -- minimiza o erro
  LIMIT 1;

  update shape
  set is_focus = 'True'
  where id_ft = fnext_id;

  fprev_id = fnext_id;
  num = num - 1;
  EXIT WHEN num <= 0;
      END LOOP;
      EXIT WHEN num = 0;
    END IF;

    fprev_id = fnext_id;
    END LOOP;
  PERFORM insertShapeDistance();
END;$$
LANGUAGE PLPGSQL;
```

Código 13 – Range Query sequencial - L2

```
CREATE OR REPLACE FUNCTION rangeQueryShapeL2 (center_id integer, radius FLOAT8) RETURNS SETOF genericQuery
      AS $$
BEGIN
  RETURN QUERY SELECT DISTINCT * FROM
  (SELECT T2.id_ft, (cube(T1.feature) <-> cube(T2.feature))
  AS dist FROM SHAPE T1, SHAPE T2
  WHERE T1.id_ft = center_id)
  AS quer WHERE dist <= radius ORDER BY dist;
END;$$
LANGUAGE PLPGSQL;
```

Código 14 – Range OMNI com 5 focos - L2

```
CREATE OR REPLACE FUNCTION rangeOmniShapeL2F5 (center_id integer, radius FLOAT8)
RETURNS SETOF genericQuery AS $$
DECLARE rec_id RECORD;feature_aux FLOAT8[];
distance FLOAT8; dist_fc FLOAT8[];
BEGIN
   select T1.feature into feature_aux FROM Shape T1 where T1.id_ft = center_id;
   dist_fc = array( select distinct dist_l2 from SHAPE_F_BASE where id_feature = center_id);
  FOR rec_id IN SELECT id_feature from SHAPE_F_BASE WHERE ((dist_l2 < (radius + dist_fc[1]))
  AND (dist_l2 > dist_fc [1] - radius)) INTERSECT
  (SELECT id_feature from SHAPE_F_BASE WHERE ((dist_l2 < (radius + dist_fc[2]))
  AND (dist_l2 > dist_fc [2] - radius))) INTERSECT
  (SELECT id_feature from SHAPE_F_BASE WHERE ((dist_l2 < (radius + dist_fc[3]))
  AND (dist_l2 > dist_fc [3] - radius))) INTERSECT
  (SELECT id_feature from SHAPE_F_BASE WHERE ((dist_l2 < (radius + dist_fc[4]))
  AND (dist_l2 > dist_fc [4] - radius))) INTERSECT
  (SELECT id_feature from SHAPE_F_BASE WHERE ((dist_l2 < (radius + dist_fc[5]))
  AND (dist_l2 > dist_fc [5] - radius))) LOOP
     select (cube(feature_aux) <-> cube(feature)) INTO distance
    FROM SHAPE T1 WHERE rec_id.id_feature = T1.id_ft ;
    IF (distance < radius) THEN
      RETURN NEXT (rec_id.id_feature, distance);
    end if ;
  END LOOP;
END;$$
LANGUAGE PLPGSQL;
```