Consultas por similaridade no modelo relacional

Gabriel Vicente De Pierro

SERVIÇO DE PÓS-GRADUAÇÃO DO ICMC-USP

Data de Depósito:

Assinatura: _____

Gabriel Vicente De Pierro

Consulta por similaridade no modelo relacional

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. *VERSÃO REVISADA*

Área de Concentração: Ciências de Computação e Matemática Computacional

Orientador: Prof. Dr. Caetano Traina Junior

USP – São Carlos
Julho de 2015

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

D419c De Pierro, Gabriel Vicente
Consultas por similaridade no modelo relacional
/ Gabriel Vicente De Pierro; orientador Caetano
Traina Jr.. -- São Carlos, 2015.
83 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, 2015.

1. BANCO DE DADOS MULTIMÍDIA. 2. BANCO DE DADOS
RELACIONAIS. I. Traina Jr., Caetano, orient. II.
Título.

Gabriel Vicente De Pierro

Similarity queries in the relational model

Master dissertation submitted to the Instituto de Ciências Matemáticas e de Computação - ICMC-USP, in partial fulfillment of the requirements for the degree of the Master Program in Computer Science and Computational Mathematics. *FINAL VERSION*

Concentration Area: Computer Science and Computational Mathematics

Advisor: Prof. Dr. Caetano Traina Junior

USP – São Carlos
July 2015

Resumo

Os Sistemas de Gerenciamento de Bases de Dados Relacionais (SGBDR) foram concebidos para o armazenamento e recuperação de grandes volumes de dados. Tradicionalmente, estes sistemas suportam números, pequenas cadeias de caracteres e datas (que podem ser comparados por identidade ou por relações de ordem – RO), porém vem se tornando necessário organizar, armazenar e recuperar dados mais complexos, como por exemplo dados multimídia (imagens, áudio e vídeo), séries temporais etc. Quando se trata de dados complexos há uma mudança de paradigma, pois as comparações entre elementos são feitas por similaridade em vez das RO utilizadas tradicionalmente, tendo como mais frequentemente utilizados os operadores de comparação por abrangência (Rq) e por k -vizinhos mais próximos (k -NN). Embora muitos estudos estejam sendo feitos nessa área, quando lidando com consultas por similaridade grande parte do esforço é direcionado para criar as estruturas de indexação e dar suporte às operações necessárias para executar apenas o aspecto da consulta que trata da similaridade, sem focar em realizar uma integração homogênea das consultas que envolvam ambos os tipos de operadores simultaneamente nos ambientes dos SGBDRs. Um dos principais problemas nessa integração é lidar com as peculiaridades do operador de busca por k -NN. Todos os operadores de comparação por identidade e por RO são comutativos e associativos entre si. No entanto o operador de busca por k -NN não atende a nenhuma dessas propriedades. Com isso, a expressão de consultas em SQL, que usualmente pode ser feita sem que a expressão da ordem entre os predicados seja importante, precisa passar a considerar a ordem. Além disso, consultas que utilizam comparações por k -NN podem gerar múltiplos empates, e a falta de uma metodologia para resolvê-los pode levar a um processo de desempate arbitrário ou insensível ao contexto da consulta, onde usuários não tem poder para intervir de maneira significativa. Em alguns casos, isso pode levar a uma mesma consulta a retornar resultados distintos em casos onde a estrutura interna dos dados estiver sujeita a modificações, como por exemplo em casos de transações concorrentes em um SGBDR. Este trabalho aborda os problemas gerados pela inserção de operadores de busca por similaridade nos SGBDR, mais especificamente o k -NN, e propõe novas maneiras de representação de consultas com múltiplos predicados, por similaridade ou RO, assim como novos operadores derivados do k -NN que são mais adequados para um ambiente relacional que permita consultas híbridas, e permitem também controle sobre o tratamento de empates.

Palavras-chave: Consultas por similaridade, SGBDR, Banco de dados multimídia, Consultas aos k -vizinhos mais próximos.

Abstract

The Relational Database Management Systems (RDBMS) were originally conceived to store and retrieve large volumes of data. Traditionally, these systems support only numbers, small strings of characters and dates (which could be compared by identity and a Order Relationship – OR). However it has been increasingly necessary to organize, store and retrieve more complex data, such as multimedia (images, audio and video), time series etc. Dealing with those data types requires a paradigm shift, as the comparisons between each element are made by similarity, and not by the traditionally used identity or OR, with the most common similarity operators used being the range (Rq) and k-Nearest Neighbors ($k-NN$). Despite many studies in the field, when dealing with similarity queries a large part of the effort has been directed towards the data structures and the necessary operations to execute only the similarity side of the query, not paying attention to a more homogenous integration of queries that involve both operator types simultaneously in RDBMS environments. One of the main problems for such integration is the peculiarities of the $k-NN$ operator. Both identity and OR operators possess the commutative and associative properties amongst themselves, but the $k-NN$ operator does not. As such, expressing SQL queries, that usually can disregard the order in which predicates appear, now needs to be aware of the ordering. Furthermore, queries that use $k-NN$ might generate multiple ties, and the lack of a methodology to solve them might lead to an arbitrary or context-detached untying process, where users have little or no control to intervene. In some applications, the lack of a controlled untying process may even lead to each query yielding distinct results if the underlying structures ought be subject to change, as it is be the case of the concurrent transactions in a relational database management system (RDBMS). This work focuses on the problems that arise from the integration of similarity based operators into RDBMS, more specifically the $k-NN$, and proposes new ways to represent queries with multiple predicates, including similarity, identity or OR, as well as new operators derived from $k-NN$ that are better suited for a RDBMS environment containing hybrid queries, and also enable control over the untying process.

Key-words: Similarity queries, RDBMS, Multimedia databases, k-nearest neighbor queries.

1	Introdução	1
1.1	Contexto e Motivação	1
1.2	Objetivos do Trabalho	3
1.3	Organização do Documento	4
2	Arquitetura de SGBDs Relacionais	5
2.1	O Modelo Relacional	6
2.2	A Álgebra Relacional e a linguagem SQL	7
2.3	Processamento de Consultas	10
2.4	Considerações Finais	15
3	Consultas por Similaridade	16
3.1	Conceitos Fundamentais	16
3.1.1	Tipos de Consulta	18
3.1.1.1	Seleções por Similaridade	18
3.1.1.2	Junções por Similaridade	22
3.2	Métodos de Acesso Métricos	24
3.2.1	Slim-Tree	27
3.3	Integração de Consultas por Similaridade em SGBDRs	29
3.3.1	Abordagens Existentes	30
3.3.2	O SIREN	32
3.4	Considerações Finais	38
4	Embutindo Consultas k-NNq em SGBDs Relacionais	40
4.1	Introdução	40
4.2	Consultas por Similaridade no Modelo Relacional	44
4.3	Embutindo o Operador de Comparação por k -NN no Modelo Relacional	46
4.4	Considerações Finais	49
5	Desempatando Consultas k-NNq em SGBDs Relacionais	51
5.1	Introdução	51
5.2	O Operador de Seleção por Similaridade com Desempate	53
5.3	Um Algoritmo para o operador de Seleção por Similaridade com Desempate	56
5.4	Uma notação para tratar desempates em SQL	61

6	Experimentos Realizados	63
6.1	Introdução	63
6.2	Contagem de Tuplas e Contagem de Valores	63
6.3	Algoritmo de Desempate	67
6.4	Considerações Finais	72
7	Conclusão	73
7.1	Considerações Finais	73
7.2	Principais Contribuições	74
7.3	Sugestões de Trabalhos Futuros	74

LISTA DE FIGURAS

2.1	A Relação Funcionário	7
2.2	Visão do processamento de consultas em um SGBDR	12
2.3	Árvore canônica.	13
2.4	Árvore de comandos reestruturada utilizando propriedades algébricas.	14
3.1	Exemplo 2D das diversas distâncias da família Minkowski.	17
3.2	Consulta por Rq às capitais europeias a até uma dada distância de Estocolmo. . .	19
3.3	Consulta por k -NN às seis capitais europeias mais próximas da cidade de Estocolmo.	21
3.4	Junções por similaridade	23
3.5	Representação lógica de uma Slim-Tree	28
3.6	Representação de uma Slim-Tree num espaço bidimensional Euclidiano	28
3.7	Arquitetura inicial do SIREN (retirada do trabalho de Barioni et al. [13])	33
3.8	Arquitetura do SIREN com o otimizador de consultas	37
3.9	Otimizador de consultas do SIREN	38
4.1	Representação visual das duas interpretações propostas para o operador k -NN . .	43
5.1	Representação gráfica de três consultas	62
6.1	Medidas obtidas para as duas interpretações do operador k -NN sobre o conjunto de dados Pinturas	66
6.2	Medidas obtidas para as duas interpretações do operador k -NN sobre o conjunto de dados NSF	68
6.3	Medidas para as consultas Q13 , Q14 e Q15 sobre o conjunto de dados PlantasLenhosas	71

LISTA DE TABELAS

2.1	Os operadores tradicionais da álgebra relacional	8
3.1	Operadores por similaridade	24
4.1	As duas interpretações do operador k - NN	47
5.1	Símbolos usados nas expressões de predicados	57

LISTA DE ALGORITMOS

1	k - NN por Contagem de Tuplas	49
2	k - NN por Contagem de Valores	49
3	Algoritmo de desempate de tuplas.	58
4	Preenche os valores do atributos.	59

GBdI-ICMC-USP	Grupo de Bases de Dados e Imagens - Instituto de Ciências Matemáticas e de Computação - Universidade de São Paulo.
SGBD	Sistemas de Gerenciamento de Bases de Dados
SGBDR	Sistemas de Gerenciamento de Bases de Dados Relacionais
RO	Relação de Ordem
Rq	Consulta por abrangência (<i>Range Query</i>)
k -NN	k -vizinhos mais próximos (<i>k-Nearest Neighbors</i>)
k -NN q	Consulta aos k -vizinhos mais próximos (<i>k-Nearest Neighbors Query</i>)
OR	<i>Order Relationship</i>
RDBMS	<i>Relational Database Management Systems</i>
DBMS	<i>Database Management Systems</i>
CBIR	<i>Content-Based Image Retrieval</i>
TBIR	<i>Tag-Based Image Retrieval</i>
SQL	<i>Structured Query Language</i>
DML	<i>Data Manipulation Language</i>
DDL	<i>Data Definition Language</i>
DCL	<i>Data Control Language</i>
MAM	<i>Metric Access Method</i> ou Método de Acesso Métrico
GH-Tree	<i>Generalized Hyperplane Tree</i>
VP-Tree	<i>Vantage Point Tree</i>
MVP-Tree	<i>Multi-Vantage Point Tree</i>
SA-Tree	<i>Spatial Approximation Tree</i>
BU-Tree	<i>Bottom-up index Tree</i>
DBM-Tree	<i>Density Based Metric Tree</i>
MST	<i>Minimal Spanning Tree</i>
MSA	<i>Multi-Similarity Algebra</i>
OQL	<i>Object Query Language</i>
NSF	<i>National Science Foundation</i>
SIREN	<i>SImilarity Retrieval ENgine</i>

LISTA DE SÍMBOLOS

$\mathbb{T}, \mathbb{T}_1, \mathbb{T}_2$	Relações nas quais as consultas são realizadas
$\mathbb{S}$	Domínio de atributos complexos.
$\mathbb{E}$	Domínio de atributos escalares.
$\mathbb{A}$	Domínio de atributos complexos ou escalares.
$\mathbb{S}$	Atributos complexos $S \in \mathbb{S}$.
$\mathbb{E}$	Atributos escalares $E \in \mathbb{E}$.
$\mathbb{A}$	Atributos complexos ou escalares $A \in \mathbb{A}$.
$S = Dom^*(S)$	Domínio ativo de atributos complexos $S \in \mathbb{S}$.
$E = Dom^*(E)$	Domínio ativo de atributos escalares $E \in \mathbb{E}$.
$A = Dom^*(A)$	Domínio ativo de atributos complexos ou escalares $A \in \mathbb{A}$.
s_i	Atributo complexo.
e_i	Atributo escalar.
a_i	Atributo complexo ou escalar.
t_i, t_j, t	Tuplas.
$t[\mathbb{T}]$	Tupla na relação $\mathbb{T}$.
$t[\mathbb{S}]$	Valor do atributo $\mathbb{S}$ na tuplas t .
$\mathbb{C}$	Subconjunto de atributos de uma tupla.
$\mathbb{T}_{R_t}$	Conjunto de tuplas $\in \mathbb{T}$ que resultaram em empate na distância máxima de uma k -NNq.
$ \mathbb{T}_{R_t} $	Cardinalidade do conjunto $\mathbb{T}_{R_t}$.
k'	Número de elementos $k' < \mathbb{T}_{R_t} $ que precisam ser selecionados de $\mathbb{T}_{R_t}$ para desempatar a consulta k -NNq.
$\mathbb{R}^+$	Números reais positivos.
$\langle \mathbb{S}, d \rangle$	Espaço métrico, onde $\mathbb{S}$ é o conjunto de todos os elementos que atendem aos requisitos do espaço e $d : \mathbb{S} \times \mathbb{S} \rightarrow \mathbb{R}^+$ é uma função de distância.
$d(s_1, s_2)$	Distância entre dois elementos s_1 e s_2 .
$\mathcal{T}$	Lista de expressões de comparação.
L_p	Funções de distância da Família Minkowski.
$L_0 L_\infty$	Função de distância de Chebychev.
L_1	Função de distância de Manhattan ou <i>city block</i> .
L_2	Função de distância Euclidiana.
σ	Operador de seleção.

π	Operador de projeção.
$\cup$	Operador de conjuntos: união.
$\cap$	Operador de conjuntos: intersecção.
$-$	Operador de conjuntos: diferença.
$\times$	Operador de produto cartesiano.
$\bowtie$	Operador de junção natural.
ρ	Operador de renomeação.
$\div$	Operador de divisão.
k	Número de elementos retornados em uma consulta.
d	Função de distância ou dissimilaridade.
s_q	Elemento central de consulta.
θ	Operador de comparação válido no domínio dos atributos comparados.
ξ	Limite de similaridade, limiar ou <i>threshold</i> .
γ	Operador de agregação.
ζ	Limite de parada dependente do predicado utilizado.
$\dot{\sigma}$	Operador de seleção por k-vizinhos mais próximos (contagem de tuplas).
$\ddot{\sigma}$	Operador de seleção por k-vizinhos mais próximos (contagem de valores).
$\ddot{\sigma}_{(S \ k\text{-}NN(d,\xi) \ s_q)} \top$	Operador de seleção por k-vizinhos mais próximos (genérico - qualquer uma das interpretações).
$\dot{\sigma}_{(S \ k\text{-}NN(d,\xi) \ s_q)} \top$	Operador de seleção por k-vizinhos mais próximos (contagem de valores), onde $S \ k\text{-}NN(d,\xi) \ s_q$ é o predicado de comparação por similaridade, e $\top$ uma relação.
$\sigma_{(S \ Rng(d,\xi) \ s_q)} \top$	Operador de seleção por k-vizinhos mais próximos (contagem de tuplas), onde $S \ k\text{-}NN(d,\xi) \ s_q$ é o predicado de comparação por similaridade, e $\top$ uma relação.
$\ddot{\sigma}_{(S \ k\text{-}NN(d,k) \ s_q)\{\langle \mathcal{T} \rangle\}} \top$	Operador de seleção por abrangência, onde $S \ Rng(d,\xi) \ s_q$ é o predicado de comparação por similaridade, e $\top$ uma relação.
	Operador de seleção por similaridade com desempate, onde $S \ k\text{-}NN(d,k) \ s_q$ é o predicado de comparação por similaridade $\langle \mathcal{T} \rangle$ é uma lista de termos e $\top$ uma relação.

1.1 Contexto e Motivação

Os Sistemas de Gerenciamento de Bases de Dados Relacionais (SGBDR) foram concebidos para o armazenamento e recuperação de grandes volumes de dados, garantindo que as consultas sempre possam ser respondidas de maneira exata e eficiente. A grande maioria dos Sistemas de Gerenciamento de Bases de Dados (SGBD) atuais baseia-se na tecnologia relacional, que começou a ser desenvolvida no final da década de 1960 [28]. Tradicionalmente, estes sistemas sempre suportaram apenas números, datas e pequenas cadeias de caracteres, que são referidos como dados escalares. Com a evolução das aplicações que os utilizam, vem se tornando necessário organizar, armazenar e recuperar dados mais complexos, como por exemplo informações georreferenciadas, dados vetoriais, dados multimídia (imagens, áudio e vídeo), séries temporais e sequências de proteínas.

Comparações baseadas em igualdade (ou seja, indicadas por $=$ e $\neq$) têm pouca utilidade em consultas sobre dados complexos, pois é raro existirem elementos exatamente iguais. Os operadores de comparação chamados relacionais, baseados na precedência de um dos elementos de qualquer par sobre o outro, a chamada Relação de Ordem (RO), e usualmente expressos pelos símbolos $<$, $\leq$, $>$ e $\geq$, também não são aplicáveis a estes tipos de dados, pois os domínios onde eles são representados usualmente não atendem às propriedades de uma RO. Esses operadores, baseados na comparação por igualdade ou por RO, são comumente tratados como “os *big six*” dos SGBDR [66], mas não são de muita aplicabilidade quando saindo deste paradigma relacional e se tratando de dados complexos.

Além disso, na grande maioria das vezes, a busca em coleções de dados complexos não é feita se comparando diretamente os elementos originais, mas sim usando atributos (características) que descrevem/representam o elemento complexo. Existem duas técnicas para associar atributos aos elementos complexos. A primeira, chamada de “recuperação por contexto”, como por exem-

plô em “Recuperação de Imagens por Contexto” (TBIR na sigla em inglês – *Tag-Based Image Retrieval*), consiste na associação de atributos ao dado, em geral de tipo textual, cujos valores são obtidos externamente ao elemento complexo [91]. Exemplos dessa técnica são a associação de descrições e palavras-chave a imagens feitas por pessoas, tais como laudos em imagens de exames médicos, e a associação automática de textos próximos às imagens em páginas da *web*. A segunda técnica é chamada de “recuperação por conteúdo”, como por exemplo em “Recuperação de Imagens por Conteúdo” (CBIR na sigla em inglês – *Content-Based Image Retrieval*) [90, 31, 68]. Ela corresponde à associação de atributos obtidos a partir do conteúdo do próprio elemento complexo, extraídos automaticamente por algoritmos. Os atributos extraídos são chamados “vetores de características”, e tanto podem formar vetores no sentido matemático do termo, quando as “características” são todas de um mesmo tipo numérico e medidas na mesma unidade – como é o caso dos histogramas de cor ou textura de imagens – quanto podem ser sequências mais gerais, envolvendo unidades diferentes para diferentes características ou com quantidade diferente de características entre elementos distintos do mesmo domínio – como é o caso das poligonais que descrevem formas nas imagens [62, 82, 10].

Apesar dos novos paradigmas providos por esse aumento de complexidade dos dados, as consultas que envolvem os dados tradicionais continuam sendo de grande importância, pois os dados complexos apenas adicionam novas formas de expressar predicados sobre o que pode ser consultado, mas não substituem os demais. Assim, surge a necessidade de se poder realizar consultas que utilizem os dois tipos de dados, pois não só há indícios que se pode obter melhores resultados nas consultas através da “contextualização” da informação [50, 41, 73, 54] como, simplesmente, pode ser necessário utilizar dados tradicionais e dados complexos. Por exemplo, um médico pode querer imagens semelhantes a uma determinada imagem de um tumor no pulmão, mas restringindo os resultados para apenas casos confirmados de tumores malignos, a fim de comparar a imagem inicial às recuperadas que ele sabe que possuem um tumor, buscando elucidar dúvidas ou confirmar suspeitas. Nesse caso, a consulta usa os dois tipos de dados, o atributo complexo que contém a imagem e o atributo tradicional obtido do laudo que indica se o tumor é maligno ou não. Algumas ferramentas de busca que permitem utilizar imagens como centro de consulta já vêm adotando processos semelhantes, e procuram inicialmente “tags” para associar à imagem fornecida, mesmo que o usuário não inclua nenhuma, visando facilitar, melhorar e agilizar os resultados por meio dessa combinação dos dois tipos de dados. Tal processo é utilizado, por exemplo, pela Google [80]. Consequentemente, é natural que se deseje integrar as capacidades de processar consultas sobre dados tradicionais dos SGBDR com as tecnologias criadas para tratar dados complexos. Não apenas isso, mas é também proveitoso realizar essa integração utilizando o suporte já oferecido pelos SGBDR, pois com 40 anos de desenvolvimento, eles estão entre os softwares mais complexos e completos existentes atualmente, sendo ferramentas poderosas para manipulação eficiente de grandes volumes de dados, garantindo persistência e segurança [42].

Neste contexto, diversos trabalhos vêm sendo realizados no GBdI-ICMC-USP para tratar

essa integração [39, 40, 38, 84, 13, 12]. Esses trabalhos têm procurado estender a álgebra relacional e a linguagem SQL para lidar também operadores de consultas por similaridade, além do desenvolvimento de todo o ferramental que processa esses novos operadores junto com os já existentes e tradicionais do modelo relacional. O presente trabalho pretende contribuir dando sequência ao estudo do uso de operadores de comparação por similaridade em SGBDR. Mais especificamente, este trabalho atenta para o fato de que o operador de busca aos k -vizinhos mais próximos (k -NN), um dos mais utilizados em domínios complexos de dados, possui várias particularidades quando inserido no ambiente relacional, exigindo um tratamento diferenciado por parte do SGBDR. Por exemplo, ele não possui as mesmas propriedades que operadores tradicionais e outros operadores por similaridade como a comutatividade [39], o que obriga a se tomar maiores cuidados tanto na expressão de consultas quanto em sua execução.

1.2 Objetivos do Trabalho

O principal objetivo deste trabalho é estudar a integração de operadores de busca por similaridade ao modelo relacional de maneira homogênea. Neste sentido, há uma prioridade para se tratar o operador k -NN, pois ele não obedece às mesmas propriedades que os operadores de busca por igualdade e RO. Assim, ao se resolver os problemas que ele apresenta, também se resolvem naturalmente aqueles associados ao outro operador de busca por similaridade mais popular, i.e. o de consulta por abrangência *Rng*, o qual se comporta de maneira mais semelhante aos operadores tradicionais.

A partir desse objetivo principal, dois pontos são destacados:

1. Tratar as ambiguidades que o operador de busca por k -NN gera quando inserido no ambiente relacional, e
2. Prover ao usuário um ferramental prático para lidar com empates provenientes do k -NN no ambiente relacional.

O primeiro ponto se advém da associação do operador de busca por k -NN a uma ferramenta relacional que aceita atributos complexos como valores de tuplas. Nesse caso, o operador de busca por k -NN leva a duas interpretações distintas. Isso ocorre pois, conceitualmente, o domínio do atributo complexo forma um espaço métrico onde não existe repetição de elementos, mas na prática podem existir diversas tuplas na base com o mesmo valor no atributo complexo. Portanto, um operador de comparação cuja resposta depende do domínio ativo do atributo¹ pode obter respostas distintas para relações de entrada distintas, mesmo que os valores das tuplas não mudem. Consequentemente, uma consulta por k -NN pode retornar $n \geq k$ tuplas mesmo quando apenas k valores distintos de atributos são recuperados. Essa duplicidade leva à dúvida de qual é o resultado

¹Atributo ativo é o subconjunto de valores do domínio do atributo que ocorre em pelo menos uma das tuplas na relação armazenada na base de dados.

esperado da consulta, e se faz necessário que ela seja mais bem definida para que fique claro se devem ser retornadas k tuplas ou k valores distintos de atributos complexos.

O segundo ponto surge não de um problema inerente da inclusão do k - NN em um ambiente relacional, mas de uma oportunidade gerada por essa integração que provê novos recursos que possibilitam a resolução de problemas antigos envolvendo o operador k - NN . De fato, uma operação de busca por k - NN pode resultar em empates e fora do modelo relacional não existem maneiras genéricas de se lidar adequadamente com eles. Em diversos casos a resolução é feita de forma arbitrária ou sem um critério propriamente definido. No entanto, quando inserido no modelo relacional, é possível se beneficiar dos outros atributos da tabela de forma a prover critérios de desempate controlados e contextualizados. Em particular, o interesse maior está nos casos onde o empate ocorre no k -ésimo elemento mais próximo, pois ele irá ou afetar a cardinalidade da consulta – quando todos os elementos empatados são retornados – ou não permitirá que a consulta seja realizada novamente com o mesmo resultado – quando apenas um número $k' < k$ de elementos, suficiente para completar os k elementos requisitados, são retornados de forma arbitrária.

1.3 Organização do Documento

O restante desta dissertação está organizado da seguinte maneira:

- **Capítulo 2 - Arquitetura de SGBDs Relacionais:** Apresenta os principais conceitos teóricos do modelo relacional pertinentes ao contexto deste trabalho;
- **Capítulo 3 - Consultas por Similaridade:** Apresenta os principais conceitos teóricos relacionados a consultas por similaridade, fazendo uma revisão dos aspectos fundamentais e dos principais métodos de acesso métricos. O capítulo discute também os principais trabalhos correlatos na área de integração de operadores de consulta por similaridade com operadores tradicionais do modelo relacional;
- **Capítulo 4 - Embutindo Consultas k - NNq em SGBDs Relacionais:** Apresenta uma nova maneira para se interpretar o operador de seleção por k - NN quando embutido em um ambiente relacional, buscando uma integração homogênea entre ambos;
- **Capítulo 5 - Desempatando Consultas k - NNq em SGBDs Relacionais:** Apresenta uma nova maneira de se resolver empates resultantes de consultas k - NNq dentro de ambientes relacionais.
- **Capítulo 6 - Experimentos Realizados:** Apresenta os experimentos realizados e seus resultados;
- **Capítulo 7 - Conclusão:** Finaliza a dissertação, apresentando as principais contribuições e sugestões de trabalhos futuros.

CAPÍTULO 2

ARQUITETURA DE SGBDS RELACIONAIS

Os SGBDR têm se disseminado em praticamente todas as atividades da sociedade moderna. Qualquer empresa, pública ou privada, que precise manipular dados, seja cadastro de funcionários, folhas de pagamento, informações sobre clientes ou qualquer dado composto de valores numéricos, pequenas cadeias de caracteres ou datas, e necessita que isso seja armazenado de forma confiável, segura e com acesso rápido, invariavelmente utilizará um SGBDR.

A partir do trabalho de Codd [28], os SGBDRs vem evoluindo continuamente e persistindo por mais de 40 anos como uma ferramenta essencial para diversas áreas da indústria e da academia. Com o passar do tempo eles vêm se tornando cada vez mais complexos, visando atender às diversas necessidades que surgem, tais como poder processar consultas em ambientes distribuídos, lidar com sistemas móveis cada vez menores, mais espalhados e com fluxo contínuo e crescente de informação (como redes de sensores), executar processamento analítico de dados e tratar dados em maior quantidade e complexidade de maneira geral.

Apesar do longo caminho percorrido pelos SGBDRs, existem diversos desafios emergentes que ainda precisam ser tratados. Este capítulo fornece uma visão geral dos principais conceitos e ferramentas conceituais utilizadas pelos SGBDRs atuais, procurando consolidar uma base teórica para permitir o desenvolvimento de novas tecnologias para integração de SGBDRs e operadores de consulta por similaridade. A Seção 2.1 faz uma breve revisão do modelo relacional. A Seção 2.2 provê uma visão geral da álgebra relacional e da linguagem SQL (do inglês: *Structured Query Language*). A Seção 2.3 faz uma breve revisão da arquitetura dos SGBDRs e do processamento de consultas. Finalmente, a Seção 2.4 conclui o capítulo. Os conceitos abordados neste capítulo são fundamentados principalmente nos trabalhos de Garcia-Molina, Ullman e Widom [42], Elmasri e Navathe [34], Date [30] e Date [29].

2.1 O Modelo Relacional

A ideia geral do modelo relacional é bastante simples: como diz o nome, ele representa os dados na forma de relações, que podem ser imaginadas como que tabelas (relação é o termo matemático para tabela, e os dois podem ser tomados como aproximadamente sinônimos neste contexto [30]) onde as informações são armazenadas, e o conjunto das relações de uma aplicação forma a base de dados relacional. Para os usuários do sistema, a forma como os dados estão armazenados fisicamente pouco importa: basta saber quais são as tabelas e a linguagem para expressar as consultas. O alto nível de abstração provido pelas relações, a flexibilidade da linguagem SQL e a sólida fundamentação matemática, foram elementos que fizeram o modelo relacional efetivamente se tornar o padrão da indústria. Seus antecessores, os modelos hierárquico e de rede, não proviam abstração aos dados e ocasionavam muita dependência entre a codificação das aplicações e a maneira como os dados eram armazenados [34].

O modelo relacional pode ser considerado como sendo constituído por duas estruturas sintáticas: **valores**, que são os dados que estão representados na base (números, cadeias de caracteres, datas, etc); e **relações**, que são as tabelas onde os dados são armazenados, e representam entidades e associações mapeadas do mundo real. Cada relação é composta por uma coleção de tuplas, que são as linhas da tabela, uma coleção de atributos, que são as colunas da tabela, e um esquema, que é a composição do nome da relação com o conjunto de atributos que a constitui.

A Figura 2.1 ilustra uma relação **Funcionario** com suas tuplas, atributos e valores. Neste caso específico, os atributos são **nome**, **cargo**, **dpto**, **dataNasc** e **salario** e a relação possui 4 tuplas. Cada tupla possui um valor para cada atributo da relação. Por exemplo, a primeira tupla possui os 5 valores {‘Carlos José’, ‘Estagiário’, ‘TI’, ‘22/06/1988’, ‘1500,00’} para os atributos **nome**, **cargo**, **dpto**, **dataNasc** e **salario** respectivamente. A Figura 2.1 mostra também o esquema da relação, com o nome da relação e o conjunto de atributos representado como:

Funcionario = {nome, cargo, dpto, dataNasc, salario};

Vale notar que, conceitualmente, as tuplas da relação formam conjuntos, e não listas. Assim, as tuplas não possuem ordem associada e não devem existir duas tuplas iguais. No entanto, quando há uma referência a linhas e colunas, e não a tuplas e atributos, há uma ideia intrínseca de ordem associada, pois no meio físico esses dados estarão armazenados em alguma ordem. A linguagem SQL e os SGBDs usam o termo “*ROW*” para se referenciar às tuplas. Assim, convencionou-se que quando se diz tabela, linha e coluna, refere a implementação e quando se usam os termos relação, tupla e atributo se refere ao conceito do modelo teórico. Além disso, no modelo relacional considera-se que os valores são sempre atômicos e monovalorados e que associado com cada atributo de uma relação há um domínio de dados que rege quais são os valores possíveis que as tuplas podem assumir para o atributo correspondente. Por exemplo, as tuplas da relação

Figura 2.1: A Relação Funcionário.

Funcionario são números, mas eles não podem assumir valores negativos na coluna salario.

Dada a natureza flexível e o alto nível de abstração do modelo relacional, as relações são bastante maleáveis. É possível alterar tuplas e atributos, adicionando ou removendo-os, ou alterar um componente específico de uma tupla específica, sempre operando em um alto nível, sem que o usuário necessite de conhecimentos específicos de onde os dados estão fisicamente armazená-los ou como acessá-los no meio físico.

2.2 A Álgebra Relacional e a linguagem SQL

A álgebra relacional pode ser considerada a linguagem teórica de consulta às relações do modelo relacional. Ela é a fundamentação formal para representar as operações de busca, fornecendo a base para a implementação e para a otimização da execução das consultas no modelo. A álgebra relacional consiste de diversos operadores algébricos que recebem pelo menos uma relação como entrada e geram uma nova relação como saída. Uma consulta à base pode ser formulada utilizando um ou mais operadores, gerando uma expressão algébrica relacional [34]. É possível agrupar os diversos operadores tradicionais (propostos inicialmente) na álgebra relacional em basicamente 4 classes [42]:

- As operações unárias de “filtragem”, que removem partes de uma relação: “**seleção**” – σ , que filtra tuplas; e “**projeção**” – π , que filtra atributos;
- As operações tradicionais sobre conjuntos, mas aplicadas à relações: “**união**” – $\cup$; “**inter-**

Nome	Representação
Seleção	σ
Projeção	π
União	$\cup$
Intersecção	$\cap$
Diferença	$-$
Produto Cartesiano	$\times$
Junção	$\bowtie$
Renomear	ρ

Tabela 2.1: Os operadores tradicionais da álgebra relacional

“selecção” – $\cap$; e “diferença” – $-$;

- As operações binárias, que combinam as tuplas de duas relações: “**produto cartesiano**” – $\times$, que gera uma nova relação contendo todas as combinações possíveis de tuplas das duas relações operadas, pareando as tuplas uma a uma; e as diversas operações de “**junção**” – $\bowtie$, que são semelhantes ao produto cartesiano, mas utilizam algum critério para o pareamento das tuplas;
- A operação de “**renomear**” – ρ , que permite alterar o nome de um determinado atributo e/ou relação.

A Tabela 2.1 mostra os operadores tradicionais da álgebra e a simbologia adotada.

Por exemplo, utilizando a relação da Figura 2.1, uma consulta simples que pergunta: “Q1 - Quais são os *funcionários* que têm como seu *cargo* o valor *Estagiário* e pertencem ao departamento (*Depto*) de *TI* ?” é expressa como:

$$\pi_{\{(nome, cargo)\}}(\sigma_{((cargo = 'Estagiario'))}(\sigma_{((depto = 'TI'))}Funcionario))$$

Vale ressaltar também que, assim como os operadores da álgebra elementar, os operadores da álgebra relacional possuem diversas propriedades que podem ser utilizadas para manipular as expressões algébricas. A seleção, por exemplo, possui diversas propriedades, como comutatividade, onde a ordem das seleções pode ser invertida sem afetar o resultado:

$$\sigma_{((cargo = 'Estagiario'))}(\sigma_{((departamento = 'TI'))}Funcionario)) = \sigma_{((departamento = 'TI'))}(\sigma_{((cargo = 'Estagiario'))}Funcionario))$$

ou idempotência, onde a mesma seleção que foi realizada anteriormente não altera o resultado:

$$\sigma(((\text{cargo} = \text{'Estagiario'}))\text{Funcionario}) = \sigma((\text{cargo} = \text{'Estagiario'}))(\sigma((\text{cargo} = \text{'Estagiario'}))\text{Funcionario}))$$

Estas não são as únicas propriedades da seleção. De fato, para cada operador da álgebra existem diversas propriedades, mas o ponto principal dessas propriedades, no contexto dos SGBDRs e deste projeto, é que diversas expressões algébricas diferentes podem ter o mesmo resultado final, ou seja, elas são equivalentes. Mas, apesar de terem o mesmo resultado, não necessariamente elas têm o mesmo custo de execução no SGBDR. Sendo assim, é possível encontrar expressões algébricas que resultem no mesmo conjunto de tuplas, mas que podem ser executadas em tempos completamente diferentes. Cabe ao SGBDR encontrar a expressão de menor custo antes de executar a consulta. Uma visão mais detalhada sobre o assunto é fornecida na Seção 2.3.

Com a evolução do modelo relacional, novos operadores foram adicionados visando sanar necessidades diversas. Alguns exemplos que podem ser citados são a semi-junção ($\bowtie$), a semi-diferença ($\triangleright$), e a divisão ($\div$), entre outros que foram sendo adicionados ao longo do tempo.

Em cima dessa arcabouço algébrico foi desenvolvida a linguagem SQL. Ela é uma interface de alto nível que aplica os operadores ao modelo relacional, efetivamente realizando as consultas. Além disso, a SQL ainda adiciona várias capacidades que de fato não fazem parte da álgebra, como as operações de inserção, remoção e atualização de relações.

As formas mais simples de consulta são aquelas que pedem pelas tuplas de uma relação que satisfaçam a uma condição simples. Tal consulta é análoga a uma seleção em álgebra relacional. Essa consulta simples, assim como a maioria das consultas em SQL, utiliza uma estrutura com 3 palavras-chave: **SELECT**, **FROM** e **WHERE**, que caracterizam a linguagem.

Por exemplo, a mesma consulta realizada em álgebra antes “*Q1 - Quais são os funcionários que têm como seu cargo o valor Estagiário e pertencem ao departamento (Depto) de TI ?*”, em SQL pode ser expressa como:

```
SELECT nome,cargo
FROM Funcionário
WHERE cargo = 'Estagiario'
AND departamento = 'TI';
```

Nesta consulta-exemplo é possível notar as funcionalidades de cada palavra-chave e seu paralelo com a álgebra. A cláusula **SELECT** corresponde ao operador de projeção: ela irá filtrar os atributos que aparecerão na resposta final. É possível usar o carácter especial ***** para obter todas as colunas no resultado e assim não usar um operador de projeção. A cláusula **FROM** indica sobre quais relações a consulta é feita, no caso do exemplo a única relação consultada é “**Funcionário**”. Por fim, a cláusula **WHERE** indica os predicados para a definição de um subconjunto: a condição imposta na cláusula determinará quais tuplas aparecerão no resultado final da consulta e portanto

corresponde a um operador de seleção. No caso específico do exemplo, apenas as tuplas em que o cargo do funcionário é “Estagiário” serão recuperadas da base. Esse é um dos tipos mais simples de consulta, utilizando apenas a seleção e a projeção, mas ainda era possível simplificar mais pois a cláusula **WHERE** é opcional. Em situações que mais relações são envolvidas, é possível utilizar operadores binários ou operadores conjuntos para combinar as tuplas de relações duas a duas da maneira que o usuário quiser.

Durante os mais de 40 anos de existência do modelo relacional, tanto a álgebra quanto a linguagem SQL foram evoluindo e crescendo. A álgebra adotou novos operadores para tratar os mais variados tipos de consulta e necessidades, enquanto a linguagem SQL expandiu não apenas para tratar consultas, mas também para tratar uma diversa gama de necessidades provenientes da evolução dos SGBDRs. De fato, a SQL é dividida em três subconjuntos de linguagens onde cada uma trata de um aspecto do gerenciamento de uma base de dados. Nesta seção foi apresentada uma breve noção da *Linguagem de Manipulação de Dados* ou DML (*Data Manipulation Language - DML*), que trata da manipulação do conteúdo de relações, possuindo os comandos **INSERT** (para inserção), **SELECT** (para realização de consultas), **UPDATE** (para atualização) e **DELETE** (para remoção). Existe ainda os subconjuntos da *Linguagem de Definição de Dados* ou DDL (*Data Definition Language - DDL*) que possui comandos de criação e alteração da estrutura de tabelas e índices e a Linguagem de Controle de Dados ou DCL (*Data Control Language - DCL*) que trata das parametrizações gerais e permissões que cada usuário tem para manipular os dados da base.

As possibilidades de expressar consultas da álgebra e a linguagem SQL são vastas e no contexto deste trabalho não é exequível cobrir todo seu conteúdo. No entanto, este trabalho visa justamente contribuir para a evolução do modelo relacional, procurando estender a álgebra e a linguagem SQL tradicional para atender a novas necessidades, como já feito outras vezes no passado, agora tendo como objetivo acrescentar ao modelo e à linguagem recursos para expressar e executar consultas por similaridade.

2.3 Processamento de Consultas

Em um SGBDR, o processador de consultas é um grupo de componentes que recebe as consultas do usuário em SQL e as transforma em uma série de operações que serão executadas na base de dados. Como a linguagem SQL permite que as consultas sejam expressas com um alto nível de abstração, o processador de consultas precisa prover um alto nível de detalhamento sobre como cada consulta será executada. Caso contrário, a consulta pode resultar em um algoritmo de execução não eficiente, que toma mais tempo do que seria necessário [42].

Como é possível ver na Figura 2.2, pode-se dividir o processamento de consultas em quatro etapas [42]:

- *Compilação*, quando o código SQL é compilado e uma árvore de comandos representando a

consulta é gerada;

- *Reescrita da consulta*, quando a árvore de comandos é modificada – o SGBDR utiliza as propriedades algébricas para reescrever a consulta e escolhe dentre os planos equivalentes um que tenha uma expectativa de custo menor, que é chamado o plano lógico ótimo;
- *Geração do plano físico de acesso*, quando o plano lógico da etapa de reescrita é transformado em um plano físico, selecionando os algoritmos que implementarão cada uma das operações do plano lógico, e selecionando a ordem de execução dos operadores. O plano físico, assim como o plano lógico, é representado por uma árvore de comandos, agora chamada plano de execução ou de acesso físico, que também inclui detalhes de como as relações consultadas serão acessadas e, se necessário, ordenadas.
- *Execução*, quando o plano físico finalmente é executado para se obter o resultado da consulta.

Depois de selecionado o plano de acesso físico, o SGBDR irá executá-lo, acessando o banco e retornando o resultado para o usuário. O trabalho do executor é gerenciar as operações de acesso, alocando memória conforme o necessário e escalonando as operações. As partes mais complexas são a geração dos planos de acesso físico e lógico, que compõem o processo de otimização de consultas. A otimização de consultas é uma etapa tão importante que pode ser o diferencial entre sucesso e fracasso de um SGBDR [49].

O compilador e o executor, apesar de presentes no processo, não são tratados neste capítulo. O compilador é construído como qualquer outro compilador de uma linguagem de programação, realizando as análises léxica, sintática e semântica, processando a SQL e gerando a árvore de comandos que será utilizada pelo sistema. O executor, por sua vez, irá pôr em prática o plano de acesso físico gerado, gerenciando *buffers* de memória e diversas variáveis que precisam ser levadas em consideração ao acessar os dados, o que deve ser feito pelos respectivos algoritmos que executam cada operação de acesso. Sendo assim, o foco aqui é em elucidar como ocorre a geração dos planos de acesso (físico), e de consulta (lógico). Essas duas etapas não ocorrem necessariamente separadas no SGBDR, e a otimização pode ir e voltar entre a geração dos dois planos, mas para fins de tornar mais clara a explicação, nesta monografia elas serão apresentadas separadamente.

Primeiramente, a geração do plano de consulta gera uma expressão algébrica idealmente ótima que representa a consulta. Para isso, o analisador recebe uma árvore de comandos de entrada (vinda do compilador), chamada **árvore canônica**, que é gerada a partir do comando em SQL, começando a análise a partir da cláusula **FROM** descendo até o fim da consulta e finalizando voltando para a cláusula **SELECT**. Essa análise gera uma árvore com os comandos na ordem de execução. A partir da **árvore canônica**, é possível utilizar as propriedades dos operadores da álgebra relacional, para procurar por uma **árvore ótima**, ou seja, uma árvore que gere uma expressão algébrica com, idealmente, o menor custo de execução possível. Nem sempre o plano encontrado é o melhor possível, pois em muitos casos existem simplesmente muitas alternativas

Figura 2.2: Visão, em alto nível, do processamento de consultas em um SGBDR.

Figura 2.3: Árvore canônica.

de plano, e buscar entre todas elas é computacionalmente inviável ou então a previsão pode não ser totalmente acertada. Portanto, o ideal é utilizar uma heurística que tente achar um plano “muito bom” rapidamente, mesmo que ele não seja o melhor possível. Embora possa não ser o ótimo teórico, utiliza-se o termo “otimizador” para esse processo.

Para ilustrar o processo executado pelo gerador do plano de consulta, considere-se que existe a relação **Funcionário** já apresentada na Seção 2.1 e também uma relação **Departamento** = {sigla, gerente, ativo, despesas}. Sobre essa base de dados, a consulta em SQL que lista o nome e o salário dos estagiários de cada departamento que possuem despesas maiores ou iguais a 100.000,00 pode ser representada em SQL como:

```

SELECT nome,salario
FROM Funcionário, Departamento
WHERE Departamento.sigla = Funcionario.departamento
      AND Departamento.despesas ≥ 100.000,00
      AND cargo = 'Estagiario';

```

Essa consulta gerará a árvore canônica representada na figura 2.3.

A partir da árvore canônica, o gerador do plano de consulta manipulará a expressão algébrica usando as propriedades de cada operador, procurando uma árvore de baixo custo. Para determinar qual operador deve ser executado primeiro para responder a consulta, são utilizadas as estimativas de **seletividade** de cada condição. A seletividade de uma condição é determinada pela equação 2.1:

$$Sel(C) = 1 - \frac{\text{Numero de tuplas no resultado}}{\text{Numero de tuplas na entrada}} \quad (2.1)$$

Ou seja, quando uma condição “filtra” um grande número de tuplas, sua seletividade é alta, e

Figura 2.4: Árvore de comandos reestruturada utilizando propriedades algébricas.

quanto mais alta a seletividade, melhor é que o operador esteja no começo da expressão algébrica, pois ele gerará tabelas menores para os operadores que o restante da consulta deve processar. Dessa maneira, naturalmente tabelas menores serão processadas mais rapidamente, pois envolvem menos acessos a disco para leitura e escrita de informações. A consulta da Figura 2.3, por exemplo, poderia ser reestruturada como mostra a Figura 2.4.

A reestruturação da árvore pode potencialmente levar a uma execução mais rápida. Nesse exemplo, as duas seleções foram adiantadas para antes da junção e ambas possuem uma certa seletividade, ou seja, ambas filtrarão a tabela eliminando diversas tuplas, processo que teria que ser feito de qualquer maneira depois da junção em uma tabela grande, mas que sendo feito antes permite que a junção, que é uma operação custosa, seja feita sobre duas tabelas menores. Além disso, também é de grande utilidade que as tabelas caibam em memória principal para executar as junções, evitando a necessidade de se acessar o disco diversas vezes para recuperar partes da tabela [42]. Portanto quanto menor forem as tabelas, maior será a probabilidade de que ela caiba inteira na memória, reduzindo substancialmente o número de acessos a disco.

Para consultas complexas envolvendo diversos operadores, é evidente que podem ser geradas muitas consultas equivalentes, não só pelo grande número de operadores possíveis, mas também pelo grande número de propriedades algébricas de cada operador. Por exemplo, para n junções, o espaço de busca é $n!$. Consequentemente, até mesmo consultas com um baixo número de operadores podem gerar centenas de alternativas. Portanto, é necessário que haja uma **heurística** que restrinja o **espaço de busca**, ou seja, determina quais serão os planos que têm maior possibilidade de estarem de fato entre os melhores e descarta o resto, e um **modelo de custo**, que proverá as informações necessárias para se poder avaliar cada plano de acesso, possuindo maneiras de calcular os valores aproximados das mais diversas operações possíveis [48].

Por fim, o gerador do plano de acesso recebe os planos lógicos criados pelo gerador de planos de consulta e determina qual operador de acesso físico disponível é o melhor para executar cada operação, já considerando como buscar de fato os dados no disco. Para isso serão avaliadas as

diferentes possibilidades de leitura dos dados, tais como acesso sequencial, acesso indexado usando um ou mais índices, recuperação dos dados diretamente nos índices, e as várias opções possíveis de junção (*nested loops*, *merge join*, *hash join*, e *hybrid join*). Portanto, essa etapa irá delinear qual será a maneira pela qual os dados serão acessados, indicando se e quando serão usados índices e como e em que ordem as diferentes junções serão executadas. Além disso, de maneira semelhante ao gerador de planos de acesso, cada uma dessas operações possui um custo associado, cabendo ao gerador de planos de acesso procurar a opção mais rápida em um curto espaço de tempo.

2.4 Considerações Finais

Neste capítulo foram brevemente revisados conceitos gerais da arquitetura de SGBDRs pertinentes a este trabalho. Primeiramente foi feita uma breve revisão sobre o modelo relacional. Em seguida foram abordadas a álgebra relacional e a linguagem SQL, além da relação entre as duas. Por fim, foi feita uma análise do processamento de consulta em SGBDRs, utilizando os conceitos previamente examinados do modelo relacional, da álgebra e da linguagem SQL.

CAPÍTULO 3

CONSULTAS POR SIMILARIDADE

Como as comparações por igualdade e por RO são de pouca utilidade nos domínios onde os dados complexos são representados, é necessário desenvolver novas maneiras de avaliar quão similar os elementos nesses domínios são entre si. Para isso são utilizadas as consultas por similaridade que utilizam, como o nome indica, uma medida de similaridade para avaliar quão parecido um elemento é de outros elementos no domínio. Não somente isso, mas com a complexidade dos elementos cada vez maior, o conceito de similaridade entre elementos representados em meios digitais tem se tornado a maneira por excelência de comparação [95].

Este capítulo apresenta uma visão geral dos conceitos de consultas por similaridade. Primeiramente, a Seção 3.1 fornece os conceitos fundamentais de consultas por similaridades sobre dados complexos e aborda os diferentes tipos de consulta. A Seção 3.2 apresenta os conceitos e os alguns dos principais métodos de acesso métrico (MAMs), estruturas utilizadas para indexação de dados complexos. Em seguida, a Seção 3.3, apresenta uma breve revisão dos trabalhos da literatura que procuram combinar consultas por similaridade sobre dados complexos e consultas de comparação por identidade ou Relação de Ordem (RO) sobre dados escalares, efetivamente buscando integrar as capacidades dos SGBDRs atuais com a possibilidade de se realizar consultas por similaridade sobre dados complexos. Finalmente, a Seção 3.4 conclui o capítulo.

3.1 Conceitos Fundamentais

No contexto de dados complexos, geralmente não é possível comparar diretamente os elementos, pois a comparação em geral se dá avaliando determinados aspectos dos elementos (é necessária alguma compreensão da semântica da comparação) e não a representação usada para armazená-los. Por exemplo, no caso de imagens e vídeos, a comparação direta seria feita quadro a quadro e pixel a pixel entre dois elementos. Tal comparação muito possivelmente não traria nenhum

Figura 3.1: Exemplo 2D das diversas distâncias da família Minkowski.

resultado significativo [36]. Uma alternativa mais apropriada quando se trata de dados complexos se dá em duas etapas: primeiramente é necessário extrair suas características e gerar um “**vetor de características**” que capture numericamente algum significado de interesse, permitindo uma comparação que forneça um valor indicativo da similaridade entre elementos segundo o aspecto extraído [35]. Por exemplo, em imagens podem ser extraídas informações sobre a textura, cor ou forma, de acordo com o que o usuário tenha interesse em avaliar.

Após a extração, é necessário definir uma maneira de se comparar os conjuntos de características extraídos de elementos distintos. Para isso é definida uma “**função de distância**” que efetivamente indica quão dissimilares são cada par de elementos do conjunto. Um valor zero ou próximo a zero indica que os elementos são iguais ou muito parecidos, e valores maiores indicam que os elementos são cada vez mais dissimilares. Na literatura é possível encontrar diversas funções de distância e, em geral, a eficácia de cada uma delas em discriminar as diferenças entre elementos depende do domínio de aplicação e da qualidade dos descritores [4, 63], ficando a critério do usuário ou, preferencialmente, de um especialista, qual delas deve ser usada. As funções mais utilizadas são aquelas da Família Minkowski, representadas pela Equação 3.1 a seguir, onde X e Y são dois vetores da forma $X = \{x_1, x_2, \dots, x_n\}$ e $Y = \{y_1, y_2, \dots, y_n\}$.

$$L_p(X, Y) = \sqrt[p]{\sum_{k=1}^n |x_k - y_k|^p} \quad (3.1)$$

Os valores de p podem variar de acordo com o interesse do usuário ou da lógica da aplicação e das características extraídas. Alguns dos valores mais utilizados são: $p = \infty$, que gera a função de distância L_0 ou L_∞ , chamada distância de Chebychev; $p = 1$, que gera a função de distância L_1 , chamada distância de Manhattan ou *city block*; $p = 2$, que gera a função de distância L_2 , chamada distância Euclidiana. A Figura 3.1 mostra como o valor de p afeta a região onde os elementos que estão à mesma distância r do centro estão posicionados. Observe também que a distância L_1 é a mais restritiva e L_∞ é a mais abrangente.

3.1.1 Tipos de Consulta

Operadores de busca por similaridade somente podem ser aplicados sobre um atributo S que toma valores de um domínio $\mathbb{S}$ de dados complexos, isto é, $\mathbb{S} = Dom(S)$. Dizemos que um domínio é complexo quando existe uma métrica $d : \mathbb{S} \times \mathbb{S} \rightarrow \mathbb{R}^+$ definida para $\mathbb{S}$, e da mesma maneira dizemos que atributos amostrados em $\mathbb{S}$ são atributos complexos. O subconjunto do domínio que correspondem aos valores efetivamente representados na relação é chamado o “domínio ativo” $S \in \mathbb{S}$ do atributo S naquela relação, representado como $S = D^*om(S)$. Na literatura encontram-se operadores de busca por similaridade correspondentes aos operadores algébricos de seleção e de junção.

3.1.1.1 Seleções por Similaridade

Uma consulta por similaridade é expressa com referência a um elemento s_q com $s_q \in \mathbb{S}$, que é chamado “*elemento central da consulta*” [47]. Aqui usamos uma notação um pouco diferente daquela encontrada na literatura em geral, adaptando-a para tratar consultas por similaridade sobre as relações de uma base de dados. Assim, expressamos um operador de busca por similaridade em álgebra relacional como $\boxed{\sigma_{(S \theta s_q)} T}$, onde T é a relação onde a busca é executada, S é um atributo complexo de T , θ é um operador de comparação válido no domínio de S e s_q é o elemento com que o valor do atributo S em cada tupla de T é comparado por θ para decidir se aquela tupla deve ou não ser selecionada.

A literatura em geral considera cada operação de busca por similaridade como se ele estivesse isolado dos demais, ou seja, considera-se apenas o conjunto de valores complexos, e retorna-se esses valores como resposta da operação – não se leva em conta que o conjunto de valores é de fato o domínio ativo de um atributo numa relação que potencialmente pode ter outros atributos, inclusive outros atributos complexos. A literatura trata dois tipos de consulta por similaridade fundamentais: as consultas por abrangência (“*similarity range queries*”) e as consultas aos k -vizinhos mais próximos (“*k-nearest neighbor queries*”) [55, 25].

A Consulta de Similaridade por Abrangência (*Similarity Range Query - Rq*): ela retorna os elementos do conjunto que estão a no máximo uma distância ξ dada na consulta do elemento central da consulta. Assumindo que Consulta de Similaridade por Abrangência é executada sobre um atributo de uma relação, então ela corresponde à existência de um **Critério de Comparação de Similaridade por Abrangência** $\theta = Rng(d, \xi)$, tal que aplicado a um par de elementos $s_i, s_j \in \mathbb{S}$, o predicado $\boxed{\theta = s_i Rng(d, \xi) s_j}$ retorna **verdade** sempre que $d(s_i, s_j) < \xi$.

A seleção sobre o atributo S da relação T usando um critério de similaridade por abrangência é então expressa como $\boxed{\sigma_{(S Rng(d, \xi) s_q)} T}$. Essa consulta recebe como parâmetros o elemento central da consulta s_q , uma função de dissimilaridade d e um grau de dissimilaridade ξ_q , e recupera todas as tuplas da relação T em que o valor do atributo S difere de s_q por no máximo a dissimilaridade

Figura 3.2: Consulta de similaridade por abrangência às capitais de cidades da Europa com $\xi_q = 5$ e centro da consulta s_q na cidade de Bruxelas.

ξ calculada pela função de distância d .

A Figura 3.2 ilustra a seguinte consulta de similaridade por abrangência sobre uma relação que armazena dados sobre as cidades do mundo, incluindo suas coordenadas geográficas: “Q2 - Quais são as capitais europeias distantes em no máximo 5 unidades de distância da cidade de Bruxelas ?” Em um mapa da Europa em que as diversas capitais de cada país estão representadas, a consulta utiliza a distância euclidiana $d = L_2$ sobre as coordenadas geográficas de cada localidade (latitude e longitude, que juntas formam um dado complexo) e $\xi_q = 5$, com o centro de consulta s_q sendo a cidade de Bruxelas, capital da Bélgica. Essa consulta retorna as cidades de Londres, capital da Inglaterra, Amsterdã, capital da Holanda, Paris, capital da França, Luxemburgo, capital de Luxemburgo e, por fim, Bruxelas que é o próprio centro de consulta s_q e neste caso faz parte do conjunto de capitais com distância zero do centro.

A Consulta aos k -vizinhos Mais Próximos (k -Nearest Neighbor Query - k -NNq): ela retorna os k elementos do conjunto que são os mais próximos do elemento central da consulta. Assumindo que Consulta aos k -vizinhos mais próximos é executada sobre um atributo de uma relação, então ela corresponde à existência de um **Critério de Comparação aos**

k -vizinhos mais Próximos $\theta = k\text{-}NN(d, k)$, tal que aplicado a um par de elementos $s_i, s_j \in \mathbb{S}$, o predicado $\boxed{\theta = s_i \text{ } k\text{-}NN(d, k) \text{ } s_j}$ retorna **verdade** sempre que s_i é um dos k elementos mais próximos a s_j no conjunto de dados.

A seleção sobre o atributo **S** da relação **T** usando um critério de k -vizinhos mais próximos é então expressa como $\boxed{\sigma_{(S \text{ } k\text{-}NN(d, k) \text{ } s_q)} T}$. essa consulta recebe como parâmetros o elemento central da consulta s_q , uma função de dissimilaridade d e uma quantidade k , e recupera as k tuplas da relação **T** que tem como valor do atributo **S** um dos k mais similares a s_q que estão no domínio ativo S do atributo **S** segundo a função de distância d .

A Figura 3.3 ilustra a seguinte consulta aos k -vizinhos mais próximos sobre a relação de cidades: “Q3 - *Quais são as 6 capitais europeias mais próximas da cidade de Estocolmo ?*” Em um mapa da Europa com as diversas capitais de cada país representadas, a consulta utiliza a distância $d = L_2$ sobre as coordenadas geográficas de cada localidade e $k = 6$, com o centro de consulta s_q sendo a cidade de Estocolmo, capital da Suécia. Essa consulta retornará as cidades de Oslo, capital da Noruega, Copenhague, capital da Dinamarca, Helsinque, capital da Finlândia, Talin, capital da Estônia, Riga, capital da Letônia e, por fim, Estocolmo que é o próprio centro de consulta s_q e neste caso faz parte do conjunto de capitais com distância zero do centro.

Existem diversas variações desses dois operadores básicos. Os mais comuns para as consultas por abrangências são:

consulta pontual (*point query*), que retorna o próprio centro de consulta se ele estiver na base, ou nulo caso contrário; e

consulta por abrangência reversa, que retorna os elementos mais distantes do que o grau de dissimilaridade ξ indicado.

As variações mais comuns para as consultas por $k\text{-}NNq$ são:

consulta ao vizinho mais próximo, que retorna o elemento mais similar ao centro de consulta dado;

consulta aos k -vizinhos mais distantes, que retorna os k elementos mais dissimilares ao elemento de consulta; e

consulta aos k -vizinhos mais próximos reversos ($k\text{-}RNNq$) [79], que retorna todos os elementos que tem o elemento central de consulta como um de seus k vizinhos mais próximos.

As consultas $k\text{-}NNq$ e suas variantes podem gerar empates na resposta, e estes devem ser tratados. Na literatura considera-se o caso em que é possível existir mais de um elemento à mesma distância do elemento mais distante retornado na consulta, de maneira que pode ocorrer que algum(ns) dentre eles tenha(m) que ser selecionado(s) dentre os que estão à mesma distância. Nesse caso, é necessário optar por retornar todos os elementos empatados (aumentando a quantidade k de elementos retornados), ou selecionar apenas algum(s) para completar a quantidade k de elementos indicada, correndo o risco de que duas consultas sucessivas possam retornar diferentes

Figura 3.3: Consulta aos k -vizinhos mais próximos sobre capitais de cidades da Europa com $k = 6$ e centro da consulta s_q na cidade de Estocolmo.

elementos [6, 5].

Consultas por Similaridade Agregada: Uma extensão das operações de seleção por similaridade básicas é permitir que as consultas envolvam mais de um elemento de referência [71]. Essa extensão pode ocorrer tanto para Rq quanto para $k-NNq$ e suas respectivas variações. Tal extensão permite que em vez de um centro de consulta, um conjunto de elementos com cardinalidade maior ou igual a um pode ser indicado para servir de centro da consulta. O operador de seleção estendido escolhe a resposta usando uma função de agregação γ das distâncias de cada elemento da base de dados a todos os elementos do conjunto de centros de consulta, que por isso é chamado de operador de **consulta por similaridade agregada**. Diversas funções γ podem ser usadas como função de agregação, como por exemplo minimizar a soma das distâncias [72, 94]. Qualquer operador de seleção por similaridade pode ser estendido para tratar a similaridade agregada. Essa variação é especialmente importante para ser usada como o operador de um SGBDR que dá suporte às consultas por similaridade. Isso ocorre porque, no caso geral, o centro da consulta pode ser o resultado de uma subconsulta, a qual potencialmente pode retornar mais de um elemento [12].

3.1.1.2 Junções por Similaridade

Os operadores de **junção por similaridade** comparam valores de dois atributos complexos S_j e S_i , ambos amostrados em um mesmo domínio $dom(S_j) = dom(S_i) = \mathbb{S}$, sendo que cada atributo vem de uma relação, isto é $S_j \in T_1, S_i \in T_2$. O operador de junção por similaridade, representado como $\boxed{T_1 \overset{(S_j \text{ pred}(d, \zeta) S_i)}{\bowtie} T_2}$, avalia o par ordenado $\langle s_j \in S_j, s_i \in S_i \rangle$ composto pelos elementos do produto cartesiano de $T_1 \times T_2$ selecionados pelo predicado $pred(d, \zeta)$, onde $pred$ define o tipo de junção a ser efetuada, d é a função de similaridade definida sobre $\mathbb{S}$, e ζ é um limite de parada que depende de $pred$.

A junção por similaridade mais comum é a chamada **Junção por Abrangência** (*Distance Range Join*) [77], onde o predicado é baseado no critério de comparação de similaridade por abrangência $Rng(d, \xi)$, e o limite de parada ζ é a dissimilaridade máxima ξ_q . Assim, a junção por abrangência $\boxed{T_1 \overset{(S_j \text{ Rng}(d, \xi) S_i)}{\bowtie} T_2}$, retorna o par ordenado de tuplas $\{\langle t_i, t_j \rangle, t_i \in T_1, t_j \in T_2\}$, tal que a distância $d(t_i[S_1], t_j[S_2])$ não exceda o limite ξ_q .

Também existem junções por similaridade onde o limite de parada ζ é uma quantidade máxima de pares retornados k , sendo as mais comuns as junções por vizinhança (*k-Nearest Neighbor Join*) e as junções por proximidade (*k-Closest Pair Join*) [92]. Ambas têm seu predicado baseado em critérios de comparação aos k vizinhos mais próximos, portanto o limite de parada ζ é um número de elementos k e as respostas dependem do domínio ativo S_1 do atributo S_1 na relação T_1 .

O operador de **junção por vizinhança** $\boxed{T_1 \overset{(S_j \text{ k-NN}(d, k) S_i)}{\bowtie} T_2}$ retorna os pares ordenados de

Figura 3.4: (a) Junção por abrangência com dissimilaridade máxima ξ_q ; (b) Junção por vizinhança com $k = 3$; (c) Junção por proximidade com $k = 3$.

tuplas $\langle t_i, t_j \rangle$ em que $t_i[S_1]$ é um dos k elementos em S_1 mais próximos a $t_j[S_2]$. Esse operador é muito utilizado em aplicações que lidam com classificação ou agrupamento. Ele garante que cada tupla da relação T_2 esteja associada a k tuplas da relação T_1 – desde que a cardinalidade de S_1 seja ao menos k . Portanto, a quantidade de elementos retornados é k vezes a cardinalidade da relação T_2 .

O operador de **junção por proximidade** $\left[T_1 \overset{(S_j \text{ } k\text{-}CN(d,k) \text{ } S_i)}{\bowtie} T_2 \right]$ retorna os k pares de elementos mais próximos entre si. Essa forma de junção é interessante porque retorna a quantidade k de pares definida na consulta, mas pode ocorrer que algumas tuplas tanto de T_1 quanto de T_2 não apareçam nenhuma vez na resposta. Em contraste com a junção por proximidade, o operador de junção por vizinhança garante que cada elemento de S_2 aparece no conjunto resposta exatamente k vezes. Tuplas de S_1 podem ocorrer no conjunto resposta zero ou mais vezes. Note-se que a junção por abrangência e a junção por proximidade são comutativas (isto é, $T_1 \overset{(S_j \text{ } Rng(d,\xi) \text{ } S_i)}{\bowtie} T_2 \equiv T_2 \overset{(S_i \text{ } Rng(d,\xi) \text{ } S_j)}{\bowtie} T_1$ e $T_1 \overset{(S_j \text{ } k\text{-}CN(d,k) \text{ } S_i)}{\bowtie} T_2 \equiv T_2 \overset{(S_i \text{ } k\text{-}CN(d,k) \text{ } S_j)}{\bowtie} T_1$), mas a junção por vizinhança não é.

A Figura 3.4 mostra exemplos dos três operadores de junção por similaridade: junção por abrangência, junção por proximidade e junção por vizinhança. Os pontos marcados como estrelas vermelhas (S_1, S_2, S_3) correspondem a tuplas da relação T_2 e as bolas pretas correspondem a tuplas da relação T_1 .

A Tabela 3.1 reúne os vários operadores de busca por similaridade descritos, com suas respectivas variantes e extensões, mostrando uma notação unificada para todos eles, que é adequada para ser utilizada em conjunto com os demais operadores da álgebra relacional. A tabela

Consulta	Representação	Param. Consulta	Pré-definido
Seleção por Abrangência	$\sigma(\{S_j \text{ Rq}(d_j, \xi_q) s_q\})^\top$	ξ_q	d_j
Seleção Pontual	$\sigma(\{S_j \text{ Rq}(d_j, 0) s_q\})^\top$		d_j
Seleção por Abrangência Reversa	$\sigma(\{S_j \text{ Rq}(d_j, \bar{\xi}_q) s_q\})^\top$	ξ_q	d_j
Seleção por Abrangência Agregada	$\sigma(\{S_j \widehat{\text{Rq}}(d_j, \gamma, \xi_q) Q\})^\top$	ξ_q	d_j, γ
Seleção por k-vizinhos mais próximos	$\sigma(\{S_j \text{ k-NN}(d_j, k) s_q\})^\top$	k	d_j
Seleção do mais próximo	$\sigma(\{S_j \text{ k-NN}(d_j, 1) s_q\})^\top$		d_j
Seleção por k-vizinhos mais distantes	$\sigma(\{S_j \text{ k-NN}(d_j, k) s_q\})^\top$	k	d_j
Seleção por k-vizinhos mais próximos agregados	$\sigma(\{S_j \widehat{\text{k-NN}}(d_j, \bar{k}) Q\})^\top$	k	d_j, γ
Seleção por k-vizinhos mais próximos reversos	$\sigma(\{S_j \text{ k-RNN}(d_j, k) s_q\})^\top$	k	d_j
Seleção do vizinho mais próximo reverso	$\sigma(\{S_j \text{ k-RNN}(d_j, 1) s_q\})^\top$		d_j
Junção por Abrangência	$\text{T}_1 \overset{S_j \text{ Rq}(d_j, \xi_q) S_i}{\bowtie} \text{T}_2$	ξ_q	d_j
Junção por k-vizinhos mais próximos	$\text{T}_1 \overset{S_j \text{ k-NN}(d_j, k) S_i}{\bowtie} \text{T}_2$	k	d_j
Junção por k-pares de vizinhos mais próximos	$\text{T}_1 \overset{S_j \text{ k-CN}(d_j, k) S_i}{\bowtie} \text{T}_2$	k	d_j

Tabela 3.1: Operadores por similaridade ($S_j \in \text{T}_1; S_i \in \text{T}_2; \mathbb{S}_j = \mathbb{S}_i$).

também indica quais são os parâmetros que devem ser necessariamente providos em cada consulta que os utilizem e quais são os parâmetros que podem ser assumidos (*default*) em uma eventual linguagem de consulta que venha a ser desenvolvida para incluí-los como operadores de consulta em um SGBDR. Assumimos aqui que os parâmetros que devem ser assumidos são a função de distância d_j associada ao domínio do elemento S_j em questão, e a função de agregação γ quando está envolvido um operador de consulta por similaridade agregada. Implementando esses operadores assumindo valores *default*, ou que podem ser estabelecidos por sessões de consulta, facilita expressar consultas, pois esses parâmetros são essencialmente invariantes para um dado domínio de dados ou aspecto de interesse de consulta de um usuário. Todos esses operadores vêm sendo definidos em razão de sua necessidade em determinadas aplicações, e os estudos que vêm sendo feito sobre eles visam a criação de métodos de indexação e algoritmos de busca que possam executá-los com cada vez maior velocidade, em diferentes tipos de dados complexos. Sua utilização em conjunto, incluindo diversos operadores de busca baseados em similaridade, igualdade e RO apenas agora está começando a ser estudada [11, 56, 96].

3.2 Métodos de Acesso Métricos

Os índices tradicionais utilizados pelos SGBDRs aproveitam as propriedades de identidade e de RO para localizar rapidamente a posição de cada elemento na estrutura. Por exemplo, um índice de palavras utiliza o fato de que a letra B sucede (baseado na ordem lexicográfica alfanumérica) que a letra A, a letra C sucede a letra B e assim por diante, possibilitando a criação de uma

estrutura de acesso rápido aos dados armazenados a partir de uma chave, seja ela uma cadeia de caracteres, como no exemplo acima, ou um valor numérico, ou também uma data. No caso das estruturas *hash*, onde os operadores baseados em RO não precisam ser usados, a comparação por identidade é fundamental. No entanto, como praticamente inexitem dois elementos complexos idênticos, uma estrutura de *hash* seria também bem pouco adequada. portanto, em se tratando de dados complexos, como não é possível utilizar as propriedades da identidade e das RO, é necessário buscar outras propriedades que permitam indexar os dados de alguma maneira, visando diminuir o número de acessos a disco e no caso de dados complexos, também reduzir o número de comparações entre os elementos (reduzir a quantidade de cálculos de distância), e para isso são utilizados os Métodos de Acesso Métricos (*Metric Access Methods - MAM*).

Para ser possível definir os Métodos de Acesso Métricos (MAMs), é necessário antes definir o que é um espaço métrico, pois os MAMs que vêm sendo desenvolvidos em sua ampla maioria utilizam as propriedades dos espaços métricos no lugar de propriedades baseadas em RO para construir as estruturas de dados utilizadas para indexação de dados complexos [88, 97]. As propriedades de um espaço métrico devem ser garantidas pela função de distância definida para o espaço. Assim, usando funções de distância que garantam determinadas propriedades, garante-se que será possível utilizar técnicas de indexação adequadas, que podem agilizar operações de busca no espaço utilizado para representar os dados [83, 44].

Formalmente, um espaço métrico é definido como o par $\langle \mathbb{S}, d \rangle$, onde $\mathbb{S}$ é o conjunto de todos os elementos que atendem aos requisitos do espaço (também chamado o domínio dos dados), e $d : \mathbb{S} \times \mathbb{S} \rightarrow \mathbb{R}^+$ é uma função de distância, chamada também de métrica, que atende às quatro propriedades seguintes para qualquer $s_1, s_2, s_3 \in \mathbb{S}$ [60]:

- **Não-negatividade:** $0 \leq d(s_1, s_2) \leq \infty$;
- **Identidade dos indiscerníveis:** $d(s_1, s_2) = 0 \Leftrightarrow s_1 = s_2$;
- **Simetria:** $d(s_1, s_2) = d(s_2, s_1)$;
- **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 num espaço métrico se $S \subset \mathbb{S}$. A métrica é usada para quantificar quão similar são dois elementos, e assim habilita que se expressem consultas baseadas em similaridade. Neste projeto chamamos o par $\langle \text{características}, \text{função de distância} \rangle$ de “Descritor”. Cada descritor forma um espaço métrico [81]. Assim, 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, englobando os casos onde os vetores de características também formam espaços vetoriais [70, 64]. De fato, com essas funções respeitando as propriedades de uma métrica, os operadores de comparação por similaridade se aplicam a muitos tipos de dados complexos, incluindo dados multidimensionais [33, 32], e se valendo dessas propriedades é possível criar estruturas de acesso para dados em espaços métricos, de maneira semelhante ao que é feito com os índices sobre dados tradicionais, que utilizam propriedades de RO. Vale notar que para

aplicações em Bases de Dados, a função de distância é considerada como uma “*caixa-preta*”, geralmente definida por um especialista no domínio da aplicação.

Na literatura existe uma vasta gama de MAMs, que começaram a ser estudados o trabalho de Burkhard e Keller [22]. O trabalho descreve a BK-Tree e propõem técnicas de particionamento do espaço métrico de forma recursiva, que são a base para construção de MAMs. A ideia geral da maioria dos MAMs é selecionar um (ou mais) elemento(s) e utilizá-lo como representante de um subconjunto dos dados, se valendo das relações de similaridade entre os elementos para selecioná-los. Assim, para cada novo elemento inserido no conjunto são calculadas e armazenadas as distâncias entre ele e seus representantes, com elas determinando a qual subconjunto cada novo elemento pertence ou, caso seja necessário, determinar novos representantes. Conhecendo as distâncias entre todos os elementos e representantes é possível utilizar as propriedades da métrica para criar índices ou, em consultas aos dados, podar elementos que não fazem parte da resposta sem precisar avaliar todos os elementos e realizando muito poucos cálculos adicionais de distâncias. A BK-Tree, por exemplo, divide um conjunto de dados recursivamente, selecionando um elemento aleatório como raiz e, a partir dele gera uma subárvore para cada distância possível, até que todos os elementos estejam na árvore. Esse trabalho inicial é bastante simples, e ele funciona apenas com distâncias discretas, o que o torna bastante limitado.

A partir da BK-Tree diversos outros trabalhos foram gerados, como a GH-tree (*Generalized Hyperplane Tree* - que divide o espaço em hiperplanos) [87], a VP-tree (*Vantage Point Tree*), que divide o espaço em bolas centradas em um “*vantage point*”, ou ponto de vista, que serve de pivô em níveis da árvore e determina as subárvores baseado nas distâncias aos pivôs [93], MVP-Tree (*Multi-Vantage Point Tree*, que estende a VP-tree) [19, 20], SA-Tree (*Spatial Approximation Tree*) [67], BU-Tree (*Bottom-up index Tree*) [61], M-Tree [27], Slim-Tree [85], a família Omni [86] e DBM-Tree (*Density Based Metric Tree*) [89]. Como é possível ver, a literatura é vasta, e existem diversos MAMs na literatura. No entanto, para o contexto deste trabalho, é interessante que além de prover o suporte à indexação para os dados, a estrutura também seja dinâmica, ou seja, que permita inserir e remover elementos após a estrutura já ter sido construída e alimentada com os dados iniciais. Essa característica é fundamental para que a estrutura possa ser utilizada junto com um SGBDR, pois os dados não são estáticos. Assim, dentre as várias estruturas mencionadas, a Slim-Tree e a M-Tree são as principais estruturas dinâmicas consideradas. A Slim-Tree estende as capacidades da M-Tree principalmente por permitir reduzir a sobreposição de elementos das subárvores através do algoritmo Slim-down. Mais detalhes sobre a Slim-Tree são apresentados na Seção 3.2.1, focando na implementação e no que será utilizado no contexto deste projeto. Em Zezula et al. [97] é possível encontrar uma descrição mais detalhada de alguns dos principais MAMs presentes na literatura.

3.2.1 Slim-Tree

A Slim-Tree possui uma estrutura em árvore dinâmica (que permite inserções após a criação da árvore) e balanceada (todas as folhas possuem a mesma altura), tendo sua altura total é minimizada e crescendo de baixo para cima (*bottom-up*), características que a tornam adequada para utilização em um SGBDR, pois os dados, geralmente, não são estáticos, podendo ocorrer continuamente um grande volume de alterações na base. Além disso, a Slim-Tree se diferencia de outros MAMs por possuir o algoritmo “*Slim-down*”, o qual busca diminuir a sobreposição de elementos entre os diversos nós. A alta sobreposição é um fator agravante para a ineficiência da árvore métrica [85], e o Algoritmo “*Slim-down*” faz com que a ela se torne mais compacta e eficiente. Na Slim-Tree, os elementos do conjunto de dados são agrupados em páginas de disco de tamanho fixo, com cada uma correspondendo a um nó da árvore, sendo que os dados são armazenados nas folhas [85].

Semelhante a outros MAMs, a Slim-Tree procura organizar os elementos em uma estrutura hierárquica, utilizando um elemento representativo de cada região, para assim superar o fato de que não há comparações por RO e conseguir dividir o espaço nas regiões cobertas pelos nós. Os nós da Slim-Tree são divididos em dois tipos principais: nós folha e nós índice. Os nós folha armazenam os elementos de fato e os nós índice armazenam os dados necessários para navegar pela árvore (como ponteiros para subárvores, raio de cobertura de cada região e distâncias entre cada elemento armazenado no nó ao elemento representativo do nó) e encontrar os elementos requisitados. A figura 3.5 representa a distribuição lógica dos dados em uma Slim-Tree e a figura 3.6 representa a estrutura espacial de uma árvore armazenando dados de um espaço bidimensional usando a distância Euclidiana. Na representação espacial, os círculos brancos representam os nós folha e os círculos cinzas representam os nós índice. As estrelas vermelhas representam os elementos representantes de cada região centrada neles, e por fim os pontos pretos representam os demais elementos indexados. Já a estrutura lógica representa a árvore que seria gerada pela representação espacial ilustrada.

Para criar essas estruturas os elementos são inseridos da seguinte maneira: a partir do nó raiz, o algoritmo tenta localizar um nó que seja adequado para receber o novo elemento pela sua região de cobertura já incluir esse elemento. Se nenhum nó atender a esse critério, será selecionado o nó cujo centro é o mais próximo do elemento. Caso mais de um nó atenda ao critério, será executado o algoritmo *ChooseSubTree* para selecionar um deles. Esse processo é aplicado recursivamente para todos os níveis da árvore. [85]. A Slim-Tree provê três opções para o algoritmo *ChooseSubTree*:

- **random** – Seleciona aleatoriamente entre os nós possíveis;
- **mindist** — A escolha recai sobre o nó que possuir a menor distância entre o novo elemento e o elemento representativo do nó;
- **minoccup** - Escolhe o nó com a menor ocupação dentre os possíveis.

Figura 3.5: Representação lógica de uma Slim-Tree com 17 elementos e com capacidade máxima do nó igual a 3.

Figura 3.6: Representação num espaço bidimensional Euclidiano de uma Slim-Tree com 17 elementos e com capacidade máxima do nó igual a 3.

Quando um nó m precisar armazenar mais elementos do que seu espaço de memória comporta, um novo nó m' é alocado no mesmo nível e os elementos já armazenados mais o novo elemento são distribuídos entre os nós m e m' . Com a separação de um nó em dois, um novo elemento representante é propagado para o nível imediatamente superior. Se esse nó também estiver cheio, ele é por sua vez desmembrado em dois, em um processo recursivo. Caso m seja o nó raiz, a árvore crescerá um nível, gerando uma nova raiz com dois elementos representantes apontando para os nós m e m' . A divisão de um nó em dois na *slim-tree* tem três possibilidades [85]:

- **random** - Os dois novos elementos centrais são selecionados aleatoriamente e os elementos existentes são distribuídos entre eles, seguindo o critério da menor distância entre eles e o novo centro;
- **minMax** - Todos os pares de elementos possíveis são considerados como os novos centros. Para cada par possível, é utilizado um algoritmo linear para designar elementos para cada um dos representantes. O par que minimizar o raio de cobertura é escolhido;
- **MST** - É gerada uma árvore de caminhos mínimo (*minimal spanning tree* - *MST*) [57] sobre os elementos a serem distribuídos e um dos maiores arcos da árvore é removido.

Apesar das similaridades com a M-Tree, a Slim-Tree se destaca pela adição do algoritmo de divisão dos nós baseado na MST, do algoritmo de inserção de elementos nas subárvores apropriadas e, principalmente, pela adição do algoritmo *Slim-down*, que cria árvores mais “esguias” e que propiciam acesso significativamente mais rápido [85].

3.3 Integração de Consultas por Similaridade em SGBDRs

Como foi visto na Seção 3.1, como consequência do aumento da complexidade e do volume dos dados, foi gerada uma grande quantidade de pesquisas para possibilitar que eles sejam tratados e manipulados de forma eficiente e eficaz. No entanto, é bastante comum a necessidade de se utilizar não apenas dados complexos em uma consulta, mas também dados escalares (números, pequenas cadeias de caracteres ou datas) que obedecem à relação de ordem e/ou que podem ser comparados por igualdade. Para os dados tradicionais, os SGBDRs modernos oferecem um suporte bem consolidado, como foi visto no Capítulo 2. No entanto, quando se trata de combinar os dois tipos de dados em uma única consulta, as tecnologias existentes ainda estão em estágio inicial, e se faz necessário estudar novos conceitos e técnicas, e soluções mais refinadas, para os problemas existentes. Sendo assim, esta seção faz uma breve revisão dos trabalhos da área que procuram integrar consultas envolvendo operadores de busca por similaridade e operadores de busca por identidade ou RO, além de discutir alguns pontos provenientes dessa integração.

3.3.1 Abordagens Existentes

A integração de consultas que envolvem operadores de busca por similaridade junto com operadores de busca por identidade e por relações de ordem ainda não está consolidada na literatura, mas existem diversos trabalhos que procuram interpretar e resolver o problema. O primeiro trabalho que procurou desenvolver uma álgebra por similaridade foi de Adali et al. [3], que argumentou sobre a necessidade de se formalizar as buscas em bases de dados multimídia e fornecer meios para otimizar essas buscas. Aquele trabalho pioneiro procurou integrar a álgebra por similaridade com os operadores relacionais, criando a MSA (“Multi-Similarity algebra”). No entanto, a MSA, apesar de apresentar uma abordagem introdutória ao tratamento do problema, ainda era muito abstrata [7] e não abordava as questões práticas necessárias.

A partir daí, surgiram diversos trabalhos e abordagens, alguns procurando tratar a questão através da OQL (*Object Query Language*) [24], uma alternativa orientada a objetos do SQL, como descrito no trabalho de Li et al. [59]. Outras abordagens procuraram utilizar lógica nebulosa (*fuzzy*) [26, 69], argumentando que a imprecisão proveniente da lógica nebulosa auxilia semanticamente na representação dos dados complexos. Os autores, inclusive, afirmam que a lógica nebulosa provê mais auxílio em eliminar o “*gap* semântico” do que a combinação de características extraídas e “*tags*” associadas a cada elemento de dado, argumentando que a lógica nebulosa é mais adequada para capturar conceitos como por exemplo a corrente artística de uma pintura ou gênero musical de uma canção. No entanto, é possível analisar o efeito da utilização de lógica nebulosa apenas como consequência da falta de operadores por RO e igualdade, pois atributos como corrente artística e gênero musical são atributos (“*tags*”) textuais associados a atributos complexos e são estes que de fato não podem ser comparados entre si por RO nem por identidade. Isso decorre da natureza dos dados complexos, onde cada domínio de dados pode necessitar de uma técnica específica para a extração de características, bem como uma ou mais funções de distância. Assumir que imprecisão é fator é intrínseco aos dados complexos nem sempre é correto, eles apenas não possuem propriedades para serem usados em comparações por RO e por igualdade.

Alguns outros trabalhos [1, 2, 58] exploraram ainda a extensão do conceito de consultas baseadas em ordenação (chamadas *ranked-based* ou *top-k queries*), utilizando funções de distância para ordenar o resultado das consultas, permitindo sua aplicabilidade no modelo relacional e efetivamente integrando os operadores de busca por similaridade com os de busca por igualdade ou RO.

Outra maneira bastante estudada para integrar os operadores por similaridade e os baseados em igualdade e RO é criar uma álgebra que inclua a ambos, adicionando a eles os operadores por similaridade da álgebra relacional, e assim criando uma álgebra por similaridade. Essa linha foi tomada nos trabalhos de Atnafu, Brunie e Kosch [8] e Atnafu et al. [9], mas ainda sem investigar todos os operadores por similaridade e suas propriedades. Nos trabalhos de Ferreira et al. [39], Ferreira et al. [40] e Traina Jr. et al. [84] as propriedades algébricas de todos os operadores por

similaridade unários são mais minuciosamente detalhadas e é possível notar os benefícios de se utilizar essa linha de integração, uma linha mais próxima da estrutura tradicional dos SGBDRs, adicionando os operadores de busca por similaridade através da extensão da álgebra, permitindo que as consultas sejam otimizadas pelas propriedades algébricas, assim como no modelo relacional original. Inclusive, é sugerido que, como consequência de possuir propriedades distintas dos outros operadores unários, o operador de seleção por k -NN é mais adequadamente representado como $\ddot{\sigma}$, em vez do σ tradicional, visando distinguir ambos.

Também é possível mencionar os trabalhos de Belohlávek, Urbanova e Vychodil [16], Belohlávek e Vychodil [17] e Belohlávek, Opichal e Vychodil [15], que combinam a abordagem algébrica e a lógica nebulosa, propondo uma extensão baseada em rankings para o modelo relacional combinada com imprecisão para consultas incertas. Essa ideia foi expandida no trabalho de Belohlávek e Vychodil [18], onde os autores propõe um modelo “*multi-ranked*”, designando *rankings* para valores específicos das tuplas, em vez da tuplas inteira.

Silva et al. [76] apresenta um sistema gerenciador de bases de dados ciente de similaridade, estendendo o SGBDR PostgreSQL e utilizando suas capacidades, junto com regras de transformação de consultas para suportar a implementação de otimizadores de planos de consulta com diversos operadores por similaridade e tradicionais. No trabalho de Silva et al. [77] os autores avançam essa ideia, estudando como operadores cientes de similaridade devem ser tratados na base em um contexto onde o processamento de consultas com múltiplos operadores está presente, buscando permitir uma melhor avaliação e otimização de operadores por similaridade complexos que envolvem múltiplos operadores mais simples como seleção, junções e agrupamento. Em linha semelhante, mas mais específica, os trabalhos de Silva, Aref e Ali [74], Silva e Pearson [75] e Marri et al. [65] estudam os operadores de agrupamento, junção e intersecção por similaridade, respectivamente, em um contexto relacional. Continuando a estudar operadores individualmente, o trabalho de Carvalho et al. [23] sugere que o operador de junção por similaridade por proximidade não é uma junção “clássica”, e sim uma operação que inclui o conceito de junção, sugerindo três operadores derivados, os inserindo na algebra relacional, e implementando-os em um SGBDR.

Algumas soluções são mais objetivas, focando em lidar com tipos de dados ou aplicações específicas, como a recuperação de imagens por conteúdo. Neste contexto, alguns trabalhos que podem ser mencionados são o FMI-SiR [52], que aproveita o ferramental da base de dados Oracle e de seu pacote interMedia para manipular dados multimídia e expande as suas capacidades existentes. Com o escopo mais preciso, visando tratar imagens médicas e suas particularidades, o FMI-SiR ainda é especializado no MedFMI-SiR [53]. Para tratar da recuperação de imagens por conteúdo especificamente, existe ainda o PostgreSQL-IE [45], que estende a base PostgreSQL para habilitar o tratamento e recuperação de imagens.

Por fim, no trabalho de Budíková, Batko e Zezula [21] os autores utilizaram um arcabouço desenvolvido previamente por Batko, Novak e Zezula [14] para consultas por similaridade em espaços métricos e procuraram estender a linguagem SQL para atender às capacidades providas

pelo arcabouço. Esses trabalhos ainda são bastante recentes, eles não focam tanto na questão da integração, pois o arcabouço foi feito mais especificamente para buscas por similaridade em espaços métricos, se preocupando com as questões da linguagem e não com possíveis problemas de eficácia e eficiência na realização das consultas que envolvam operadores diversos. No entanto, é feita uma análise de requisitos valiosa para trabalhos que procurem adequar a linguagem SQL para atender a consultas por similaridade, além da linguagem desenvolvida ser bastante flexível. Assim, apesar desses trabalhos ainda estarem em estágio inicial, os trabalhos futuros decorrentes podem ser promissores.

3.3.2 O SIREN

No contexto deste trabalho, no que diz respeito à integração de consultas utilizando operadores de comparação por similaridade e operadores de busca tradicionais dos SGBDs relacionais, será utilizada a ferramenta *Similarity Retrieval Engine* (SIREN), desenvolvida no trabalho de Barioni et al. [13]. O SIREN é ideal para o desenvolvimento deste trabalho pois apresenta uma extensão da linguagem SQL que lida com dados complexos além de um *middleware* que interliga um compilador e interpretador da linguagem estendida com um MAM e uma base de dados relacional. Fazendo essa interligação, o SIREN consegue distinguir as partes que tratam de dados complexos das que tratam de dados tradicionais, e lida de maneira transparente com a comunicação entre compilador, SGBDR e MAM, além da criação de estruturas de suporte aos dados complexos. A extensão da SQL corresponde a construções sintáticas para indicar operações de busca por similaridade. Se alguma dessas construções ocorre no comando, elas são tratadas de maneira diferenciada, reescrevendo o comando e tratando as operações necessárias internamente ao SIREN, enquanto as operações sobre os dados tradicionais são tratadas no SGBDR.

Como é ilustrado na Figura 3.7, retirada do trabalho de Barioni et al. [13], a arquitetura inicial do SIREN possuía três principais componentes. O interpretador de comandos, como já foi comentado, funciona como um compilador e interpreta o comando SQL recebido pela ferramenta. O extrator de características é o responsável por, como o nome indica, extrair as características de elementos complexos para que seja possível representá-los e indexá-los. Sempre que um novo elemento de dados complexo é inserido na base, suas características são extraídas e armazenadas. Por fim, o indexador utiliza um MAM (Slim-Tree) para executar as consultas por similaridade de maneira eficiente, atuando como um índice primário para os elementos complexos, implementado pela biblioteca Arboretum [43] (uma biblioteca *open-source* que implementa diversos MAMs), para prover acesso rápido aos dados buscando minimizar número de acessos ao disco e cálculos de distância realizados durante o processamento dos operadores de busca por similaridade.

Para lidar com os diferentes tipos de dados complexos, foram definidos dois “superdomínios”: PARTICULATE e MONOLITHIC, que se diferenciam sobre a maneira como os dados complexos propriamente ditos são armazenados na base. O primeiro armazena cada elemento

Figura 3.7: Arquitetura inicial do SIREN (retirada do trabalho de Barioni et al. [13])

como sendo constituído por vários pedaços de dados tradicionais, como por exemplo: os dados geográficos, os quais podem ser representados por duas coordenadas (dois atributos escalares); ou as séries temporais que também são armazenadas como sequências de dados escalares. O segundo considera que cada elemento de dados é armazenado como um único elemento binário não analisável pelo SGBD tradicional, ou seja, como um dado BLOB (*Binary Large Object*). Exemplos deste caso são imagens, áudios, vídeos, sequências genéticas etc. [13].

Quando as consultas por similaridade tratam dados **PARTICULATE**, elas são intrinsecamente mais simples, pois basta associar uma função de distância e é possível comparar dois atributos diretamente, mesmo num SGBD tradicional. Isso porque os dados **PARTICULATE** são compostos de atributos escalares que já são tratados pelos SGBDR. No entanto, quando se trata de dados **MONOLITHIC**, é necessário extrair características representativas desses dados para que eles só então sejam comparados. A extração, a armazenagem e a indexação de características são atividades trabalhosas, que devem ser abstraídas das aplicações e tratadas pelo SGBDR. Como **MONOLITHIC**, o SIREN permite que sejam definidos vários tipos de dados, como por exemplo **STILLIMAGES** (imagens) ou **AUDIO**, permitindo especificar o tipo que está sendo tratado e os correspondentes extratores de características que podem ser aplicados (imagens no primeiro caso e áudio no segundo). A partir dos vetores de características extraídos, devem ser então associadas uma ou mais funções de distância, para que seja possível medir qual é a proximidade entre cada par de elementos seguindo um determinado descritor, isto é, cada par característica/função de distância. Na versão inicial do SIREN, por exemplo, a partir de atributos de tipo **STILLIMAGE** era possível extrair características como textura [37], forma, baseada em momentos de Zernike[51], ou histogramas normalizado de cores [78].

Os únicos comandos novos que o SIREN inclui na linguagem SQL destinam-se a tratar da necessidade de utilização de métricas quando lidando com dados complexos (as demais extensões apenas alteram a sintaxe dos comandos já existentes na linguagem). Tais comandos são **CREATE METRIC**, **ALTER METRIC** e **DROP METRIC**, que são comandos da *Data Definition Language (DDL)*. Essa extensão permite que pares função de distância e extrator de características sejam criados e associados a um ou vários elementos complexos, habilitando sua utilização durante consultas envolvendo operadores por similaridade. A definição da métrica associada ao atributo complexo num comando **CREATE METRIC** contempla a especificação de todos os aspectos necessários para a avaliação de similaridade, incluindo extração de características (para domínios **MONOLITHIC**), normalização de atributos e comparação entre pares de elementos.

Por fim, ainda é necessário tratar o armazenamento dos dados complexos na base. Para dados **PARTICULATE** não é necessário modificar nada, uma vez que esse tipo de dado é composto de atributos escalares e seus vários componentes são armazenados diretamente na tabela como diversos atributos. No entanto, para tratar dados como **STILLIMAGES** ou **AUDIO**, é necessário incluir também as características extraídas. Como o usuário não deve precisar explicitar nenhum atributo na tabela para armazenar essas características, cabe ao sistema prover um espaço para elas. Assim, a associação de tais características com o BLOB correspondente é feita de forma transparente ao usuário. Para isso, em vez de armazenar o valor do elemento complexo no atributo que o usuário criou no comando **CREATE TABLE** que definiu a tabela onde o atributo complexo existe, o SIREN armazena uma referência para uma tabela controlada pelo sistema que de fato armazena o elemento em forma de BLOB. Como essa tabela é controlada pelo sistema, pode-se acrescentar automaticamente a ela as características extraídas em cada extrator utilizado. Portanto, para cada atributo complexo, o SIREN cria uma tabela interna e toda vez que o usuário utiliza um comando **INSERT** o SIREN irá interceptá-lo e realocará os dados adequadamente nas tabelas. Quando uma consulta necessita dos dados complexos, o SIREN executa uma junção entre as tabelas do usuário e as tabelas internas do sistema, removendo as características e fazendo com que o resultado final seja transparente para o usuário, o qual sempre vê como resposta uma tabela que tem a estrutura que ele criou [13].

A seguir são exemplificados os comandos necessários para criar uma tabela que possua um atributo complexo e associar a métrica que será utilizada nas consultas realizadas utilizando operadores por similaridade sobre atributos complexos.

Criação de uma métrica:

Neste exemplo deve-se criar uma métrica chamada **HistLp2**, que usa características extraídas de imagens (**STILLIMAGE**) usando como extrator um histograma de cor (indicado por **HISTOGRAMEXT**). A função de distância é a métrica euclidiana L_2 (indicada como **LP2**). O comando em SQL estendido é:

```
CREATE METRIC HistLp2
  USING LP2 FOR STILLIMAGE(HISTOGRAMEXT);
```

Criação de uma tabela com um atributo complexo:

O comando seguinte cria uma tabela *Itens* com um atributo *Monolithic* de tipo imagem chamado *Foto* e outros atributos de tipos escalares.

```
CREATE TABLE Itens AS (
  Código INTEGER PRIMARY KEY
  Nome VARCHAR(50),
  Data DATE,
  Foto STILLIMAGE,
  METRIC (Foto) USING (HistLp2)
);
```

O atributo complexo “Foto” é definido como sendo do tipo *STILLIMAGE*, que é um dos novos tipos *MONOLITHIC* criados pelo SIREN. A ele é associada a métrica *HistLp2* definida anteriormente. A associação da métrica é considerada uma restrição feita ao atributo (como a indicação de *PRIMARY KEY FOREIGN KEY*, e portanto pode ser feita como restrição de tabela ou de atributo. Nesse exemplo, ela foi feita como restrição de tabela.

Para possibilitar a realização de consultas utilizando os operadores por similaridade, a linguagem SQL foi também estendida para acomodar consultas aos *k*-vizinhos mais próximos e por abrangência, além das junções por abrangência, por proximidade e por vizinhança. As variações dos comandos de seleção descritas na tabela 3.1 também são atendidas com variações nas construções sintáticas, sendo destacadas as consultas de seleção por similaridade agregada. Por exemplo, utilizando a métrica e a relação criadas nos exemplos anteriores, para consultar o código e o nome dos 5 vizinhos mais próximos de uma determinada imagem, o comando em SQL seria:

```
SELECT código, nome
  FROM Itens
 WHERE Foto NEAR (SELECT Foto FROM Itens WHERE código = 10)
    STOP AFTER 5;
```

Neste exemplo, a cláusula *WHERE* está utilizando o atributo complexo “Foto” como filtro num predicado de *k*-*NN*, expresso como “Foto NEAR (SELECT Foto FROM Itens WHERE código = 10)”. A primeira referência a *Foto* nesse predicado corresponde ao atributo *FOTO* da tabela que

está sendo pesquisada, o qual é comparado por similaridade (indicado pelo operador de comparação **NEAR**) com a **Foto** do item cujo **código** tem valor 10, obtido pela sub-seleção indicada entre parênteses, que será utilizado como centro da consulta. Também é possível explicitar como centro da consulta uma imagem armazenada no computador em vez de realizar uma busca na base.

A cláusula “**STOP AFTER 5**” está associada ao operador **NEAR** indicando o predicado k -*NN*, e que serão selecionados os 5 vizinhos mais próximos. Caso se desejasse realizar a consulta utilizando abrangência em vez de k -*NN*, o comando “**STOP AFTER**” seria substituído por “**RANGE ξ** ” indicando que devem ser selecionados os elementos que distam até ξ unidades de distância do centro de consulta.

Note-se que por haver sido utilizado o predicado **WHERE código = 10** na sub-seleção, onde o atributo **código** é chave da relação, o resultado da sub-seleção terá no máximo uma **Foto**. Portanto, nesse exemplo o operador **NEAR** tem um centro de consulta e o operador de comparação é um k -*NN* simples. Se o atributo usado na sub-seleção não fosse chave, o resultado poderia ter diversas tuplas. Nesse caso, haveriam diversos centros de consulta, e o operador de comparação seria um k -*NN* por similaridade agregada.

O SIREN conta ainda com diversos operadores de comparação por similaridade para responder às consultas que além de seleção por similaridade (por abrangência, por k vizinhos mais próximos e suas respectivas variações), envolvam também junção por similaridade. As junções por similaridade são indicadas usando a mesma sintaxe das seleções por similaridade, mas os predicados são expressos na cláusula **FROM** do comando **SELECT**, seguindo a mesma sintaxe dos operadores tradicionais. A variação de seleção do vizinho mais próximo reverso é expressa para o SIREN como uma variação da sintaxe para a junção por proximidade.

Percebe-se assim que a extensão da linguagem SQL atendida pelo SIREN tem muito poucas alterações sintáticas sobre o SQL padrão, mas é muito poderosa para expressar consultas por similaridade, pois consegue representar de maneira simples e intuitiva a grande maioria dos operadores de consulta por similaridade que vêm sendo tratados na literatura – pelo menos todos aqueles discutidos neste trabalho. No entanto, a experiência que vem sendo obtida com o uso prático desses conceitos, tornada possível com a disponibilidade do SIREN, mostra que existem situações de conflito, principalmente semântico, que ocorrem em casos limites do uso desses operadores junto aos demais conceitos da teoria relacional. Neste trabalho de mestrado, investigam-se dois conflitos:

- Múltiplas tuplas com o mesmo valor do atributo complexo,
- Empates nas medidas de similaridade de elementos que podem ocupar a k -ésima posição na resposta a uma busca por k -*NNq*.

Em bases de dados tradicionais, a otimização de consultas é de grande importância, pesando de maneira significativa no tempo de processamento para gerar o resultado final e portanto no desempenho do SGBDR [49]. Assim, é natural que se procure estender as ideias desenvolvidas

Figura 3.8: Arquitetura do SIREN com o otimizador de consultas (retirada do trabalho de Ferreira, Traina Jr. e Traina [38]).

nos otimizadores de bases de dados relacionais para um otimizador de consultas por similaridade em dados complexos.

No trabalho de Ferreira, Traina Jr. e Traina [38], foi integrado ao SIREN um otimizador de consultas, baseado em um arcabouço que permite a reescrita de consultas. Ele se vale do fato de que uma mesma consulta pode ser escrita de diversas maneiras, mas nem todas elas possuem o mesmo custo de execução. Para que isso seja possível foram desenvolvidas estruturas de representação dos operadores de similaridade nas árvores de consulta, além de propriedades algébricas que podem ser utilizadas para reestruturar as árvores. Na Figura 3.8 é exemplificada a nova arquitetura do SIREN.

A nova arquitetura preserva a mesma extensão da linguagem SQL e funciona de maneira semelhante à anterior. Da mesma maneira que antes, os comandos que não contêm operadores de consulta por similaridade serão enviados diretamente ao SGBDR para serem executados (e otimizados) lá, enquanto as consultas que incluem operadores por similaridade são processa-

Figura 3.9: Otimizador de consultas do SIREN (retirada do trabalho de Ferreira, Traina Jr. e Traina [38]).

das parcialmente pelo SGBDR e parcialmente pelo SIREN. No entanto, agora as consultas com operadores por similaridade passam primeiro pelo otimizador antes da árvore de comandos ser realmente executada. O otimizador interno do SIREN recebe a árvore canônica de comandos e, utilizando propriedades algébricas baseadas nas propriedades dos operadores de busca por similaridade, efetua diversas modificações na consulta procurando por uma expressão que minimize o custo de acesso. Este processo é ilustrado na figura 3.9. O otimizador reescreve as consultas recebendo como entrada a árvore de comandos canônica obtida pelo módulo *query compiler* (a “*Canonical operator tree*”) e, valendo-se das características que podem ser extraídas, das estruturas de índice e das propriedades algébricas disponíveis, gera uma representação otimizada da consulta (a *Optimized operator tree*).

No trabalho de Ferreira et al. [40] é feito um estudo aprofundado sobre as propriedades algébricas dos operadores por similaridade e é possível verificar, através de ganhos de desempenho e nos resultados das consultas, que o estudo contínuo de tais propriedades é de grande valor, pois assim como nos SGBDRs, as consultas por similaridade podem se beneficiar da otimização dos planos de consulta e acesso, tanto em performance, como em obtenção de um resultado final mais próximo das expectativas do usuário do sistema.

3.4 Considerações Finais

Neste capítulo foram revisados os principais conceitos relacionados a consultas por similaridade e pertinentes à realização deste trabalho, o qual busca avançar o estudo da integração de consultas

por similaridade em SGBDRs. Para isso, foram primeiramente revisados os conceitos de funções de distância e extração de características, seguido dos principais tipos de consulta por similaridade e de uma breve revisão sobre espaços e métodos de acesso métricos. Por fim, foi feita uma breve revisão dos principais trabalhos que procuram integrar consultas por similaridade aos SGBDRs, permitindo que sejam realizadas consultas constituídas de operadores de busca por relação de ordem e identidade sobre dados tradicionais juntamente de consultas por similaridade sobre dados complexos.

CAPÍTULO 4

EMBUTINDO CONSULTAS K - NNQ EM SGBDS RELACIONAIS

4.1 Introdução

O presente trabalho tem como objetivo resolver alguns dos problemas gerados pela inclusão de consultas por similaridade, e recuperação de dados complexos de maneira geral, no contexto do modelo relacional, além de prover ferramentas que facilitem essa integração. As soluções propostas possuem foco no operador k - NN , pois é ele que trás a maioria dos problemas e, devido às suas particularidades, o tratamento do k - NN naturalmente resolve o operador por abrangência Rng .

Para atingir esse objetivo, é necessário definir novos operadores algébricos para o modelo relacional, estendendo as regras algébricas e os algoritmos correspondentes, de maneira a atender às consultas por similaridade, pois a maioria dos componentes de um SGBDR (estruturas de indexação, otimizadores de consulta, estimadores de seletividade) consegue apenas lidar com dados que atendam as propriedades de RO.

Este projeto de mestrado lida com dois problemas associados ao uso do operador de k - NN em um SGBDR, com este capítulo focando na resolução do seguinte problema:

Quando inserido em uma ferramenta relacional que aceita atributos complexos como valores de tuplas, o operador k -NN pode tomar duas interpretações distintas. Isso ocorre pois, conceitualmente, o domínio de um atributo complexo forma um espaço métrico, mas em aplicações práticas diversas tuplas em uma base de dados podem assumir o mesmo valor para o atributo complexo. Portanto, um operador de comparação cuja resposta depende do domínio ativo inteiro proverá respostas distintas para relações de entrada distintas, mesmo que os valores da tupla não mudem. Como consequência, um operador k -NN pode retornar qualquer número $n \geq k$ de tuplas mesmo quando se obtêm apenas k valores distintos de atributos complexos. Essa duplicidade leva à questão de qual resultado a consulta deve retornar de fato, i.e. “o que o usuário espera como resultado final?”. Assim, uma consulta por k -NN deve ser bem definida de maneira que a consulta expresse de claramente se devem ser retornadas k tuplas ou k valores distintos de atributos complexos.

Para ilustrar o problema de interpretação, tome-se como exemplo uma hipotética galeria de arte que vende cópias de pinturas e registra cada venda em um SGBDR na seguinte relação:

`ItemVendido={Item, DataVenda, NomeCliente, NomePintura, NomeArtista, ImgPintura}`

onde `ImgPintura` é um atributo complexo que contém uma imagem da pintura vendida, e portanto é passível de ser buscado por similaridade, como por uma seleção k -NNq. Essa relação pode ser criada como uma tabela na linguagem SQL estendida do SIREN pelo comando:

```
CREATE TABLE ItemVendido AS (  
  Item INTEGER PRIMARY KEY  
  DataVenda VARCHAR(50),  
  NomeCliente VARCHAR(50),  
  NomePintura VARCHAR(50),  
  NomeArtista VARCHAR(50),  
  ImgPintura STILLIMAGE METRIC USING (HistLp2 DEFAULT, TextureLP1)  
);
```

Aqui foi indicado que a imagem tem duas características extraídas: Histograma de cor segundo a mesma definição da métrica `HistLp2` mostrada no exemplo da Seção 3.3.2; e Textura usando a função de distância `Manhattan` pré-definida como `TextureLP1`. A métrica `HistLp2` é usada por `DEFAULT`.

Considere-se a seguinte consulta feita sobre esta relação:

- “ Q_4 - Quais são os itens vendidos cuja imagem é uma das 5 mais similares à imagem s_q ?”

A consulta Q_4 exemplifica a utilização de um operador k -NN simples, contextualizado dentro do ambiente relacional, onde dados complexos são armazenados como valores da relação,

de maneira equivalente aos dados tradicionais. `ImgPintura` é um atributo complexo na relação `ItemVendido` e a consulta requisita as tuplas cuja imagem esteja entre as 5 mais similares ao centro de consulta s_q .

Note que apesar de ser uma seleção k - NNq com $k = 5$, o conjunto resposta da consulta pode ter cardinalidade de $n \geq k$ tuplas, com n dependendo da quantidade de itens com imagem repetidas vendidos, o que é irrelevante para a consulta. No entanto, se o usuário quer exatamente 5 tuplas que possuam uma imagem que esteja entre as mais semelhantes ao centro de consulta s_q , a pergunta seria:

- “*Q5 - Quais são os 5 itens vendidos com imagem entre as mais similares à imagem s_q ?*”

A consulta $Q5$ também é uma aplicação do operador de comparação k - NNq , mas agora com uma perspectiva focada na tupla, ou seja, o gerenciador deve considerar k como sendo o número de tuplas resultantes no conjunto resposta, e não a quantidade de valores de atributos complexos distintos. Essa segunda perspectiva é complementar à da consulta $Q4$, e ambas se dão a partir da inserção do operador k - NN no modelo relacional. Note-se que seguindo a maneira como tradicionalmente as consultas por similaridade são tratadas na literatura, considerando que as buscas são feitas em um conjunto de dados complexos (não uma relação) onde não existem dois elementos com o mesmo valor, ambas as interpretações levam ao mesmo resultado. No entanto, quando o conjunto de busca está armazenado como um atributo complexo de uma relação junto com outros atributos, não existe a imposição que o atributo complexo deva ser chave da relação. Assim, o operador de k - NNq não pode mais ter apenas uma representação. Para melhor ilustrar as duas interpretações, a Figura 4.1 mostra uma representação visual de possíveis respostas às consultas $Q4$ e $Q5$ com o centro de consulta s_q sendo a figura da “Mona Lisa”. Como pode ser visto, os conjuntos que são obtidos para cada interpretação podem variar drasticamente em cardinalidade, de acordo com as particularidades do conjunto de dados. Note-se também que a resposta de $Q4$ contém a resposta de $Q5$.

Neste trabalho as duas interpretações foram incluídas na extensão da linguagem SQL com suporte a operadores por similaridade da ferramenta SIREN apresentada no Capítulo 3. Além disso, adota-se o símbolo $\ddot{\sigma}$, em vez do tradicional σ para representar o operador de seleção por k - NN , pois consultas baseadas neste operador não possuem as mesmas propriedades das demais formas de seleção. A seguir, a Seção 4.2 define um conjunto de operadores algébricos de busca por similaridade que comporta adequadamente a expressão de consultas por similaridade no modelo relacional. A Seção 4.3 define duas interpretações distintas para o operador de seleção por k - NN dentro do contexto relacional. Finalmente, a Seção 4.4 conclui o capítulo.

Figura 4.1: Representação visual das duas interpretações propostas para o operador k -NN em uma consulta sobre a base de imagens de pinturas, tendo a imagem da “Mona Lisa” como centro da consulta s_q . Acima da figura estão indicados os dados representados em cada passo.

4.2 Consultas por Similaridade no Modelo Relacional

Para tratar adequadamente o problema da dupla interpretação, é importante definir mais precisamente como integrar a busca sobre atributos complexos no modelo relacional. Assim, primeiramente são delineadas algumas definições e simbologia que formam um arcabouço para um modelo relacional estendido que se adequa ao paradigma de dados complexos.

No modelo relacional, uma relação T armazenada em uma base – ou o “estado atual da relação”, a que por simplicidade chamamos apenas “relação” quando não há dúvida que se trata da relação atualmente armazenada – é um conjunto de tuplas cujo esquema é denotado por $T = (A_1, \dots, A_n)$, onde $A_1, \dots, A_n$ são os atributos da relação e cada atributo A_i é dito ser um papel adotado por um domínio $\mathbb{A}_i$ na relação, isto é, $Dom(A_i) = \mathbb{A}_i$. Cada tupla é denotada como $t[T]$, e o valor de um subconjunto C de seus atributos é denotado como $t[C]$. O domínio ativo do atributo $A = D^*om(A)$ é o subconjunto de todos os valores $t[A] \in \mathbb{A}$ que ocorrem em pelo menos uma tupla da relação T .

Neste trabalho assume-se que os atributos podem ter tanto de domínios complexos quanto escalares. Para distinguir atributos tradicionalmente tratados em SGBDRs (chamados atributos escalares na teoria relacional) daqueles que podem ser comparados por similaridades, chamamos os últimos de atributos complexos. Note-se que essa distinção não altera o modelo relacional, uma vez que a teoria relacional não especifica os domínios dos quais os atributos podem ser.

Uma relação T pode ter qualquer número de atributos complexos e escalares. Reservamos o símbolo E para representar atributos escalares, os quais podem ser comparados por identidade e por RO, e o símbolo S para referenciar atributos complexos, que podem ser comparados por similaridade. O símbolo A é empregado para indicar tanto atributos complexos quanto escalares. Correspondente ao atributo A , cujo domínio $Dom(A) = \mathbb{A}$ e domínio ativo $D^*om(A) = A$, utilizamos os símbolos $E, \mathbb{E}$ e E para atributos escalares, e os símbolos $S, \mathbb{S}$ e S para atributos complexos. Portanto, o esquema de uma relação pode ser denotado como $(E_1, \dots, E_q, S_1, \dots, S_p)$.

Finalmente, considerando que T é um conjunto de tuplas, a tupla $t_i[T] = \langle e_1, \dots, e_q, s_1, \dots, s_p \rangle \in T$ tem cada valor $e_{ih} = t_i[E_h]$, $(1 \leq h \leq q)$ obtido no domínio $\mathbb{E}_h$ e cada valor $s_{ig} = t_i[S_g]$, $(1 \leq g \leq p)$ obtido no domínio $\mathbb{S}_g$. A quantidade n de tuplas na relação é chamada de cardinalidade, e a quantidade $d = q + p$ é chamada de dimensionalidade. Considera-se que um atributo S é complexo se estiver associado a uma função de distância $d : \mathbb{S} \times \mathbb{S} \rightarrow \mathbb{R}^+$ definida no domínio do atributo $\mathbb{S}$ que mede a distância (ou dissimilaridade) entre quaisquer pares de elementos em $\mathbb{S}$. Assim, um atributo E_i escalar que estiver associado a uma função de distância é também complexo e potencialmente pode ser empregado tanto em consultas por similaridade, quanto em baseadas em comparações por identidade e RO. Quando a função de distância atende às propriedades de uma métrica, diz-se que o atributo está definido num espaço métrico.

Para incluir atributos complexos na álgebra relacional, é necessário reavaliar os operadores

por similaridade que vêm sendo tradicionalmente desenvolvidos, pois é necessário contemplar a mudança de foco para o contexto de tuplas, em vez de apenas um atributo complexo de cada vez.

No Capítulo 3 foram apresentados os dois tipos de operadores por similaridade mais frequentemente estudados na literatura: por abrangência Rq e por k -vizinhos mais próximos $k-NNq$. Uma consulta por abrangência pode ser expressa usando um operador de seleção $\sigma_{(S \text{ } Rng(d,\xi) \text{ } s_q) \text{ } T}$, onde S é um atributo complexo e Rq é o operador de comparação por abrangência, que retorna verdadeiro para cada tupla $t \in T$ da maneira que $d(t[S], s_q) \leq \xi$.

Todas as propriedades algébricas do operador de seleção σ se mantêm quando a comparação emprega o operador de comparação Rq , a seleção por abrangência. Por exemplo, a bastante útil propriedade de comutatividade se mantêm, e portanto:

$$\sigma_{(S_1 \text{ } Rng(d_1,\xi_1) \text{ } s_{q1})} (\sigma_{(S_2 \text{ } Rng(d_2,\xi_2) \text{ } s_{q2})} T) \equiv \sigma_{(S_2 \text{ } Rng(d_2,\xi_2) \text{ } s_{q2})} (\sigma_{(S_1 \text{ } Rng(d_1,\xi_1) \text{ } s_{q1})} T) \quad (4.1)$$

também é garantida para qualquer atributo complexo S_1 e S_2 , função de distância d_1 e d_2 , raio de consulta ξ_1 e ξ_2 e centro de consulta s_1 e s_2 .

Esse tipo de consulta não leva à ambiguidade em sua interpretação, pois, como indicado pela equivalência (4.1), obter “todos os elementos mais próximos que uma certa distância a s_q e selecionar todas as tuplas que tem quaisquer um desses elementos como valores do atributo S ” gera um conjunto igual ao que é gerado por “todas as tuplas cujo valor de S está mais próximo que uma dada distância a s_q ”.

Já uma consulta aos k -vizinhos mais próximos não pode ser expressa usando apenas um operador de seleção. De fato, se tal tentativa for feita, utilizando uma comparação por k -vizinhos mais próximos em uma seleção, como $\ddot{\sigma}_{(S \text{ } k-NN(d,k) \text{ } s_q) \text{ } T}$, o operador resultante não possui várias das propriedades do operador de seleção. Por exemplo, não possui a propriedade de comutatividade, e portanto:

$$\ddot{\sigma}_{(S_1 \text{ } k-NN(d_1,k_1) \text{ } s_{q1})} (\ddot{\sigma}_{(S_2 \text{ } k-NN(d_2,k_2) \text{ } s_{q2})} T) \not\equiv \ddot{\sigma}_{(S_2 \text{ } k-NN(d_2,k_2) \text{ } s_{q2})} (\ddot{\sigma}_{(S_1 \text{ } k-NN(d_1,k_1) \text{ } s_{q1})} T) \quad (4.2)$$

Assim, dado que uma seleção aos k -vizinhos mais próximos não possui as mesmas propriedades que o operador de seleção, uma variante deve ser acrescentada como um operador derivado de maneira a representá-lo adequadamente, além de ter suas propriedades identificadas.

Além disso, a variação do k -vizinhos mais próximo também pode levar à ambiguidade na interpretação de consultas, pois obter “os k valores de atributos mais próximos a s_q e selecionar todas as tuplas que possuam quaisquer desses elementos como valores do atributo S ” gera um conjunto diferente do conjunto composto por “todas as k tuplas cujos valores do atributo S estão entre os mais próximos a s_q ”. De fato, esse é o significado da não equivalência expressa em (4.2). Portanto, é necessário que o operador de comparação por k -vizinhos mais próximos gere duas variantes da seleção: ele pode retornar todas as tuplas $t \in T$ de maneira que o atributo $t[S]$ é um

dos k mais próximos a s_q ; ou ele pode retornar as k tuplas $t \in T$ de maneira que o valor $t[S]$ é um dos mais próximos a s_q .

Como os dois operadores são derivados das propriedades algébricas fundamentais, eles não alteram o poder expressivo da álgebra, mas removem as ambiguidades na interpretação de consultas, de maneira a facilitar a expressão das mesmas. Além disso, eles possuem as mesmas propriedades algébricas já estudadas na literatura para operadores de seleção baseados em comparação por $\ddot{o}$. Vale notar que se o atributo S é chave em T , ambas as interpretações serão equivalentes.

4.3 Embutindo o Operador de Comparação por k -NN no Modelo Relacional

Com a forma de se integrar similaridade em um SGBDR bem definida, pode-se agora tratar a duas interpretações distintas do operador de comparação por k -NN. Considerando que uma seleção obtém um conjunto-resultado $T_R \subset T$. Seja $n = |T_R|$ a cardinalidade do resultado, e $m = |\pi_{\{S\}} T_R|$ a cardinalidade do domínio ativo de S em T_R , isto é, a quantidade de valores distintos do atributo complexo S que ocorrem em T_R .

A primeira interpretação do operador de seleção por k -NN obtém um conjunto-resultado composto das tuplas que têm o atributo complexo S com um dos k valores mais próximos ao centro de consulta s_q . Sob essa interpretação, não é a quantidade de tuplas obtidas que é pré-definida, mas sim a quantidade de valores distintos do atributo complexo. Portanto, o número de tuplas retornadas n é igual ou maior do que k (se ao menos k tuplas existirem em T) e o número de valores distintos do atributo S no resultado é $m = k$, desconsiderando a possibilidade de empates, que podem ser tratados posteriormente. Essa interpretação, que limita a quantidade de valores do atributo complexo no conjunto resultado é denominada **contagem de valores**. Chamamos aqui o operador de seleção correspondente de “ **k -seleção**”, e o denotamos como $\boxed{\ddot{o}_{(S \text{ } k\text{-}NN(d,\xi) \text{ } s_q)} T}$.

A segunda interpretação obtém um conjunto-resultado T_R composto das k tuplas cujo valor do atributo complexo S está entre os mais próximos do centro de consulta s_q . Sob essa interpretação, a quantidade de tuplas obtidas é $n = k$, mas a quantidade m de valores distintos do atributo complexo obtida pode ser qualquer $0 \leq m \leq k$, inclusive zero. Essa interpretação, que limita a quantidade de tuplas no conjunto-resultado é denominada **contagem de tuplas**. Chamamos aqui o operador de seleção correspondente de “ **kt -seleção**” e o denotamos como $\boxed{\ddot{o}_{(S \text{ } k\text{-}NN(d,\xi) \text{ } s_q)} T}$.

Existindo duas interpretações, cada operador de seleção por similaridade baseada em k -NN pode resultar em duas consultas diferentes, e os dois conjuntos-resultado podem ter cardinalidades distintas. A Tabela 4.1 mostra as representações algébricas das consultas Q_4 e Q_5 do exemplo dado na Seção 4.1, com suas cardinalidades correspondentes, tanto em número de tuplas (n)

Consulta	Variação	Representação Algébrica	n	m
Q4 - Quais são os itens vendidos que tem sua imagem dentre as 5 mais similares a imagem s_q ?	Contagem de valores	$\ddot{\sigma}_{(\text{ImgPintura } k\text{-}NN(d,5) s_q)} \text{ItemVendido}$	≥ 5	5
Q5 - Quais são os 5 itens vendidos cuja imagem está entre as mais similares a imagem s_q ?	Contagem de tuplas	$\dot{\sigma}_{(\text{ImgPintura } k\text{-}NN(d,5) s_q)} \text{ItemVendido}$	5	≤ 5

Tabela 4.1: Representação das duas consultas derivadas das duas interpretações possíveis do operador $k\text{-}NN$.

quanto em número de valores de atributos complexos retornados (m).

Levando em consideração a extensão da linguagem SQL tratada pelo SIREN, como apresentado no Capítulo 3, a distinção entre ambas as interpretações pode ser sumarizada acrescentando apenas uma palavra-chave opcional `[VALUES | TUPLES]` na cláusula `AFTER  $k$` . Assim, $Q4$ e $Q5$ podem ser expressas como:

```
SELECT Item
FROM ItemVendido
WHERE ImgPintura NEAR  $s_q$  STOP AFTER 5 [ VALUES | TUPLES ]
```

onde a palavra-chave opcional `VALUES` indica a interpretação de “*contagem de valores*” (i.e. $Q4$, referente a valores distintos de atributos complexos), enquanto a palavra-chave `TUPLES` indica a opção pela “*contagem de tuplas*” (i.e. $Q5$). Definimos aqui que se a cláusula for omitida a opção padrão é contagem de valores.

O operador $k\text{-}NN$ retorna **verdadeiro** para um predicado $(t_i[S] \text{ } k\text{-}NN(d, k) s_q)$ sobre a tuplas $t_i \in \mathbb{T}$ se $t_i[S]$ é um dos k elementos no espaço da busca mais próximos do centro de consulta. Portanto, o espaço de busca tanto de $\ddot{\sigma}_{(S \text{ } k\text{-}NN(d, k) s_q)} \mathbb{T}$ quanto de $\dot{\sigma}_{(S \text{ } k\text{-}NN(d, k) s_q)} \mathbb{T}$ é o domínio ativo S do atributo S na relação $\mathbb{T}$. No entanto, se qualquer um dos dois operadores de seleção baseados no $k\text{-}NN$ são aplicados sobre o resultado $\mathbb{T}_{Temp}$ de uma seleção prévia (independente do tipo de seleção), o espaço de busca será $\pi_{\{\mathbb{T}_{Temp}\}}$. Portanto, nenhum dos operadores é comutativo, disjuntivo ou associativo. De fato, nenhuma das consultas

$$\ddot{\sigma}_{(k\text{-}NN\text{Pred})} \mathbb{T} \bigcap \sigma_{(\text{qquerPred})} \mathbb{T}, \quad (4.3)$$

$$\ddot{\sigma}_{(k\text{-}NN\text{Pred})} \left(\sigma_{(\text{qquerPred})} \mathbb{T} \right), \quad (4.4)$$

$$\sigma_{(\text{qquerPred})} \left(\ddot{\sigma}_{(k\text{-}NN\text{Pred})} \mathbb{T} \right) \quad (4.5)$$

são equivalentes entre si, onde $\ddot{\sigma}$ é um operador de seleção por $k\text{-}NN$ seguindo qualquer das interpretações $\ddot{\sigma}$ ou $\dot{\sigma}$.

Propomos aqui que um comando SQL que tenha uma construção “WHERE $\langle \text{QualquerPred} \rangle$ AND $\langle k\text{-}NN\text{Pred} \rangle$ ” seja traduzido para uma expressão algébrica da forma expressa em (4.3). A razão para isso é que o operador de intersecção de conjuntos continua válido e portanto

$$\ddot{\sigma}_{(k\text{-}NN\text{Pred})} \top \bigcap \sigma_{(\text{qualquerPred})} \top \Leftrightarrow \sigma_{(\text{qualquerPred})} \top \bigcap \ddot{\sigma}_{(k\text{-}NN\text{Pred})} \top. \quad (4.6)$$

Assim, traduzindo o comando para a expressão (4.3) permite que a expressão de conjunções de predicados num comando SQL continue a poder ser feita sem que a ordem precise ser levada em conta pelo programador.

Portanto, todas as seleções não baseadas em similaridades podem ser ordenadas da maneira tradicional, e o resultado é a intersecção com o resultado de cada seleção baseada em similaridade. Se a intenção for obter uma sequencia de seleções em uma ordem específica, como por exemplo

$\sigma_{(\text{qualquerPred})} \left(\ddot{\sigma}_{(k\text{-}NN\text{Pred})} \top \right)$, então tal ordenação deverá ser explicitada na consulta usando a cláusula **FROM** para gerar o espaço de busca $T_{Temp} = \pi_{\left\{ \ddot{\sigma}_{(k\text{-}NN\text{Pred})} \right\}} \top$. Desta maneira, todas

as propriedades da teoria dos conjuntos se mantém e podem ser empregadas para interpretar, otimizar e executar as consultas, desde que os respectivos operadores de k -seleção e kt -seleção sejam implementados no SGBDR alvo.

Os Algoritmos 1 e 2 descrevem uma implementação básica de cada interpretação, para executar as seleções baseadas em $k\text{-}NN$ por contagem de tuplas ou contagem de valores, respectivamente. Eles mostram que o respectivo operador de seleção deve ser implementado utilizando o operador de comparação por $k\text{-}NN$, indicado como uma chamada ao procedimento “i-th-NearestNeighbor” para escolher entre as tuplas a serem recuperadas, que indica uma chamada ao procedimento “getTuples”. Note que os procedimentos “i-th-NearestNeighbor” e “getTuples”, assim como os que implementam os métodos de acesso que executam o operador de seleção na base de dados já estão disponíveis na implementação do SIREN que não contempla as múltiplas interpretações. Portanto, cada método de acesso capaz de executar operadores de seleção por similaridade deve ser modificado para implementar os algoritmos apresentados, incluindo *table scan*, *index scan*, *multi-index scan* e *index-only scan*.

Algorithm 1: k -NN por Contagem de Tuplas

Data: k, s_q
Result: resultSet
initialization;
begin
 for $i = 1:k$ **do**
 complexAtr :=
 i-th-NearestNeighbor(Sq);
 tuples := getTuples(
 complexAtr);
 resultSet += tuples;
 if $resultSet.size() \geq k$ **then**
 break;

Algorithm 2: k -NN por Contagem de Valores

Data: k, s_q
Result: resultSet
initialization;
begin
 for $i = 1:k$ **do**
 complexAtr[i] :=
 i-th-NearestNeighbor(Sq);
 tuples := getTuples(complexAtr
);
 resultSet := tuples;

Estes algoritmos foram empregados na realização dos experimentos mostrados no Capítulo 6, e para obter os resultados da quarta e quinta coluna mostradas na Figura 4.1.

4.4 Considerações Finais

Neste capítulo foram apresentadas duas novas maneiras de interpretar o operador k -NN integrado a um SGBD relacional que comporta consultas por similaridade. As novas interpretações se tornam necessárias quando os dados complexos passam a ser armazenados como valores de atributos em relações. Seguindo a maneira tradicional como os dados são acessados pelas técnicas de busca tratadas na literatura, considera-se que os dados complexos ficam armazenados como um conjunto de elementos, e portanto os dados não se repetem. Quando os dados complexos são armazenados como valores de atributos numa relação, cada tupla pode ter outros atributos, e nada obriga a que o atributo complexo seja chave na relação. Assim, é possível que a coleção de tuplas armazenadas na relação tenha atributos complexos com valores repetidos. Quando isso ocorre, surge a necessidade de distinguir entre as duas interpretações apresentadas neste capítulo.

A primeira interpretação é a que se dá tradicionalmente quando se faz uma busca por similaridade utilizando o operador k -NN, onde cada valor ocorre apenas uma vez ou, contextualizado para o ambiente relacional, em apenas uma tupla – ou seja, quando o atributo complexo também é chave na relação. Essa interpretação, chamada *contagem de valores*, é derivada do paradigma dos espaços métricos, que impõe que repetições não podem ocorrer. No entanto, quando se integra consultas por similaridade a um SGBDR é preciso considerar que, segundo o modelo relacional, a tupla inteira é que não pode ocorrer repetida (e nem isso é garantido pelas implementações dos SGBDR em aplicações reais). Portanto, múltiplas tuplas podem guardar valores repetidos de

atributos complexos quando a relação é composta por mais atributos do que apenas um complexo. Esse fato torna a interpretação da *contagem de valores* insuficiente, criando a necessidade de uma nova interpretação: a da *contagem de tuplas*. Ambas as interpretações foram definidas e detalhadas neste capítulo, sendo proposto que cada interpretação leva a um novo operador algébrico de seleção por similaridade: a *k*-seleção e a *kt*-seleção.

Também foram apresentadas a forma como as duas interpretações são integradas à linguagem SQL estendida para permitir expressar consultas por similaridade, como essas consultas podem ser mapeadas para expressões algébricas, e como elas devem interagir com os demais operadores de seleção por similaridade e tradicionais. Por fim, foram introduzidos dois algoritmos que permitem implementar ambos os operadores algébricos em métodos de acesso físicos.

CAPÍTULO 5

DESEMPATANDO CONSULTAS k - NNQ EM SGBDS RELACIONAIS

5.1 Introdução

Conforme foi colocado no início do Capítulo 4, este projeto de mestrado cuida de dois problemas associados ao uso do operador de k - NN em um SGBDR, sendo o primeiro problema tratado naquele capítulo. O segundo problema corresponde à possibilidade de empates que podem correr na escolha do k -ésimo elemento a ser colocado no resultado de uma busca por k - NN . Mais especificamente, quando uma consulta k - NNq é realizada sobre um conjunto de dados complexos utilizando apenas a distância (ou similaridade) entre os elementos, é impossível definir um critério que resolva os empates baseado apenas nos dados. Tradicionalmente, o problema do desempate é resolvido por uma de duas maneiras:

- Usam-se informações externas codificadas no algoritmo de busca, como por exemplo se escolhe o elemento pela ordem em que estão fisicamente armazenados; ou
- Retornam-se todos os elementos empatados, aceitando mais do que k elementos na resposta.

Ambas as possibilidades são problemáticas quando os dados estão num SGBDR.

Usar informações externas pode levar a consultas não repetíveis. Fora do contexto de bases de dados, é comumente aceita a escolha de $k' \leq k$ elementos dentre os empatados quando utilizando o operador de consulta k - NNq , pois em ambientes com um único usuário e com o mesmo conjunto de dados a consulta sempre resultará no mesmo resultado. No entanto, essa suposição não é verdadeira no ambiente multiusuário de um SGBD, onde transações concorrentes podem mudar as estruturas internas que armazenam os dados, mesmo que os valores não sejam alterados. Por outro lado, retornar todos os elementos empatados pode prejudicar a seletividade de consultas que de outra maneira teriam uma cardinalidade bem definida, piorando o desempenho

do gerenciador.

Este capítulo trata do problema de empates, focando na seguinte observação:

O tratamento de empates em consultas por k -NN sobre dados complexos armazenados como atributos de uma relação pode levar em conta os outros atributos da relação para o processo de desempate. Ou seja, em uma dada relação T composta pelos atributos $\{A_1, \dots, A_d\}$, onde o número de atributos $d > 1$, consultas envolvendo o operador de comparação por k -NN aplicado sobre um atributo $A_i \in T$ podem empregar outros atributos $A_j \in T \setminus \{A_i\}$ para ajudar no processo de desempate quando há múltiplos elementos empatados no k -ésimo vizinho em uma consulta k -NNq sobre um atributo A_i .

Vale observar que não é possível garantir que usando outros atributos sempre serão resolvidos todos os empates. Mesmo que todos os atributos A_j em T forem utilizados no processo, ainda é factível que, dependendo dos dados armazenados em T , alguns empates permaneçam. Assim, o objetivo não é necessariamente eliminar todos os empates, mas sim prover um método determinístico e mais significativo para o processo, provendo o ferramental para que analistas de dados possam tratar melhor o problema de maneira mais granular, mesmo que alguns empates possam não ser resolvidos.

Portanto, neste capítulo é introduzido um novo ferramental conceitual para tratar empates originados pelo operador de seleção por k -NN dentro de SGBDRs, se aproveitando da integração e do ambiente relacional para prover contexto e suporte ao processo. Apesar dos conceitos abordados neste capítulo serem igualmente aplicáveis para empates ocorrendo em qualquer elemento que seja retornado, há aqui um interesse maior em empates envolvendo o k -ésimo elemento mais próximo, pois é ele que tem potencial para alterar a cardinalidade do resultado, ou empates que ocorram antes do k -ésimo elemento, mas nos quais o número de elementos empatados seja grande o suficiente de maneira que o k -ésimo elemento pertença ao conjunto de elementos empatados. Para simplificar, ambos os casos serão referidos como empates no k -ésimo elemento.

Assumimos os mesmos operadores algébricos e notações utilizadas no Capítulo 4. O restante do capítulo é composto pela Seção 5.2 que apresenta um novo operador de seleção por similaridade para k -NN que permite desempates, pela Seção 5.3 que apresenta um algoritmo para executar o novo operador de desempate, e pela Seção ?? que conclui o capítulo.

5.2 O Operador de Seleção por Similaridade com Desempate

Assumindo que seja solicitada a consulta $\boxed{\ddot{\sigma}_{(S \text{ } k\text{-}NN(d,k) \text{ } s_q)} \text{ } T}$, onde $\ddot{\sigma}$ é o operador de seleção por $k\text{-}NN$ seguindo qualquer das interpretações por valor ou por tupla, e s_q é o centro da consulta com o qual o valor do atributo S em cada tupla é comparado. Considere-se que existam empates nas tuplas que podem ocupar a k -ésima posição da lista de tuplas retornadas, ordenados pela distância de cada um ao centro de consulta. Seja T_{R_t} o conjunto de tuplas da relação T cujos valores do atributo S estão empatados nessa distância e $k' < |T_{R_t}|$ o número de tuplas que precisa ser selecionado de T_{R_t} para completar k elementos na resposta. Ou seja, a resposta conterá $k - k'$ tuplas em que os valores de S estão a distâncias menores em relação ao centro de consulta, mas $(k - k') + |T_{R_t}| > k$.

Note-se que podem existir empates em tuplas que na resposta ordenada ocorrem antes da k -ésima posição, mas como já mencionado, estas não são de interesse para este trabalho. Note-se também que os empates podem ocorrer em qualquer das interpretações, e ambos são tratados da mesma maneira no desenvolvimento feito a seguir. Assim, qualquer das interpretações pode ser usada para obter um conjunto T_{R_t} de tuplas empatadas na k -ésima posição, e essas são as tuplas que devem passar pelo processo de desempate.

Vamos assumir que exista outro atributo A (que pode ser complexo ou não) além de S na relação T . Então, é possível definir um critério de desempate das tuplas que farão parte da resposta da consulta usando esse atributo para prover uma ordem parcial sobre as tuplas em T_{R_t} . Assim, definimos aqui um novo operador de “**Seleção por Similaridade com Desempate**”, que é expresso usando a seguinte notação:

$$\boxed{\ddot{\sigma}_{(S \text{ } k\text{-}NN(d,k) \text{ } s_q) \{ \langle T \rangle \}} T},$$

onde

- $\ddot{\sigma}$ é um operador de seleção por similaridade $k\text{-}NN$ seguindo qualquer das interpretações por valor ($\ddot{\sigma}$) ou por tupla ($\dot{\sigma}$),
- T é uma relação contendo pelo menos um atributo complexo S e um outro atributo A complexo ou não;
- $S \text{ } k\text{-}NN(d,k) \text{ } s_q$ é o predicado de comparação por similaridade tal como definido anteriormente para esse operador de seleção, e
- $\langle T \rangle$ é uma lista de termos em que cada termo é uma expressão de comparação da forma $\boxed{\mathcal{T} = A \theta \text{ value}}$, onde θ é um operador de comparação válido no domínio de A e value pode ser:
 - ‘cte’: um valor constante do mesmo domínio de A ; ou

- #: o símbolo ‘#’ representando o valor do atributo A em qualquer tupla $t_j \in T_{R_t}$.

Nesse operador, cada expressão $\mathcal{T}$ pode prover uma ordem parcial sobre as tuplas em T_{R_t} que empatam na consulta $\ddot{\sigma}$, e portanto auxiliar a desempatar as tuplas que farão parte da resposta da consulta k -NN. Os termos são avaliados em sequência na ordem da lista indicada, até que exista um número adequado de tuplas no resultado, ou até que se esgotem todos os termos da sequência.

O número adequado de tuplas desempatadas no resultado depende da interpretação que esteja sendo usada para o operador de seleção. Se a interpretação é por tupla, então haverá um número adequado de tuplas no resultado sempre que houver exatamente k tuplas no resultado. Se a interpretação é por valor, então haverá um número adequado de tuplas no resultado sempre que houver tuplas suficientes no resultado para que o domínio ativo S do atributo S no resultado tenha a cardinalidade k .

É importante destacar que a notação proposta usa a mesma notação já definida antes para o operador de Seleção por Similaridade, acrescentando a lista $\{\langle \mathcal{T} \rangle\}$ de termos para desempate. Assim, caso a lista seja nula, o operador de Seleção por Similaridade com Desempate cai no caso trivial do operador de seleção por similaridade já definido anteriormente, e pode-se considerar que o único operador necessário é aquele aqui definido.

Cada termo de desempate pode usar qualquer operador de comparação definido no domínio do respectivo atributo. O atributo pode ser comparado com um valor constante ou com os valores obtidos no domínio ativo desse atributo no conjunto de tuplas T_{R_t} . Sempre que um termo é avaliado, T_{R_t} é particionado de acordo com o resultado da comparação, como definido a seguir.

- Comparação por valor constante: $\boxed{A \theta c}$

As tuplas em T_{R_t} são divididas em dois subconjuntos: aqueles que atendem ao critério $A \theta c$ e aqueles que não atendem. Se a cardinalidade do subconjunto de tuplas que atendem ao critério é maior ou igual a k' , então esse subconjunto se torna T_{R_t} , isto é, eliminam-se das tuplas empatadas aquelas que não atendem ao critério do termo avaliado. Caso contrário, o termo não filtra nenhuma tupla.

- Comparação pelo domínio ativo de A no conjunto T_{R_t} : $\boxed{A \theta \#}$

As tuplas em T_{R_t} são divididas em k_A subconjuntos, onde k_A é a cardinalidade do domínio ativo A do atributo A no conjunto de tuplas T_{R_t} . Para cada valor $a_i \in A$, obtêm-se as tuplas que satisfazem $A \theta a_i$. Essas tuplas são repassadas para o resultado do processamento desse termo e removidas do conjunto T_{R_t} original. O processamento prossegue então até que todos os elementos em A sejam processados ou até que o resultado do processamento desse termo atinja cardinalidade maior ou igual a k' .

Alguns casos de maior interesse são ilustrados a seguir.

Atributo comparado por igualdade com valor constante – $\boxed{A \theta c}$ com $\theta \in \{=, \neq\}$

Aqui, o atributo A deve ser comparável por identidade (é um atributo escalar E). Assuma-se por exemplo que $\theta = '='$. Caso exista uma quantidade maior ou igual a k' de tuplas com o atributo A igual à constante c , todas as tuplas com valor diferente são eliminadas da lista de empate. Se não houver tuplas suficiente, o termo não é usado para eliminar empates.

Atributo comparado por relação de ordem com valor constante – $\boxed{A \theta c}$ com $\theta \in \{<, \leq, >, \geq\}$

Esse caso é equivalente ao anterior, e o atributo A deve ser comparável por relação de ordem (é um atributo escalar E). Por exemplo, assumamos que $\theta = '<'$. Então, caso exista uma quantidade maior ou igual a k' de tuplas em que o atributo A é menor do que constante c , então todas as tuplas com valor maior ou igual são eliminadas da lista de empate. Se não houver tuplas suficiente, o termo não é usado para eliminar empates.

Atributo comparado por similaridade com valor constante – $\boxed{A \theta c}$ com $\theta \in \{Rng(d, \xi), k-NN(d, k)\}$

Aqui, o atributo A deve ser comparável por similaridade (é um atributo complexo S). Assuma-se que $\theta = 'Rng(d, \xi)'$. Então, caso exista uma quantidade maior ou igual a k' de tuplas em que o atributo A está à no máximo a distância ξ de c medida pela métrica d , então todas as tuplas com valor de A com maior distância a c são eliminadas da lista de empate.

Assuma-se agora que $\theta = 'k-NN(d, k)''$. Então, caso exista uma quantidade maior ou igual a k' de tuplas em que o atributo A é um dos k'' vizinhos mais próximos a c medida pela métrica d , então todas as demais tuplas são eliminadas da lista de empate. Em ambos os casos, se não houver tuplas suficiente, o termo não é usado para eliminar empates.

Atributo comparado por identidade com domínio ativo – $\boxed{A \theta \#}$ com $\theta \in \{=, \neq\}$

Aqui vale lembrar que o valor do atributo A é comparado com o valor desse atributo nas demais tuplas de T_{R_t} , mas não com a própria tupla. Então, assumamos por exemplo que $\theta = '='$. Neste caso, uma tupla será selecionada apenas se existir uma quantidade maior ou igual a k' de tuplas em que o atributo A tem o mesmo valor. Nesse caso, todas as tuplas são mantidas na lista de empate, e as demais são eliminadas. Se não houver tuplas suficiente, o termo não é usado para eliminar empates.

Atributo comparado por relação de ordem com domínio ativo – $\boxed{A \theta \#}$ com $\theta \in \{<, \leq, >, \geq\}$

Por exemplo, assumamos que $\theta = '<'$. Procura-se uma tupla em que o atributo A é menor

do que o valor desse atributo em qualquer outra tupla em T_{R_t} . Se essa tupla existir ela é passada para o resultado, removida de T_{R_t} , e o processo é repetido até que nenhuma outra tupla atenda a essa condição, ou até que k' tuplas sejam repassadas ao resultado.

Assuma-se agora que $\theta = '\leq'$. Procura-se uma ou mais tuplas em que o atributo A é menor ou igual ao valor desse atributo em qualquer outra tupla em T_{R_t} . Se ao menos uma tupla existir, elas são passada para o resultado, removidas de T_{R_t} , e o processo é repetido até que nenhuma outra tupla atenda a essa condição, ou até que pelo menos k' tuplas sejam repassadas ao resultado.

Atributo comparado por similaridade com domínio ativo – $\boxed{A \theta \#}$ com $\theta \in \{Rng(d, \xi), k-NN(d, k)\}$

Assuma-se que $\theta = 'Rng(d, \xi)'$. Então, caso exista uma quantidade maior ou igual a k' de tuplas em que o atributo A está à no máximo a distância ξ do que o valor desse atributo em qualquer outra tupla em T_{R_t} medida pela métrica d , então todas essas tuplas permanecem na lista de empate, e as demais são descartadas.

Assuma-se agora que $\theta = 'k-NN(d, k)''$. Então, caso exista uma quantidade maior ou igual a k' de tuplas em que o atributo A é um dos k'' vizinhos mais próximos ao valor desse atributo em qualquer outra tupla em T_{R_t} medida pela métrica d , então todas as demais tuplas são eliminadas da lista de empate. Em ambos os casos, se não houver tuplas suficiente, o termo não é usado para eliminar empates.

Note-se que quando um termo compara um atributo com uma constante na forma $A \theta c$, os operadores negados são equivalentes a seus opostos. Por exemplo, $'\neg ='$ é equivalente a $'\neq'$ e $'\neg <'$ é equivalente a $'\geq'$. No entanto, quando um termo compara um atributo com os valores no domínio ativo na forma $A \theta \#$, então essa equivalência não se mantém. A Tabela 5.1 mostra o significado dos diversos outros predicados que podem ser usados e ressalta essa diferença.

5.3 Um Algoritmo para o operador de Seleção por Similaridade com Desempate

O Algoritmo 3 seleciona os k' elementos que atendem à lista de expressões $L = \langle \mathcal{T} \rangle$ especificada para uma operação de seleção por similaridade com desempate. Ele extrai dois subconjuntos mutuamente exclusivos de T_{R_t} : T_u possuindo $|T_u| \leq k'$ são as tuplas que podem ser desempatadas seguindo as expressões listadas na lista L ; T_{temp} são as tuplas que não podem ser desempatadas seguindo as expressões listadas em L . Veja que $|T_u| + |T_{temp}| \leq |T_{R_t}|$. Quando $|T_u| = k'$ então T_{temp} estará vazia, e isso significa que a lista de predicados L foi capaz de extrair a resposta necessária sem empates. Se T_{temp} não estiver vazia, então essas são as tuplas que não podem ser desempatadas pelos predicados na lista de expressões L .

Tabela 5.1: Significado dos símbolos usados nas expressões de predicados.

θ	$\mathcal{T} = A \theta \text{ 'cte'}$	$\mathcal{T} = A \theta \#$
$=$	Recupera cada tupla $t_i \in \mathcal{T}_{R_t}$ de maneira que $t_i(A) = \text{'cte'}$	Recupera cada tupla $t_i \in \mathcal{T}_{R_t}$ de maneira que para todas as outras tuplas $t_j \in \mathcal{T}_{R_t}, j \neq i$, ocorra que $t_i(A) = t_j(A)$
		Quando toda a tupla $t_j \in \mathcal{T}_{R_t}, j \neq i$ tenha o atributo A com o mesmo valor que $t_i(A)$, retorna a tupla t_i
$\neq$	Recupera cada tupla $t_i \in \mathcal{T}_{R_t}$ de maneira que $t_i(A) \neq \text{'cte'}$	Recupera cada tupla $t_i \in \mathcal{T}_{R_t}$ de maneira que para nenhuma tupla $t_j \in \mathcal{T}_{R_t}, j \neq i$, ocorra que $t_i(A) = t_j(A)$
$\neg =$	Igual a $\neq$	Recupera cada tupla $t_i \in \mathcal{T}_{R_t}$ de maneira que exista a tupla $t_j \in \mathcal{T}_{R_t}, j \neq i$ de maneira que $t_i(A) \neq t_j(A)$
$\neg \neq$	Igual a $=$	Recupera cada tupla $t_i \in \mathcal{T}_{R_t}$ de maneira que exista a tupla $t_j \in \mathcal{T}_{R_t}, j \neq i$ de maneira que $t_i(A) = t_j(A)$
$< \text{ ou } >$	Recupera cada tupla $t_i \in \mathcal{T}_{R_t}$ de maneira que $t_i(A)$ é menor/maior que 'cte'	Recupera a tupla $t_i \in \mathcal{T}_{R_t}$ de maneira que $t_i(A)$ é o menor/maior valor do atributo A em $\mathcal{T}$ (se existir e for único)
$\neg < \text{ ou } \neg >$	Igual a $\geq / \leq$	Recupera a tupla $t_i \in \mathcal{T}_{R_t}$ de maneira que $t_i(A)$ não é o menor/maior valor do atributo A em $\mathcal{T}$
...		
$Rng(d, \xi)$	Recupera cada tupla $t_i \in \mathcal{T}_{R_t}$ de maneira que $t_i(A)$ está dentro de ξ de 'cte'	Recupera cada tupla $t_i \in \mathcal{T}_{R_t}$ de maneira que $t_i(A)$ está dentro de ξ dos valores $t_j(A)$ para todas as outras tuplas $t_j \in \mathcal{T}_{R_t}, j \neq i$
$\overset{>}{Rng}(d, \xi)$	Recupera cada tupla $t_i \in \mathcal{T}_{R_t}$ de maneira que $t_i(A)$ está mais longe que ξ de 'cte'	Recupera cada tupla $t_i \in \mathcal{T}_{R_t}$ de maneira que $t_i(A)$ está mais longe que ξ dos valores $t_j(A)$ para todas as outras tuplas $t_j \in \mathcal{T}_{R_t}, j \neq i$
$\neg Rng(d, \xi)$	Igual a $\overset{>}{Rng}(d, \xi)$	Recupera cada tupla $t_i \in \mathcal{T}_{R_t}$ de maneira que $t_i(A)$ está mais longe que ξ dos valores $t_j(A)$ de pelo menos uma tupla $t_j \in \mathcal{T}_{R_t}, j \neq i$.
$\neg \overset{>}{Rng}(d, \xi)$	Igual a $\neg Rng(d, \xi)$	Recupera cada tupla $t_i \in \mathcal{T}_{R_t}$ de maneira que $t_i(A)$ está dentro de ξ do valor $t_j(A)$ de pelo menos uma tupla $t_j \in \mathcal{T}_{R_t}, j \neq i$.
$k\text{-}NN(d, k)$	Recupera cada tupla $t_i \in \mathcal{T}_{R_t}$ de maneira que $t_i(A)$ é um dos k -vizinhos mais próximos de 'cte'	Recupera cada tupla $t_i \in \mathcal{T}_{R_t}$ de maneira que $t_i(A)$ é um dos k -vizinhos mais próximos dos valores $t_j(A)$ para todas as outras tuplas $t_j \in \mathcal{T}_{R_t}, j \neq i$
$\neg k\text{-}NN(d, k)$	Recupera cada tupla $t_i \in \mathcal{T}_{R_t}$ de maneira que $t_i(A)$ não é um dos k -vizinhos mais próximos de 'cte'	Recupera cada tupla $t_i \in \mathcal{T}_{R_t}$ de maneira que $t_i(A)$ não é um dos k -vizinhos mais próximos dos valores $t_j(A)$ para todas as outras tuplas $t_j \in \mathcal{T}_{R_t}, j \neq i$
...		

Algorithm 3: Algoritmo de desempate de tuplas.

Data:

T_{R_t} : subconjunto de tuplas de $T = (A_1, \dots, A_N)$ a serem desempatadas;

k' : número de tuplas necessárias, $k' \leq |T_{R_t}|$;

L : lista priorizada de termos $\mathcal{T}$ sobre os atributos $A_i \in T$

Result:

$T_u \subseteq T_{R_t}$: tuplas desempatadas;

$T_{temp} \subseteq T_{R_t}$: tuplas que não podem ser desempatadas por L ;

de forma que $T_u \cup T_{temp} = T_{R_t}$ e $|T_u| \leq k'$ atenda a L .

begin

set T_u null;

set $k' = \min(k', |T_{R_t}|)$;

while L not null **do**

$\mathcal{T} := \text{Get}(L)$;

 set T_{temp} null;

$n = |T_{R_t}|$;

for $i = 1:n$ **do**

$\backslash\backslash$ bloco otimizável

if $\mathcal{T}$ não possui valores $\#$ **then**

if $\mathcal{T}$ é verdadeiro para $t_i[T_{R_t}]$ **then**

 mover t_i de T_{R_t} para T_{temp} ;

else

$\backslash\backslash$ comparar às outras tuplas

 set flag=verdadeiro;

for $t_j \in T_{R_t} \setminus \{t_i\}$ **do**

$\mathcal{T}' = \text{FillExpression}(\mathcal{T}, t_j)$;

if $\mathcal{T}'$ é verdadeiro para $t_i[T_{R_t}]$ **then**

 set flag=falso;

break;

if flag **then**

 mover t_i de T_{R_t} para T_{temp} ;

if $|T_{temp}| = 0 \vee |T_{temp}| = |T_{R_t}|$ **then**

 Pop(L);

continue;

if Enough(T_u, T_{temp}) **then**

 set $T_u := T_u \cup T_{temp}$;

$k' = k' - |T_{temp}|$;

if $k' = 0$ **then**

 set T_{temp} null;

break;

else

continue;

 set $T_{R_t} = T_{temp}$;

retornar T_u e T_{temp}

Se algum termo $\mathcal{T}$ da lista $\langle \mathcal{T} \rangle$ tiver o formato $\mathcal{T} = A \theta \#$, o Algoritmo 3 precisará substituir ' $\#$ ' com cada valor t_i correspondente no domínio ativo A do atributo A para cada tupla $t_j \in$

$T, j \neq i$. Essa operação é realizada pelo Algoritmo 3 na chamada da função “FillExpression”, apresentada como Algoritmo 4. Aplicada sobre a tupla t_j , a função “FillExpression” substitui $\#$ em cada termo pelo valor t_i correspondente do atributo A na tupla t_j .

A avaliação “Enough(T_u, T_{temp})” verifica se o termo $\mathcal{T}$ selecionou tuplas adequadas, que não ultrapassem a quantidade k' de tuplas necessárias para atender aos critérios de desempate, levando em conta se a seleção está sendo feita sob a interpretação por tupla ou por valor. Se a seleção estiver sendo feita por tupla, o desempate induzido por $\mathcal{T}$ pode ser aproveitado sempre que $|T_{temp}| \leq k'$. Se a seleção estiver sendo feita por tupla, o desempate pode ser aproveitado sempre que a cardinalidade do domínio ativo de A em $T_u \cup T_{temp}$ for menor ou igual a k' .

Algorithm 4: Preenche os valores dos atributos.

Data:

expressão $\mathcal{T}$ com comparações a valores de atributos;
tupla t_j

Result:

expressão $\mathcal{T}'$ com as comparações a atributos preenchidos com o valor correspondente da tupla t_j

begin

for cada termo $\mathcal{T} = A \theta value$ **em** L **do**

if *value é um atributo* A_c **then**

 set $value = t_j[A_c]$;

 retorna $\mathcal{T}$

Alguns exemplos práticos podem ajudar a ilustrar como o novo operador e algoritmo podem ser usados para desempatar tuplas numa consulta por k -NN. Para isso, suponha-se que existe a seguinte relação de plantas lenhosas, (árvores, arbustos e lianas), definida com o seguinte esquema:

PlantaLenhosa = { Gênero, Nome,
FolhaTipo, FolhaArranjo, FolhaForma, FolhaMargem, FolhaTamanho,
FlorTipo, FlorCor, FlorTamanho,
FormaTipo, FormaTamanho,

Essa relação pode ser criada como uma tabela na linguagem SQL estendida do SIREN pelo comando:

```
CREATE TABLE PlantaLenhosa AS (
  Genero VARCHAR(50) NOT NULL,
  Nome VARCHAR(50) NOT NULL,
  FolhaTipo VARCHAR(50),
  FolhaArranjo VARCHAR(50),
  FolhaForma VARCHAR(50),
  FolhaMargem VARCHAR(50),
  FolhaTamanho VARCHAR(50),
```

```

FlorTipo VARCHAR(50),
FlorCor VARCHAR(50),
FlorTamanho VARCHAR(50),
FormaTipo VARCHAR(50),
FormaTamanho VARCHAR(50),
Folha PARTICULATE,
Flor PARTICULATE,
Forma PARTICULATE,
PRIMARY KEY (Genero, Nome)
);

```

Cada atributo que descreve as folhas, flores e formas são atributos de tipo textual contendo descrições padronizadas. Cada uma das características **folha**, **flores** e **forma** definem um novo atributo (definido em SIREN como um atributo de tipo PARTICULATE) que permite a comparação por similaridade com aquela característica. Por exemplo, a avaliação da similaridade pelo atributo complexo **Folha** considera os atributos textuais **FolhaTipo**, **FolhaArranjo**, **FolhaForma**, **FolhaMargem** e **FolhaTamanho**. Assim, uma comparação pela característica **folha**, por exemplo usando a distância de Jaccard, compara os valores dos atributos textuais, e quanto mais semelhantes todos eles são, menor a distância nas comparações envolvendo o atributo **Folha**. Como os atributos que compõem cada característica têm como valor os nomes (textos) associados a ela, é de se esperar que podem existir muitos empates.

Sobre a relação considera-se as consultas indicadas a seguir. Seja inicialmente uma consulta por k -NN que não leva empates em consideração:

- “**Q6** - Quais são as plantas que têm suas folhas entre as cinco mais similares a folha da *Magnolia ashei*”?

Essa consulta pode ser expressa como:

$$\ddot{O}_{(\text{Folha } k\text{-NN}(\text{jaccard}, 5) \text{ Magnoliaashei})} \text{PlantaLenhosa},$$

onde *Magnoliaashei* corresponde aos valores dos atributos $\langle \text{FolhaTipo}, \text{FolhaArranjo}, \text{FolhaForma}, \text{FolhaMargem}, \text{FolhaTamanho} \rangle$, que podem ser obtidos da base por uma consulta tradicional à tupla cujos atributos **Gênero** e **Nome** são respectivamente '*Magnolia*' e '*ashei*'.

Essa consulta inicial não possui nenhum controle sobre processo de empate. Se for desejado controlar o processo, o usuário pode formular **Q7** e se quiser pode continuar adicionando predicados, como em **Q8**:

- “**Q7** - Quais são as plantas que têm suas folhas entre as k mais similares a folha s_q ? Em caso de empates, utilize as plantas com flor mais similar à da *Magnolia ashei* como critério de desempate.”

Essa consulta pode ser expressa como:

$\ddot{\sigma}_{(\text{Folha } k\text{-}NN(\text{jaccard},5) \text{ Magnoliaashei})\{\text{Flor } k\text{-}NN(\text{jaccard},1) \text{ Magnoliaashei}\} \text{ PlantaLenhosa}}.$

- “**Q8** - Quais são as plantas que têm suas folhas entre as k mais similares a folha s_q ? Em caso de empates, utilize as plantas com flor mais similar à da *Magnolia ashei* como critério de desempate. Se ainda existirem empates, desempate priorizando as que não forem arbustos.”

Essa consulta pode ser expressa como:

$\ddot{\sigma}_{(\text{Folha } k\text{-}NN(\text{jaccard},5) \text{ Magnoliaashei})\{\text{Flor } k\text{-}NN(\text{jaccard},1) \text{ Magnoliaashei}, \text{FormaTipo} \neq \text{arbusto},\} \text{ PlantaLenhosa}}.$

A Figura 5.1 mostra um exemplo de execução possível do Algoritmo 3 para as consultas Q6, Q7 e Q8 sobre a relação de plantas lenhosas.

5.4 Uma notação para tratar desempates em SQL

A extensão da linguagem SQL proposta para o SIREN original já considera uma notação para expressar seleções por similaridade baseada em k -NN. No Capítulo 4, essa notação foi ampliada para tratar também das duas possibilidades de interpretação, por tupla e por valor. Apresentamos aqui uma proposta final para expressar essas seleções, que leva em conta essas possibilidades e adiciona o tratamento de empates. Com isso, a sintaxe final para expressar um predicado de busca por k -vizinhos mais próximos em SQL estendido é definida numa cláusula **WHERE**, e de maneira geral:

```

SELECT ...
FROM T
WHERE ... S {NEAR | FAR}  $s_q$ 
      STOP AFTER  $k$  [{VALUES | TUPLES}]
      [UNTIE USING  $\mathcal{T}_1$  [UNTIE USING  $\mathcal{T}_2$ ]...]
      [WITH TIE LIST];

```

Note-se que os termos $\mathcal{T}_i$ podem ser quaisquer predicados que podem ser expressos na cláusula **WHERE**, incluindo outros predicados de comparação por similaridade k -NN e as palavras chave

Figura 5.1: Representação gráfica das consultas Q6, Q7 e Q8 seguindo os passo do Algoritmo 3

CAPÍTULO 6

EXPERIMENTOS REALIZADOS

6.1 Introdução

Este capítulo apresenta os resultados de experimentos realizados para avaliar a performance dos operadores apresentados no Capítulo 4 e do ferramental proposto para tratamentos de empates introduzido no Capítulo 5. Ambos foram implementados estendendo a ferramenta “SImlarity Retrieval ENgine” – SIREN – e usando a implementação da *slim-tree* disponível na biblioteca Arboretum [43] como método de acesso métrico, ambos apresentados no Capítulo 3.

Os experimentos da seção 6.2 se referem aos operadores apresentados no Capítulo 4, e foram realizados em um computador com processador Intel Core i7-2720QM 2.20GHz e 8GB de RAM, sobre o sistema operacional Ubuntu Linux 12.04. Já os experimentos da Seção 6.3 se referem ao ferramental para tratamento de empates apresentado no Capítulo 5, e utilizaram uma máquina virtual com processador de dois núcleos, 4 GB de RAM sobre o sistema operacional Ubuntu Linux 14.04, com seu hospedeiro tendo um processador Intel Core i5-4200M 2.50GHz.

6.2 Contagem de Tuplas e Contagem de Valores

Esta seção apresenta experimentos que avaliam os operadores de seleção por similaridade segundo as duas interpretações propostas no Capítulo 4: contagem de valores e contagem de tuplas. Para isso, dois conjuntos de dados foram utilizados:

- **Pinturas:** Um conjunto de dados representando uma galeria de arte, que permite avaliar os exemplos apresentados na seção 4, onde cópias de pinturas vendidas são armazenadas e a informação de cada venda é registrada em um SGBDR. Este conjunto de dados é baseado

em um conjunto de dados de pinturas do mundo real¹, contendo 1300 pinturas distintas. Os dados sobre as compras (atributos `DataVenda`, `Item` e `NomeCliente`) foram geradas aleatoriamente seguindo uma distribuição uniforme para a quantidade de vendas, variando de 1 a 10 para cada pintura original, resultando em um conjunto de dados com 6890 tuplas. O atributo complexo utilizado foi o atributo `ImgPintura` de tipo `StillImage`, que armazena as imagens das pinturas vendidas. De cada `ImgPintura` foram extraídas características utilizando descritores de textura de Haralick [46], e foi usada a função de distância Euclidiana (L_2) para as comparações entre elas, declarando:

```
CREATE METRIC Haralick USING LP2 FOR STILLIMAGE(HaralickEXT);
```

- **NSF**: Um conjunto de dados contendo 129.000 resumos e metadados (ano, instituição etc.) descrevendo os auxílios à pesquisa concedidos pela Fundação Nacional da Ciência norte americana (NSF – *National Science Foundation*) entre 1990 e 2003². Cada auxílio contém um atributo complexo `BOWords` (“bag of words”) de tipo `LargeText`, que corresponde a uma lista de palavras, cada palavra associada com o número de vezes que ela aparece no resumo de cada projeto. Para as comparações entre os elementos foi utilizada a função de distância de Jaccard para multiconjuntos (multiset Jaccard – `MSJaccard`) sobre esse atributo, declarando:

```
CREATE METRIC JaccardBOW USING MSJaccard FOR LARGETEXT(BagOfWordsExt);
```

As consultas **Q9** (seleção usando contagem de valores) e **Q10** (contagem de tuplas) descritas a seguir, foram executadas no conjunto **Pinturas**, variando os valores de k de 1 a 25 em incrementos de 5 em 5.

- “**Q9** (contagem de valores) - Quais são os itens vendidos que têm sua imagem dentre as k mais similares a imagem s_q ?” Essa consulta corresponde à execução da expressão

$$\ddot{\sigma}(\text{ImgPintura } k\text{-NN}(\text{Haralick}, k) s_q) \text{ItemVendido}.$$

- “**Q10** (contagem de tuplas) - Quais são os k itens vendidos cuja imagem está entre as mais similares a imagem s_q ?” Essa consulta corresponde à execução da expressão

$$\dot{\sigma}(\text{ImgPintura } k\text{-NN}(\text{Haralick}, k) s_q) \text{ItemVendido}.$$

As medidas foram obtidas a partir de 100 consultas para cada valor de k , variando os centros cada consulta s_q . Os centros foram selecionados aleatoriamente dos valores armazenados em `ItemVendido.ImgPintura`.

Em SQL estendido essas consultas podem ser expressas como:

¹<http://commons.wikimedia.org/wiki/Category:Paintings> – Acessado pela última vez: Apr 25, 2014

²<http://archive.ics.uci.edu/ml/datasets.html> – Acessado pela última vez: Apr 25, 2014

Q9:

```
SELECT *  
FROM ItemVendido  
WHERE ImgPintura  
NEAR  $s_q$  STOP AFTER k VALUES;
```

Q10:

```
SELECT *  
FROM ItemVendido  
WHERE ImgPintura  
NEAR  $s_q$  STOP AFTER k TUPLES;
```

O objetivo desse experimento foi medir a performance do algoritmo implementado para executar cada uma das duas interpretações e avaliar o resultado final da cardinalidade de tuplas e de valores de atributos complexos nesse conjunto de dados. Com essas finalidades, quatro medidas distintas foram coletadas:

- Tempo total de consulta para valores variados de k ;
- o número médio de acessos a disco e número médio de cálculos de distâncias;
- Número de tuplas recuperadas por cada consulta;
- Número de imagens distintas (valores do atributo complexo) recuperadas em cada consulta.

Na Figura 6.1 são apresentados os resultados para ambos os operadores aplicados sobre o conjunto de dados **Pinturas**, mostrando as quatro medidas coletadas. Como o conjunto de dados possui múltiplos valores repetidos para cada atributo complexo **ImgPintura**, é evidente a diferença entre ambos os operadores em se tratando do número de tuplas e atributos complexos distintos recuperados. Como consequência, executar a interpretação da contagem de valores é mais custoso, pois a busca continua por possíveis resultados mesmo depois da contagem de tuplas já haver parado, sendo em média 36% mais demorada, embora retornando um conjunto-resultado com 550% mais tuplas.

Por outro lado, as medidas de desempenho, mostradas pelas curvas de número de acessos a disco, número de cálculos de distância e tempo total, indicam um comportamento claramente sublinear em relação ao parâmetro k para a interpretação da contagem de tuplas, e também para a interpretação da contagem de valores a menos do tempo total, que apresenta um comportamento aproximadamente linear. Isso garante que o algoritmo utilizado pode ser usado em SGBDR para responder a ambas as interpretações da consulta k -NN.

Buscando prover uma perspectiva diferente, foram também realizados experimentos sobre o conjunto de dados **NSF**. Em contraste com a galeria de arte, o conjunto de dados **NSF** é maior em tamanho e não contém repetições, sendo puramente métrico. Consequentemente, ambas as interpretações do operador k -NN possuem o mesmo resultado final (valores de atributos complexos retornado e cardinalidade das tuplas). Assim, o propósito deste experimento é verificar se há uma diferença significativa de performance entre os operadores, sabendo-se que o conjunto resultado final será igual. Por outro lado, o uso de um conjunto de dados puramente métrico demonstra a ampla aplicabilidade dos conceitos e algoritmos desenvolvidos.

Figura 6.1: Medidas obtidas para as duas interpretações do operador k -NN realizadas sobre o conjunto de dados **Pinturas** variando k de 1 a 25. Cada medida é a média de 100 consultas utilizando centros variados. (a) Número de tuplas recuperadas. (b) Tempo total de uma consulta. (c) Número de valores complexos distintos (i.e. pinturas) recuperados. (d) Número médio de acessos a disco. (e) Número médio de cálculos de distância realizados.

A relação contendo o conjunto de dados NSF é composta de três atributos: um *ID* único, o ano de publicação e uma “*bag of words*” BOWords. Sobre esse conjunto foram realizadas as seguintes consultas:

- “**Q11** (contagem de valores) - Quais são os artigos cujas palavras-chave estão entre as k mais similares às do artigo s_q ?” Essa consulta corresponde à execução da expressão $\tilde{\sigma}_{(\text{BOWords } k\text{-NN}(\text{JaccardBOW}, k) s_q)}^{\text{NSF}}$.
- “**Q12** (contagem de tuplas) - Quais são os k artigos cujas palavras-chave estão entre as mais similares às do artigo s_q ?” Essa consulta corresponde à execução da expressão

$$\sigma_{(\text{BoWords } k\text{-}NN(\text{JaccardBoW}, k) s_q)} \text{NSF}.$$

Em SQL estendido essas consultas podem ser expressas como:

Q11:

```
SELECT *
FROM NSF
WHERE BoWords
      NEAR  $s_q$  STOP AFTER k VALUES;
```

Q12:

```
SELECT *
FROM NSF
WHERE BoWords
      NEAR  $s_q$  STOP AFTER k TUPLES;
```

Na figura 6.2 são apresentados os resultados da execução de ambos os operadores para o conjunto de dados **NSF**. Os resultados obtidos a partir da realização de 100 consultas com centros variados para cada valor de k . Foi medido o tempo total das 100 consultas, e como indicadores de performance foram calculadas as médias do número médio de acessos a disco e do número de cálculos de distâncias necessários. Os experimentos indicam que, na ausência de repetição nos valores do atributo complexo, as duas interpretações convergem, tanto em cardinalidade do conjunto resultado quanto em performance.

6.3 Algoritmo de Desempate

Nesta seção são apresentados os resultados experimentais referentes à implementação do Algoritmo 3 apresentado no Capítulo 5 para desempate de tuplas resultantes de uma consulta utilizando o operador $k\text{-}NN$ em um ambiente relacional. Neste experimento, o seguinte conjunto de dados foi utilizado:

- **PlantasLenhosas:** Um conjunto de dados que representa plantas lenhosas (árvores, arbustos e lianas), que permite avaliar os exemplos apresentados Capítulo 5, onde diferentes espécies de plantas e suas características são registradas em um SGBDR. Este conjunto de dados é baseado em um do mundo real³. Foram usados os dados sobre a folha, flor e a forma da planta como um todo, de cada planta. A relação que armazena esses dados tem o seguinte esquema:

```
PlantaLenhosa = { Gênero, Nome,
                  FolhaTipo, FolhaArranjo, FolhaForma, FolhaMargem, FolhaTamanho,
                  FlorTipo, FlorCor, FlorTamanho,
                  FormaTipo, FormaTamanho }
```

A criação de uma tabela representante dessa relação utilizando o SQL estendido da ferramenta SIREN pode ser encontrada no Capítulo 5. Além dos comandos de criação da tabela,

³<http://dendro.cnre.vt.edu/> – Acessado pela última vez: Sep 1, 2014

Figura 6.2: Medidas obtidas para cada uma das duas interpretações do operador k -NN executadas pelas consultas **Q11** e **Q12** sobre o conjunto de dados **NSF** com valores de k variando de 1 a 25. (a) Tempo total de consulta. (b) Número de tuplas recuperadas. (c) Número de valores complexos distintos recuperados. (d) Número médio de acessos a disco. (e) Número médio de cálculos de distância.

é necessário associar os atributos complexos **PARTICULATE** com a métrica utilizada, em processo semelhante ao realizado na Seção 6.2. Para isso são realizadas as definições:

```
CREATE METRIC JaccardDist USING MSJaccard
FOR PARTICULATE( F1 VARCHAR(50), F2 VARCHAR(50), F3 VARCHAR(50), F4
VARCHAR(50), F5 VARCHAR(50));

METRIC Folha REFERENCES(
FolhaTipo VARCHAR(50), FolhaArranjo VARCHAR(50), FolhaForma VARCHAR(50),
```

```
FolhaMargem VARCHAR(50), FolhaTamanho VARCHAR(50))
  USING (JaccardDist);
```

Essas duas definições, a criação da métrica e da referência, efetivamente geram os atributos complexos particulados na ferramenta SIREN. Primeiro criando uma métrica que indica a função de distância e os tipos de atributos utilizados, e em seguida associando essa métrica a um atributo PARTICULATE – nesse caso o atributo Folha – e seus componentes na tabela – nesse caso os atributos FolhaTipo, FolhaArranjo, FolhaForma, FolhaMargem e FolhaTamanho. Um processo análogo é realizado para os atributos Flor e Forma. Vale notar que os atributos tradicionais individualmente (não PARTICULATE), mesmo fazendo parte de um atributo complexo ainda são passíveis de busca e podem ser usados como predicados de desempate individualmente.

Os dados sobre as plantas foram extraídos e normalizados para se adequar mais rigidamente a um conjunto de descrições, sendo algumas entradas necessariamente eliminadas, pois na fonte há valores faltantes ou casos que não seguem um padrão bem definido. A relação resultante contém 912 tuplas (plantas) distintas. Cada atributo que descreve as folhas, flores e formas é de tipo varchar, contendo descrições padronizadas. Por exemplo, para o atributo FolhaTipo, os valores amostrados no conjunto são: (Simples, composta, composta bi-foliolada, tri-foliolada, digitada, composta pinada, composta bi-pinada, composta tri-pinada...)

A avaliação da similaridade entre cada folha foi feita considerando os cinco atributos referentes à suas características presentes na relação (FolhaTipo, FolhaArranjo, FolhaForma, FolhaMargem e FolhaTamanho) como um atributo complexo particulado, representado acima, e a comparação entre elementos foi feita utilizando a distância de Jaccard. Um processo semelhante foi feito aplicado para avaliar flores e formas.

Para avaliar a performance do algoritmo neste conjunto de dados, foram executadas as consultas Q13, Q14 e Q15:

- “**Q13** Quais são as plantas que têm suas folhas entre as k mais similares a folha s_q ? Essa consulta corresponde à execução da expressão $\ddot{O}_{(\text{Folha } k\text{-NN}(\text{jaccard}, k) s_q)} \text{PlantaLenhosa}$.
- “**Q14** Quais são as plantas que têm suas folhas entre as k mais similares a folha s_q ? Em caso de empates, utilize as plantas com flor mais similar à de s_q como critério de desempate. Essa consulta corresponde à execução da expressão $\ddot{O}_{(\text{Folha } k\text{-NN}(\text{jaccard}, k) s_q)\{\text{Flor } k\text{-NN}(\text{jaccard}, 1) s_q\}} \text{PlantaLenhosa}$.
- “**Q15** Quais são as plantas que têm suas folhas entre as k mais similares a folha s_q ? Em caso de empates, utilize as plantas com flor mais similar à de s_q como critério de desempate. Se ainda existirem empates, desempate priorizando as que não forem arbustos. Essa consulta corresponde à execução da expressão $\ddot{O}_{(\text{Folha } k\text{-NN}(\text{jaccard}, k) s_q)\{\text{Flor } k\text{-NN}(\text{jaccard}, 1) s_q, \text{FormaTipo} \neq \text{arbusto}\}} \text{PlantaLenhosa}$.

Em SQL estendido essas consultas podem ser expressas como:

<p>Q13:</p> <pre> SELECT * FROM PlantaLenhosa WHERE Folha NEAR (SELECT Folha FROM PlantaLenhosa WHERE (Genero, Nome) = s_q) STOP AFTER k TUPLES WITH TIE LIST;</pre>	<p>Q14:</p> <pre> SELECT * FROM PlantaLenhosa WHERE Folha NEAR (SELECT Folha FROM PlantaLenhosa WHERE (Genero, Nome) = s_q) STOP AFTER k TUPLES UNTIE USING Flor Near (SELECT Flor FROM PlantaLenhosa WHERE (Genero, Nome) = s_q) WITH TIE LIST;</pre>	<p>Q15:</p> <pre> SELECT * FROM PlantaLenhosa WHERE Folha NEAR (SELECT Folha FROM PlantaLenhosa WHERE (Genero, Nome) = s_q) STOP AFTER k TUPLES UNTIE USING Flor Near (SELECT Flor FROM PlantaLenhosa WHERE (Genero, Nome) = s_q) UNTIE USING FormaTipo $\neq$ 'arbusto' WITH TIE LIST;</pre>
--	--	--

A consulta **Q13** foi utilizada como “*baseline*”, sem nenhum predicado de desempate. O seu resultado foi comparado com o das consultas **Q14** e **Q15**, as quais possuem 1 e 2 predicados de desempate respectivamente. O centro da consulta s_q foi amostrado aleatoriamente das folhas presentes no conjunto de dados, com valores de k variando entre 1 e 25 com incrementos de 5. As medidas foram obtidas a partir da média de 100 consultas com centros s_q variados.

As três consultas foram executadas com o objetivo de avaliar a performance do algoritmo de desempate, tanto em desempenho computacional quanto em sua capacidade de desempatar tuplas e seu efeito na cardinalidade final do conjunto resultado. Para isso, foram coletadas as seguintes medidas:

- Tempo total das 100 consultas para cada valor de k ;
- Quantidade de tuplas recuperadas pela consulta.
- Quantas tuplas a mais que k estavam presentes no resultado final ?

Também foram avaliados o número médio de acessos a disco e cálculos de distância necessários.

Na Figura 6.3 são apresentados os resultados para as três consultas aplicadas sobre o conjunto de dados **PlantasLenhosas**, mostrando as três medidas coletadas. Os resultados mostram que a consulta **Q13** retorna, em média, 2,91 tuplas a mais do valor k requisitado originalmente. A consulta **Q14** abaixa esse valor para 1,31, e a consulta **Q15** para 1,03, apresentando portanto

mais de 50% de redução na quantidade de empates, em média. Isso é obtido com baixo impacto no desempenho, custando menos de 1% do tempo total.

É importante ressaltar que os resultados podem ser bastante diversos, e podem variar de acordo com as características do conjunto de dados e com a utilização de predicados de desempate utilizados. O fator mais importante é que esses critérios podem ser definidos como os mais adequados para cada cenário e conjunto específico. Além da maleabilidade, torna-se assim possível prover ao usuário maior controle sobre o processo, de maneira que este não seja arbitrário e possa ser tratado de maneira significativa para a aplicação.

Figura 6.3: Medidas obtidas para as consultas *Q13*, *Q14* e *Q15* executadas sobre o conjunto de dados *PlantasLenhosas*. (a) Tempo total de consulta. (b) Número de tuplas retornadas. (c) Número de tuplas recuperadas acima do valor k requisitado. (d) Número médio de acessos a disco. (e) Número médio de cálculos de distância.

6.4 Considerações Finais

Os experimentos realizados mostram que os novos operadores e o ferramental para tratamento de empates propostos proveem novas capacidades na representação e manipulação de consultas por similaridade quando inseridas no ambiente relacional, sem impactar significativamente a performance. Os dois operadores derivados das consultas por k -NN munem o usuário com um poder expressivo adicional, que se faz necessário quando este operador está inserido em um ambiente relacional. Já o ferramental para tratamentos de empate capacita o usuário a prover contexto e sentido para o desempate de tuplas provenientes de uma consulta que utiliza o operador k -NN dentro do ambiente relacional.

7.1 Considerações Finais

O estudo dos operadores por similaridade dentro do ambiente relacional é de grande importância para realização de uma integração mais robusta dos SGBDR com tipos de dados complexos. É de grande valia tratar consultas envolvendo esses operadores acompanhados dos operadores tradicionais de maneira homogênea, buscando que o ferramental inteiro dos SGBDR possa ser utilizado durante todas as etapas de processamento igualmente.

Neste trabalho, foram apresentadas duas contribuições inéditas para avançar no desenvolvimento deste ideal. A primeira trata da ambiguidade que ocorre com a inserção de operadores de consulta por similaridade no modelo relacional, mais especificamente o k -NN. Identificou-se que existem de fato duas interpretações quando se refere ao significado do valor k nesse contexto, o qual pode ser interpretado como quantidade de tuplas ou de valores de elementos complexos. Para atender ao problema, foram propostos os dois novos operadores: $\tilde{\sigma}$ e $\hat{\sigma}$, que executam o operador de seleção por similaridade baseado em comparação por k -NN e eliminam a ambiguidade por serem inequívocos no que se referem, o primeiro considerando k como quantidade de valores distintos de elementos complexos, e o segundo como quantidade de tuplas.

A segunda contribuição se dá na forma da introdução de um ferramental teórico e prático que possibilita o tratamento de empates que podem ocorrer quando se emprega comparações por k -NN. A solução proposta se aproveita da integração das comparações por k -NN ao modelo relacional e possibilita a utilização dos demais atributos da relação sendo avaliados na consulta, de maneira que o usuário pode influenciar de forma significativa o processo de desempate evitando que o mesmo ocorra de forma arbitrária ou sem informações de contexto.

Essas contribuições complementam e permitem avançar na construção da integração de buscas por similaridade aos SGBD Relacionais, estendendo diversos trabalhos que já vinham

sendo desenvolvidos, alguns dentro do GBdI-ICMC-USP, de forma a impulsionar cada vez mais os estudos e tecnologias relativos a integração de dados complexos com o modelo relacional e com os SGDB relacionais de forma geral.

7.2 Principais Contribuições

As principais contribuições deste trabalho são:

- Definição de dois novos operadores de seleção por similaridade por k -NN, mais adequados à expressão de consultas por similaridade em um ambiente relacional;
- Revisão dos operadores de busca por similaridade para permitir uma melhor integração entre tipos de dados complexos e escalares dentro do ambiente relacional;
- Definição de um processo controlado e bem definido de resolução de empates gerados pelo operador k -NN dentro de um ambiente relacional;
- Implementação das soluções propostas na ferramenta SIREN, estendendo suas capacidades;
- Publicação do trabalho intitulado “*Embedding k -Nearest Neighbor Queries into Relational Database Management Systems*” no periódico “*Journal of Information and Data Management* Vol. 5 No. 3, e apresentação do mesmo no 29º Simpósio Brasileiro de Bases de Dados (SBBD XXIX), realizado em 2014 na cidade de Curitiba/PR.

7.3 Sugestões de Trabalhos Futuros

Dentre as sugestões de trabalhos futuros destacam-se:

- Identificar as propriedades algébricas dos operadores de seleção por contagem de tuplas e de valores, visando possibilitar reescritas e otimização durante o processamento de consultas;
- Identificar as propriedades algébricas dos operadores de seleção por similaridade com desempate, visando possibilitar reescritas e otimização durante o processamento de consultas;
- Identificar operações de seleção por similaridade com desempate que possuam operadores de comparação θ específicos ocorrendo frequentemente em aplicações específicas, e desenvolver algoritmos especializados para executá-los com maior eficiência;
- Identificar oportunidades de poda que possam ser antecipadas a partir das extensões desenvolvidas, visando agilizar os operadores básicos de busca por k -NN;
- Estudar a possibilidade da inclusão de múltiplos índices em consultas que envolvam tanto predicados por RO e identidade quanto predicados por similaridade, de maneira semelhante ao que já ocorre em consultas envolvendo apenas predicados tradicionais;

- Verificar de qual maneira as duas novas interpretações do k - NN afetam junções por similaridade e funções de agregação baseadas em similaridade, tanto de uma perspectiva conceitual quanto de uma perspectiva algorítmica;
- Estender o trabalho feito com o k - NN para outros operadores por similaridade, como operadores de junção ou que utilizem múltiplos centros de consulta, de maneira a obter uma integração bem definida entre operadores algébricos por similaridades e operadores tradicionais nos SGBDR.

- [1] Sibel Adali, Corey Bufl e Maria Luisa Sapino. “Ranked Relations: Query Languages and Query Processing Methods for Multimedia”. Em: *Multimedia Tools Appl* 24.3 (2004), pp. 197–214.
- [2] Sibel Adali, Maria Luisa Sapino e Brandeis Marshall. “A rank algebra to support multimedia mining applications”. Em: *Proceedings of the 8th international workshop on Multimedia data mining: (associated with the ACM SIGKDD 2007)*. MDM '07. San Jose, California: ACM, 2007, 5:1–5:9. ISBN: 978-1-59593-837-4.
- [3] Sibel Adali et al. “A Multi-Similarity Algebra”. Em: *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD-98)*. Vol. 27,2. ACM SIGMOD Record. New York: ACM Press, 1998, pp. 402–413.
- [4] Charu C Aggarwal. “Re-designing distance functions and distance-based applications for high dimensional data”. Em: *ACM SIGMOD Record* 30.1 (2001), pp. 13–18.
- [5] Adriano S Arantes et al. “Efficient algorithms to execute complex similarity queries in RDBMS”. Em: *Journal of the Brazilian Computer Society* 9.3 (2004), pp. 5–24.
- [6] Adriano S Arantes et al. “Operadores de seleção por similaridade para sistemas de gerenciamento de bases de dados relacionais”. Em: *XVIII Brazilian Symposium on Database* (2003), pp. 341–355.
- [7] Solomon Atnaflu, Lionel Brunie e Harald Kosch. “Similarity-based algebra for multimedia database systems”. Em: *ADC*. 2001, pp. 115–122.
- [8] Solomon Atnaflu, Lionel Brunie e Harald Kosch. “Similarity-Based Operators and Query Optimization for Multimedia Database Systems”. Em: *IDEAS*. Ed. por Michel E. Adiba, Christine Collet e Bipin C. Desai. IEEE Computer Society, 2001, pp. 346–355. ISBN: 0-7695-1140-6.
- [9] Solomon Atnaflu et al. “Integrating similarity-based queries in image DBMSs”. Em: *SAC*. Ed. por Hisham Haddad et al. ACM, 2004, pp. 735–739. ISBN: 1-58113-812-1.

- [10] Mohammad Awrangjeb, Guojun Lu e Clive S. Fraser. “Performance Comparisons of Contour-Based Corner Detectors”. Em: *Image Processing, IEEE Transactions on* 21.9 (2012), pp. 4167–4179.
- [11] Maria Camila Nardini Barioni et al. “Querying Multimedia Data by Similarity in Relational DBMS”. Em: *Advanced Database Query Systems: Techniques, Applications and Technologies*. Ed. por Li Yan e Zongmin Ma. Chapter 14. Hershey, NY, USA: IGI Global, 2010, pp. 323–359.
- [12] Maria Camila Nardini Barioni et al. “Seamlessly Integrating Similarity Queries in SQL”. Em: *Software: Practice and Experience* 39.4 (2009), pp. 355–384.
- [13] Maria Camila Nardini Barioni et al. “SIREN: A Similarity Retrieval Engine for Complex Data”. Em: *Demo session of the International Conference on Very Large Data Bases*. Seoul, South Korea: ACM, 2006, pp. 1155–1158.
- [14] Michal Batko, David Novak e Pavel Zezula. “MESSIF: Metric Similarity Search Implementation Framework”. Em: *DELOS Conference*. Ed. por Costantino Thanos, Francesca Borri e Leonardo Candela. Vol. 4877. Lecture Notes in Computer Science. Springer, 2007, pp. 1–10. ISBN: 978-3-540-77087-9.
- [15] Radim Belohlávek, Stanislav Opichal e Vilém Vychodil. “Relational Algebra for Ranked Tables with Similarities: Properties and Implementation”. Em: *IDA*. Ed. por Michael R. Berthold, John Shawe-Taylor e Nada Lavrac. Vol. 4723. Lecture Notes in Computer Science. Springer, 2007, pp. 140–151. ISBN: 978-3-540-74824-3.
- [16] Radim Belohlávek, Lucie Urbanova e Vilém Vychodil. “Sensitivity Analysis for Declarative Relational Query Languages with Ordinal Ranks”. Em: *CoRR* abs/1109.6299 (2011).
- [17] Radim Belohlávek e Vilém Vychodil. “Query systems in similarity-based databases: logical foundations, expressive power, and completeness”. Em: *SAC*. Ed. por Sung Y. Shin et al. ACM, 2010, pp. 1648–1655. ISBN: 978-1-60558-639-7.
- [18] Radim Belohlávek e Vilém Vychodil. “Relational Algebra for Multi-ranked Similarity-based Databases”. Em: *IEEE Symposium on Foundations of Computational Intelligence*. Singapore, Singapore, 2013, pp. 1–8.
- [19] Bozkaya e Ozsoyoglu. “Indexing Large Metric Spaces for Similarity Search Queries”. Em: *ACMTDS: ACM Transactions on Database Systems* 24 (1999).
- [20] Tolga Bozkaya e Meral Ozsoyoglu. “Distance-Based Indexing for High-Dimensional Metric Spaces”. Em: *SIGMOD Record (ACM Special Interest Group on Management of Data)* 26.2 (1997), pp. 357–368.
- [21] Petra Budíková, Michal Batko e Pavel Zezula. “Query Language for Complex Similarity Queries”. Em: *CoRR* abs/1204.1185 (2012).

- [22] Burkhard e Keller. “Some Approaches to Best-Match File Searching”. Em: *CACM: Communications of the ACM* 16 (1973).
- [23] Luiz Olmes Carvalho et al. “A Wider Concept for Similarity Joins”. Em: *Journal of Information and Data Management* 5.3 (2014), pp. 210–223.
- [24] R. G. G. Cattell. *The Object Database Standard OMDG-93*. Morgan-Kaufmann, 1993.
- [25] Kaushik Chakrabarti et al. “Evaluating Refined Queries in Top-k Retrieval Systems”. Em: *IEEE Transactions on Knowledge and Data Engineering (TKDE)* 16.1 (2004). Adriano, Gisele, pp. 256–270.
- [26] Ciaccia et al. “Imprecision and User Preferences in Multimedia Queries: A Generic Algebraic Approach”. Em: *FOIKS: International Symposium on Foundations of Information and Knowledge Systems*. LNCS, 2000.
- [27] Paolo Ciaccia, Marco Patella e Pavel Zezula. “M-tree: An efficient access method for similarity search in metric spaces”. Em: *International Conference on Very Large Databases*. Athens, Greece: Morgan Kaufmann, 1997, pp. 426–435.
- [28] Edward F. Codd. “A Relational Model of Data for Large Shared Data Banks.” Em: *Communications of the ACM (CACM)* 13.6 (1970), pp. 377–387.
- [29] Chris J. Date. *SQL and Relational Theory - How to Write Accurate SQL Code*. O’Reilly, 2009, pp. I–XIX, 1–404. ISBN: 978-0-596-52306-0.
- [30] C.J. Date. *An Introduction to Database Systems*. 8^a ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2003. ISBN: 0321197844.
- [31] Thomas M. Deserno, Sameer Antani e Rodney Long. “Ontology of Gaps in Content-based Image Retrieval”. Em: *Springer Journal of Digital Imaging* 22.2 (2008). Springer On-line First: Friday, February 01, 2008, pp. 202–215.
- [32] Carlotta Domeniconi et al. “Locally adaptive metrics for clustering high dimensional data”. Em: *Data Mining and Knowledge Discovery* 14.1 (2007). Robson, pp. 63–97.
- [33] Ömer Egecioglu, Hakan Ferhatosmanoglu e Umit Ogras. “Dimensionality Reduction and Similarity Computation by Inner-Product Approximations”. Em: *IEEE Transactions on Knowledge and Data Engineering (TKDE)* 16.6 (2004), pp. 714–726.
- [34] Ramez Elmasri e Shamkant Navathe. *Fundamentals of Database Systems*. 6th. USA: Addison-Wesley Publishing Company, 2010. ISBN: 0136086209, 9780136086208.
- [35] Christos Faloutsos. “Indexing of multimedia data”. Em: *Multimedia Databases in Perspective*. Springer, 1997, pp. 219–245.
- [36] Christos Faloutsos. *Searching multimedia databases by content*. Vol. 3. Springer, 1996.
- [37] Joaquim Cezar Felipe e Agma Juci Machado Traina. “Utilizando Características de Textura para Identificação de Tecidos em Imagens Médicas”. Em: *2º Workshop de Informática Médica (WIM) 2001 - in CD-ROM*. Gramado, RS, 2002, in CD-ROM.

- [38] Mônica Ribeiro Porto Ferreira, Caetano Traina Jr. e Agma Juci Machado Traina. “An Efficient Framework for Similarity Query Optimization”. Em: *ACM International Symposium on Advances in Geographic Information Systems*. Seattle, WA, USA: ACM, 2007, pp. 396–399.
- [39] Mônica Ribeiro Porto Ferreira et al. “Algebraic Properties to Optimize kNN Queries”. Em: *Journal of Information and Data Management* 2.3 (2011), pp. 385–400.
- [40] Mônica Ribeiro Porto Ferreira et al. “Identifying Algebraic Properties to Support Optimization of Unary Similarity Queries”. Em: *Alberto Mendelzon International Workshop on Foundations of Data Management*. Vol. 450. CEUR Workshop Proceedings. Arequipa, Peru: CEUR-WS, 2009, pp. 1–10.
- [41] Flavio Figueiredo et al. “Assessing the quality of textual features in social media”. Em: *Information Processing and Management* 49.1 (2013), pp. 222–247.
- [42] Hector Garcia-Molina, Jeffrey D. Ullman e Jennifer Widom. *Database systems - the complete book (international edition)*. Pearson Education, 2002, pp. I–XXVII, 1–1119. ISBN: 978-0-13-098043-4.
- [43] GBDI-ICMC-USP. *GBDI Arboretum Library*. <http://www.gbdi.icmc.usp.br/downloads/arboretum/>. Acessado: 2014-04-24.
- [44] Mark O. G’uld et al. “A Generic Concept for the Implementation of Medical Image Retrieval Systems”. Em: *International Journal of Medical Informatics (IJMI)* 76 (2007), p. 252.
- [45] Denise Guliato et al. “POSTGRESQL-IE: an Image-handling Extension for PostgreSQL”. Em: *Journal of Digital Imaging* 22.2 (2009), pp. 149–165.
- [46] Robert M. Haralick, K. Sam Shanmugam e Its’hak Dinstein. “Textural Features for Image Classification”. Em: *IEEE Transactions on Systems, Man, and Cybernetics* 3.6 (1973), pp. 610–621.
- [47] Haibo Hu e Dik Lun Lee. “Range Nearest-Neighbor Query”. Em: *IEEE Transactions on Knowledge and Data Engineering (TKDE)* 18.1 (2006). Adriano, Ana Paula, pp. 78–91.
- [48] Ioannidis. “Query Optimization”. Em: *CSURV: Computing Surveys* 28 (1996).
- [49] Yannis E. Ioannidis. “Query Optimization”. Em: *The Computer Science and Engineering Handbook*. 1997, pp. 1038–1057.
- [50] Ramesh Jain e Pinaki Sinha. “Content without context is meaningless”. Em: *ACM Multimedia*. Ed. por Alberto Del Bimbo, Shih-Fu Chang e Arnold W. M. Smeulders. ACM, 2010, pp. 1259–1268. ISBN: 978-1-60558-933-6.
- [51] N. K. Kamila, S. Mahapatra e S. Nanda. “Retracted Paper: Invariance image analysis using modified Zernike moments”. Em: *Pattern Recognition Letters* 26.6 (maio de 2005), pp. 747–753.

- [52] Daniel S. Kaster et al. “FMI-SiR: a Flexible and Efficient Module for Similarity Searching on Oracle Database”. Em: *Journal of Information and Data Management* 1.2 (2010), pp. 229–244.
- [53] Daniel S. Kaster et al. “MedFMI-SiR: a Powerful DBMS Solution for Large-Scale Medical Image Retrieval”. Em: *International Conference on Information Technology in Bio- and Medical Informatics*. Toulouse, France, 2011, pp. 16–30.
- [54] Bevan Koopman et al. “An evaluation of corpus-driven measures of medical concept similarity for information retrieval”. Em: *Proceedings of the 21st ACM international conference on Information and knowledge management*. CIKM '12. Maui, Hawaii, USA: ACM, 2012, pp. 2439–2442. ISBN: 978-1-4503-1156-4.
- [55] Flip Korn et al. “Fast Nearest Neighbor Search in Medical Image Databases”. Em: *International Conference on Very Large Databases (VLDB)*. Ed. por T. M. Vijayaraman et al. Bombay, India: Morgan Kaufmann, 1996, pp. 215–226.
- [56] Harald Kosch e Andreas W’olf. “Large-Scale Similarity-Based Join Processing in Multimedia Databases”. Em: *18th International Conference, MMM Advances in Multimedia Modeling*. Ed. por Klaus Schoeffmann et al. Vol. 7131. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2012, pp. 418–428.
- [57] J. B. Kruskal. “On the shortest spanning subtree of a graph and the travelling salesman problem”. Em: *Proc. Am. Math. Soc.* Published as Proc. Am. Math. Soc., volume 7, number 1. Fev. de 1956, pp. 48–50.
- [58] Chengkai Li et al. “RankSQL: Query Algebra and Optimization for Relational Top-k Queries”. Em: *SIGMOD Conference*. Ed. por Fatma Özcan. ACM, 2005, pp. 131–142. ISBN: 1-59593-060-4.
- [59] John Z. Li et al. *MOQL: A Multimedia Object Query Language*. en. 1997.
- [60] Elon Lages Lima. *Espaços Métricos*. Instituto de Matemática Pura e Aplicada, 1993.
- [61] Bing Liu et al. “A Bottom-Up Distance-Based Index Tree for Metric Space”. Em: *RSKT*. Ed. por Guoyin Wang et al. Vol. 4062. Lecture Notes in Computer Science. Springer, 2006, pp. 442–449. ISBN: 3-540-36297-5.
- [62] Ying Liu et al. “A Survey of Content-based Image Retrieval with High-level Semantics”. Em: *Pattern Recognition Letters* 40 (2007). Elsevier Ltd., pp. 262 –282.
- [63] Jakub Lokoc et al. “Visual Image Search: Feature Signatures or/and Global Descriptors”. Em: *Similarity Search and Applications*. Ed. por Gonzalo Navarro e Vladimir Pestov. Vol. 7404. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2012, pp. 177–191. ISBN: 978-3-642-32152-8.

- [64] Rahul Malik et al. “MLR-Index: An Index Structure for Fast and Scalable Similarity Search in High Dimensions”. Em: *21st International Conference on Scientific and Statistical Database Management, SSDBM 2009*. Ed. por Marianne Winslett. Vol. 5566. Lecture Notes in Computer Science. New Orleans, LA: Springer, 2009, pp. 167–184.
- [65] Wadha J. Al Marri et al. “The Similarity-Aware Relational Intersect Database Operator”. Em: *Similarity Search and Applications - 7th International Conference, SISAP 2014, Los Cabos, Mexico, October 29-31, 2014. Proceedings*. 2014, pp. 164–175.
- [66] Jim Melton e Alan R. Simon. *SQL:1999 Understanding Relational Language Components*. 1ª ed. The Morgan Kaufmann series in Data Management Systems. Morgan Kaufmann, 2002.
- [67] Gonzalo Navarro. “Searching in metric spaces by spatial approximation”. Em: *VLDB J* 11.1 (2002), pp. 28–46.
- [68] Aurélie Névéol et al. “Natural language processing versus content-based image analysis for medical document retrieval”. Em: *Journal of the American Society for Information Science and Technology (JASIST)* 60.1 (2009), pp. 123–134.
- [69] Wilma Penzo. “Rewriting Rules To Permeate Complex Similarity and Fuzzy Queries within a Relational Database System”. Em: *IEEE Trans. Knowl. Data Eng* 17.2 (2005), pp. 255–270.
- [70] Ives Rene Venturini Pola, Agma Juci Machado Traina e Caetano Traina Jr. “Easing the Dimensionality Curse by Stretching Metric Spaces”. Em: *21st International Conference on Scientific and Statistical Database Management, SSDBM 2009*. Ed. por Marianne Winslett. Vol. 5566. Lecture Notes in Computer Science. New Orleans, LA: Springer, 2009, pp. 417–434.
- [71] Humberto Luís Razente et al. “A Novel Optimization Approach to Efficiently Process Aggregate Similarity Queries in MAM”. Em: *ACM 17th International Conference on Information and Knowledge Management (CIKM 08)*. Ed. por James G. Shanahan et al. Napa Valley, CA: ACM Press, 2008, pp. 193–202.
- [72] Humberto Luís Razente et al. “Constrained Aggregate Similarity Queries in Metric Spaces”. Em: *22nd Simpósio Brasileiro de Bases de Dados (SBBD 07)*. Vol. 1. João Pessoa, PB: SBC, 2007, pp. 143–159.
- [73] Elton Serra et al. “On Using Wikipedia to Build Knowledge Bases for Information Extraction by Text Segmentation”. Em: *Journal of Information and Data Management* 2.3 (2011), pp. 259–272.
- [74] Yasin N. Silva, Walid G. Aref e Mohamed H. Ali. “Similarity Group-By”. Em: *Proceedings of the 25th International Conference on Data Engineering, ICDE 2009, March 29 2009 - April 2 2009, Shanghai, China*. 2009, pp. 904–915.

- [75] Yasin N. Silva e Spencer Pearson. “Exploiting Database Similarity Joins for Metric Spaces”. Em: *Proc. VLDB Endow.* 5.12 (2012), pp. 1922–1925.
- [76] Yasin N. Silva et al. “SimDB: a Similarity-aware Database System”. Em: *ACM SIGMOD International Conference on Management of Data*. Indianapolis, Indiana, USA, 2010, pp. 1243–1246.
- [77] Yasin N. Silva et al. “Similarity Queries: their Conceptual Evaluation, Transformations, and Processing”. Em: *The VLDB Journal* 22.3 (2013), pp. 395–420.
- [78] Swain e Ballard. “Color Indexing”. Em: *IJCV: International Journal of Computer Vision* 7 (1991).
- [79] Yufei Tao et al. “Multidimensional reverse kNN search”. Em: *International Journal on Very Large Data Bases* 16.3 (2007), pp. 293–316.
- [80] Google Image Search Team. *Improving Photo Search: A Step Across the Semantic Gap*. <http://googleresearch.blogspot.com.br/2013/06/improving-photo-search-step-across.html>. Acessado: 2015-05-22.
- [81] Ricardo S. Torres et al. “Visual structures for image browsing”. Em: *International Conference on Information and Knowledge Management*. New Orleans, LA: ACM, 2003, pp. 49–55.
- [82] Agma Juci Machado Traina et al. “Feature Extraction and Selection for Decision Making over Medical Images”. Em: *Biomedical Image Processing - Methods and Applications*. Ed. por Thomas M. Deserno. Springer-Verlag, 2010, pp. 181–209.
- [83] Agma Juci Machado Traina et al. “How to Cope with the Performance Gap in Content-based Image Retrieval Systems”. Em: *International Journal of Healthcare Information Systems and Informatics (IJHISI)* 4.1 (2009), pp. 47–67.
- [84] Caetano Traina Jr. et al. “Efficient processing of complex similarity queries in RDBMS through query rewriting”. Em: *CIKM*. Ed. por Philip S. Yu et al. ACM, 2006, pp. 4–13. ISBN: 1-59593-433-2.
- [85] Caetano Traina Jr. et al. “Fast Indexing and Visualization of Metric Datasets Using Slim-trees”. Em: *IEEE Transactions on Knowledge and Data Engineering* 14.2 (2002), pp. 244–260.
- [86] Caetano Traina Jr. et al. “The OMNI-Family of All-Purpose Access Methods: A Simple and Effective Way to Make Similarity Search More Efficient”. Em: *The International Journal on Very Large Databases* 16.4 (2007), pp. 483–505.
- [87] Uhlmann. “Satisfying General Proximity / Similarity Queries with Metric Trees”. Em: *IPL: Information Processing Letters* 40 (1991).

- [88] Jayendra Venkateswaran et al. “Reference-based indexing for metric spaces with costly distance measures”. Em: *The International Journal on Very Large Databases* 17.5 (2009), pp. 1231–1251.
- [89] Marcos R. Vieira et al. “DBM-Tree: A Dynamic Metric Access Method Sensitive to Local Density Data”. Em: *JIDM* 1.1 (2010), pp. 111–128.
- [90] Petra Welter et al. “Generic integration of content-based image retrieval in computer-aided diagnosis”. Em: *Computer Methods and Programs in Biomedicine* 108.2 (2012), pp. 589–599.
- [91] Lei Wu. “Flickr Distance: A Relationship Measure for Visual Concepts”. Em: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34.5 (2012), pp. 863–875.
- [92] Bin Yao, Feifei Li e Piyush Kumar. “K nearest neighbor queries and kNN-Joins in large relational databases (almost) for free”. Em: *International Conference on Data Engineering*. Ed. por Feifei Li e Piyush Kumar. Long Beach, CA, USA: IEEE Computer Society, 2010, pp. 4–15.
- [93] Yianilos. “Data Structures and Algorithms for Nearest Neighbor Search in General Metric Spaces”. Em: *SODA: ACM-SIAM Symposium on Discrete Algorithms (A Conference on Theoretical and Experimental Analysis of Discrete Algorithms)*. 1993.
- [94] Man Lung Yiu, Nikos Mamoulis e Dimitris Papadias. “Aggregate nearest neighbor queries in road networks”. Em: *IEEE Transactions on Knowledge and Data Engineering (TKDE)* 17.6 (2005), pp. 820–833.
- [95] Pavel Zezula. “Future Trends in Similarity Searching”. Em: *Similarity Search and Applications*. Ed. por Gonzalo Navarro e Vladimir Pestov. Vol. 7404. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2012, pp. 8–24. ISBN: 978-3-642-32152-8.
- [96] Pavel Zezula. “Multi Feature Indexing Network MUFIN for Similarity Search Applications”. Em: *SOFSEM 2012: Theory and Practice of Computer Science*. Ed. por Mária Bieliková et al. Vol. 7147. Lecture Notes in Computer Science. Špindlerův Mlýn, Czech Republic: Springer Berlin / Heidelberg, 2012, pp. 77–87.
- [97] Pavel Zezula et al. *Similarity Search: The Metric Space Approach*. Advances in Database Systems. New York, NY, USA: Springer New York, 2006.