Suporte a consultas por similaridade unárias em SQL

Mônica Ribeiro Porto Ferreira

Suporte a consultas por similaridade unárias em SQL

Mônica Ribeiro Porto Ferreira

Orientador: ***Prof. Dr. CaetanoTraina Junior***

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.

USP – São Carlos
Janeiro/2008

"O professor é o pai intelectual do discípulo."

Machado de Assis

Agradecimentos

Ao Prof. Dr. Caetano Traina Júnior, meu orientador e amigo, por acreditar em mim desde o início, quando me convidou para fazer iniciação científica. Por todos os anos de orientação. Pelas oportunidades e desafios que me proporcionou, por seu apoio e incentivo em todos os momentos e, principalmente, por sua confiança.

Ao Leandro, meu amado noivo, por estar sempre presente, compartilhando as alegrias dos momentos de sucesso e me incentivando e ajudando a superar os momentos difíceis. Por compreender minha ausência nos inúmeros finais de semana e feriados em que tive que trabalhar. Por me ajudar a revisar toda a dissertação. Pelo abraço e palavras carinhosas que faz tudo parecer muito mais fácil.

A meus pais, José Maria e Mayra, à minha avó Ladice e a todo o resto da minha família, que é muito grande e não dá para citar um por um, pelo amor, apoio e estímulo dedicados incondicionalmente durante toda a minha vida. Por entenderem minha ausência nos aniversários, feriados e reuniões em família. Por vibrarem com cada uma de minhas conquistas, grandes ou pequenas.

Em especial, à minha avó e heroína, Carminda, pelo amor e apoio. Por ser meu exemplo de vida. Pelo estímulo dedicado incondicionalmente durante toda a minha vida. Por vibrar muito com cada uma de minhas conquistas. Por ler toda a dissertação e me ajudar com a revisão gramatical.

À minha cunhada, Elaine, especialmente, por me ensinar e me ajudar em muitas coisas durante todo o meu mestrado e, principalmente, na minha formação acadêmica. Por me ajudar a revisar toda a dissertação.

Aos meus irmãos, José Maria Jr. e José Guilherme, à minha cunhada, Josélia, e ao meu sobrinho, João Pedro, pela cumplicidade e por estar sempre por perto para o que for preciso.

À Profa. Agma Traina, pelos trabalhos realizados em conjunto, pelas palavras de incentivo, pela dedicação e pelo carinho demonstrados de uma maneira sempre tão gentil.

À Profa. Ires Dias, pelos trabalhos realizados em conjunto, pela dedicação e por me ajudar com a álgebra.

Aos amigos e colegas do GBDI, especialmente à Marcela, Carolina e Camila e ao Humberto, Daniel, Ives, André, Junior, Robson, pela colaboração importante nas reuniões do grupo e fora delas.

A Beth, Laura e Ana Paula, por todo o auxílio nos assuntos de competência da Secretaria de Pós-Graduação.

Ao Instituto de Ciências Matemáticas e de Computação da USP São Carlos, pela estrutura acadêmica que tornou possível o desenvolvimento deste trabalho.

À FAPESP, Fundação de Amparo à Pesquisa do Estado de São Paulo, pelo auxílio financeiro.

Resumo

Os operadores convencionais para comparação de dados por igualdade e por relação de ordem total não são adequados para o gerenciamento de dados complexos como, por exemplo, os dados multimídia (imagens, áudio, textos longos), séries temporais e seqüências genéticas. Para comparar dados desses tipos, o grau de similaridade entre suas instâncias é, em geral, o fator mais importante sendo, portanto, indicado que as operações de consulta sejam realizadas utilizando os chamados operadores por similaridade. Existem operadores de busca por similaridade tanto unários quanto binários. Os operadores unários são utilizados para implementar operações de seleção, enquanto os operadores binários destinam-se a operações de junção. A álgebra relacional, usada nos Sistemas de Gerenciamento de Bases de Dados Relacionais, não provê suporte para expressar critérios de busca por similaridade. Para suprir esse suporte, está em desenvolvimento no Grupo de Bases de Dados e Imagens (GBdI-ICMC-USP) uma extensão à álgebra relacional que permite representar as consultas por similaridade em expressões algébricas. Esta dissertação incorpora-se nesse empreendimento, abordando o tratamento aos operadores unários por similaridade na álgebra, bem como a implementação do otimizador de consultas por similaridade no SIREN (*Similarity Retrieval Engine*) para que as consultas por similaridade possam ser respondidas pelos Sistemas de Gerenciamento de Bases de Dados relacionais.

Abstract

Conventional operators for data comparison based on exact matching and total order relations are not appropriate to manage complex data, such as multimedia data (e.g. images, audio and large texts), time series and genetic sequences. In fact, the most important aspect to compare complex data is usually the similarity degree between instances, leading to the use of similarity operators to perform search and retrieval operations. Similarity operators can be classified as unary or as binary, respectively used to implement selection operations and joins. However, the Relation Algebra, employed in Relational Database Management Systems (DBMS), does not provide resources to express similarity search criteria. In order to fulfill this lack of support, an extension to the Relational Algebra is under development at GBdI-ICMC-USP (Grupo de Bases de Dados e Imagens), aiming to represent similarity queries in algebraic expressions. This work contributes to such an effort by dealing with unary similarity operators in Relational Algebra and by developing a similarity query optimizer for SIREN (Similarity Retrieval Engine), therefore allowing similarity queries to be answered by Relational DBMS.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

Lista de Siglas e Acrônimos

ACT	Tabela de Condições e Atributos ou *Attributes and Conditions Table.*
b-Rdnn-tree	*bichromatic Rdnn-tree.*
Ball Decomposition	Decomposição por bolas.
BLOB	*Binary Large Object.*
bu-tree	*bottom-up index tree.*
CPU	*Central Processing Unit* ou Unidade Central de Processamento.
DBM-tree	*Density-Based Metric tree.*
DBMS	*Database Management Systems.*
DF-tree	*Distance Fields tree.*
EGNAT	*Evolutionary Geometric Near-Neighbor Access Tree.*
E/S	Entrada/Saída.
FQ-tree	*Fixed Queries tree.*
GBdI-ICMC-USP	Grupo de Bases de Dados e Imagens - Instituto de Ciências Matemáticas e de Computação - Universidade de São Paulo.
GeVAS	*Generalized VA-File-based Search.*
GH-tree	Decomposição de hiperplanos generalizados ou *Generalized Hyperplane tree.*
GNAT	*Geometric Near-Neighbor Access Tree.*
ICMC	Instituto de Ciências Matemáticas e de Computação.
MAM	*Metric Access Method* ou Método de Acesso Métrico.
MM-tree	*Memory-based Metric tree.*
MSA	*Multi-similarity algebra.*
MVP-tree	*Multi-Vantage-Point tree.*
MuX	*multipage index.*

Rdnn-tree	*R-tree with distance to nearest neighbor.*
ROT	Relação de Ordem Total.
RSJ	*R-tree Spatial Join.*
SA-tree	*Spatial approximation tree.*
SGBD	Sistema de Gerenciamento de Bases de Dados.
SIG	Sistema de Informação Geográfica.
SIREN	*Similarity Retrieval Engine.*
SQL	*Structured Query Language.*
VA-File	*Vector Approximation File.*
VP-tree	*Vantage-Point tree.*

Lista de Símbolos

$\mathbb{S}, \mathbb{A}, \mathbb{T}$	Domínio de elementos.
T, T_1, T_2	Relação ou conjunto de dados no qual as consultas são realizadas ($T, T_1, T_2 \subseteq \mathbb{S}$).
d	Função distância, função de dissimilaridade ou métrica ($d : \mathbb{S} \times \mathbb{S} \rightarrow \mathbb{R}^+$).
s_q	Elemento de consulta ou elemento central de consulta ($s_q \in \mathbb{S}$).
L_2	Distância ou função Euclidiana.
h	Número de operações relacionais em uma consulta.
g	Número de métodos que cada operação pode ser avaliada.
U, V, W	Relação ou conjunto de dados.
c, c_1, c_2	Condições de seleção ou de junção.
σ	Operador de seleção.
$\wedge$	Operador Booleano E (*and*).
$\cap$	Operador de conjuntos: Interseção.
$\bowtie$	Operador de junção natural.
$\times$	Operador de produto cartesiano.
AL	Atributos projetados.
α, α'	Atributo da relação U.
β, β'	Atributo da relação V.
π	Operador de projeção.
$\overset{c}{\bowtie}$	Operador de junção-θ.
op	Operador de igualdade ou relacional ($=$, $\neq$, $<$, $\leq$, $>$ ou $\geq$).
Op_1, Op_2	Operação.
R^+	Números reais positivos.
$M =< \mathbb{S}, d >$	Espaço Métrico.
A	Atributo amostrado em um domínio $\mathbb{A}$ que atende à ROT.

S, S_1, S_2	Atributo amostrado no domínio $\mathbb{S}$ de um espaço métrico.
t, t_i, t_j, t_n, t_m	tupla.
a, s	Elementos de um domínio ($a \in \mathbb{A}$ e $s \in \mathbb{S}$).
$t_i(atributo)$	valor da i-ésima tupla no atributo.
$\hat{\sigma}$	Operador de seleção por similaridade.
$\langle S\ Op_s(d, lim)\ s_q \rangle$	Predicado de seleção por similaridade.
$\langle Op_s(d, lim) \rangle$	Operador unário de seleção por similaridade.
lim	Limite de dissimilaridade ou *threshold*.
T'	Subconjunto resposta.
R_q	Similaridade por Abrangência ou *Similarity Range query*.
ξ	Limite de dissimilaridade ou *threshold*.
$d(t_i(S), s_q)$	Distância do elemento de consulta s_q ao elemento $t_i(S)$.
kNN	Similaridade aos k-vizinhos mais próximos ou *k-Nearest Neighbor Query*.
k	Número de elementos retornados na consulta.
$\|T\|$	Cardinalidade da relação T.
$\langle S_1\ Op_j(d, lim)\ S_2 \rangle$	Predicado de junção por similaridade.
$\langle Op_j(d, lim) \rangle$	Operador binário de junção por similaridade.
$< t_i, t_j >$	Par de elementos em que $t_i \in T_1$ e $t_j \in T_2$.
$\overset{R_q}{\bowtie}$	Junção por abrangência ou *Range Join*.
$\overset{kNN}{\bowtie}$	Junção por vizinhança ou *k-Nearest Neighbor Join*.
$\overset{kCN}{\bowtie}$	Junção por proximidade ou *k-Closest Neighbor Join*.
kCN	*k-Closest Neighbor*.
s_{rep}	Elemento representante do conjunto ($s_{rep} \in \mathbb{S}$).
s_i	Novo elemento inserido em um MAM $s_i \in \mathbb{S}$.
$Range(s_q, \xi)$	Algoritmo para consultas por abrangência.
$Nearest(s_q, k)$	Algoritmo para consultas aos k-vizinhos mais próximos.
$range$	Predicado *range*.
$k - nearest$	Predicado *k-nearest*.
$\vee$	Operador booleano OU.
$\neg$	Operador booleano Negação.

D	Atributo implícito de distância.
R_q^{-1}	Seleção por similaridade por abrangência inversa ou *Reversed Range query*.
$\ddot{\sigma}$	Operador de seleção por similaridade por vizinhança.
kFN	Similaridade aos k-vizinhos mais distantes *k-Farthest Neighbor Query*.
$\cup$	Operador de conjuntos: União.
$-$	Operador de conjuntos: Diferença.
θ	Operador genérico.
a	Constante.
$\varnothing$	Conjunto vazio.
AR_q	Similaridade agregada por abrangência.
$AkNN_q$	Similaridade agregada aos k-vizinhos mais próximos.

Capítulo

1

Introdução

1.1 Motivação

Os Sistemas de Gerenciamento de Bases de Dados (SGBDs) relacionais foram inicialmente projetados para suportar dados de tipos numéricos e pequenas cadeias de caracteres, que nesta dissertação são chamados de "**dados tradicionais**". Sobre esses dados existem fundamentalmente dois tipos de operadores de comparação que são amplamente utilizados: operadores para comparação por igualdade e operadores relacionais. Os **operadores de comparação por igualdade** ($=$ e $\neq$) podem ser universalmente aplicados a qualquer tipo de dados, pois é sempre possível decidir se dois elementos são iguais ou não. Os **operadores relacionais** ($<$, $\leq$, $>$ e $\geq$) necessitam que os dados comparados estejam em domínios que atendam à chamada "**Relação de Ordem Total**" (ROT). Esta propriedade permite comparar quaisquer pares de elementos de dados e decidir qual deles precede/sucede ao outro. Juntos, os operadores relacionais e por igualdade compõem os operadores mais comuns encontrados em SGBDs relacionais, os chamados de "*the big six*" da linguagem SQL - *Structured Query Language* (Melton et al., 2002). Esses operadores são amplamente aplicáveis aos tipos de dados tradicionais dos SGBDs relacionais, como números, pequenas cadeias de caracteres, datas e períodos de tempo. Já outros operadores de comparação, suportados em SGBDs relacionais, são aplicáveis a apenas alguns desses tipos de dados tradicionais, tais como os operadores de continência, que são aplicáveis apenas a cadeias de caracteres e a períodos de tempo.

Os SGBDs relacionais atuais aproveitam a propriedade de ROT existente entre os elementos dos domínios de dados tradicionais para executar as operações de consulta e atualização dos dados. Esta propriedade garante que, dados dois elementos distintos quaisquer do mesmo domínio, sempre se pode dizer qual elemento precede o outro. Mesmo

quando uma operação de busca envolve apenas operações de comparação por igualdade, as estruturas de indexação usadas para agilizar a consulta dependem de que a propriedade de ROT seja atendida pelo respectivo domínio de dados. A aplicabilidade dos operadores relacionais e de igualdade a todos os tipos de dados tradicionais em SGBDs relacionais levou ao desenvolvimento tanto de técnicas de indexação muito eficientes quanto de estruturas genéricas para a representação sintática nas linguagens de consulta para todos esses tipos de dados.

No entanto, os requisitos impostos por muitas das novas aplicações sobre os SGBDs relacionais têm gerado a necessidade de suporte tanto para novos tipos de dados quanto para novos tipos de consultas que sejam mais adequadas a eles. Um exemplo de tipos de dados que não atendem à ROT e que requerem tipos de consultas específicas são os chamados tipos de **dados espaciais**, dos quais aqueles com dimensionalidade dois e três são muito usados, por exemplo, em Sistemas de Informações Geográficas (SIGs), e aqueles com dimensionalidade maior são muito usados, por exemplo, em Sistemas de Sensoriamento Remoto. Nesses sistemas, usualmente são necessários operadores de consulta específicos que envolvem a noção de dimensões espaciais, tais como as consultas topológicas (`intercepta`, `adjacente a`, entre outras) e as consultas cardinais (baseadas em ângulos, tais como `ao norte`, `a sudeste`, `acima`, `à esquerda`, entre outras) (Gaede e Günther, 1998).

Porém, muitos **domínios de dados complexos**, como os domínios de dados multimídia (imagens, áudio, textos longos), séries temporais, seqüências genéticas, entre outros, geralmente não atendem à ROT e tampouco apresentam a noção de dimensões, impedindo também consultas topológicas ou cardinais. De fato, os operadores relacionais não são aplicáveis a dados de tipos complexos. Por exemplo: genericamente, não é possível ordenar imagens, a menos que elas sejam associadas a algum atributo extra não complexo (como: nome ou data). Em domínios de **dados complexos**, mesmo as operações de comparação por igualdade têm pouca utilidade, uma vez que a existência de dois elementos exatamente iguais é muito rara (Faloutsos, 1996) e praticamente inútil para operações de recuperação nestes domínios. Assim, para dados complexos, as **consultas por similaridade** tornam-se a solução mais adequada e o grau de similaridade entre os dados é o fator mais importante (Faloutsos, 1997).

Os operadores de consulta por similaridade aplicam-se a muitos dos tipos de dados complexos, incluindo os dados espaciais e diversos outros. Com a emergência do suporte a **dados multimídia** em SGBDs relacionais, os operadores por similaridade vêm despertando muito interesse, principalmente para a recuperação por conteúdo de dados complexos de diversos tipos.

Para os operadores de consulta por similaridade serem aplicáveis a um determinado domínio de elementos $\mathbb{S}$, é necessário que esteja definida no domínio uma **função de similaridade** d, também chamada de **função de distância** ou **métrica**. Uma função

de distância quantifica quão similares dois elementos são e habilita a representação de consultas baseadas na similaridade dos elementos. Neste trabalho, são usadas as funções de distância que atendam às propriedades de simetria, não-negatividade e desigualdade triangular. As funções de distância que atendem a essas três propriedades associadas a um domínio de dados criam o que se denomina de **espaço métrico**.

Tais como as consultas baseadas em ROT, as consultas por similaridade são baseadas em critérios de comparação, chamados de "**critérios de similaridade**" ou "condições de similaridade". Os principais critérios de similaridade usados são os **unários** ou os **binários**. Os critérios unários, utilizados em operações de **seleção por similaridade**, comparam elementos de apenas um conjunto de dados $T \subseteq \mathbb{S}$ com um ou mais elementos centrais de consulta $s_q \in \mathbb{S}$, dados como parte do predicado da consulta. Existem basicamente dois tipos de seleção por similaridade: **similaridade por abrangência** (*Similarity Range query*) e **similaridade aos k vizinhos mais próximos** (k-*Nearest Neighbor query*).

Os critérios binários, utilizados em operações de **junção por similaridade**, comparam elementos de dois conjuntos de dados $T_1, T_2 \subseteq \mathbb{S}$ e, nesse caso, o elemento central de consulta s_q não existe. Há basicamente três tipos de junção por similaridade (Böhm e Krebs, 2002): **junção por abrangência** (*Range join*), **junção por proximidade** (k-*Closest Neighbor join*) e **junção por vizinhança** (k-*Nearest Neighbor join*).

Embora recentemente tenha havido muitos trabalhos direcionados ao desenvolvimento de algoritmos e estruturas de indexação para a execução de consultas por similaridade, principalmente para os operadores unários, as operações de consulta por similaridade não são suportadas pela **álgebra relacional**. No GBdI-ICMC-USP, tem-se trabalhado para estender a álgebra relacional incluindo maneiras de representar essas consultas com o mínimo de impacto nesta álgebra. Portanto, esta dissertação pretende contribuir com a incorporação das consultas por similaridade nos SGBDs relacionais, propondo a **álgebra por similaridade** e suas regras algébricas para os operadores unários por similaridade.

1.2 Definição do problema

Para que um SGBD relacional dê suporte a dados complexos, precisa responder eficientemente a consultas por similaridade. Uma das maneiras para que este suporte seja oferecido é incluir as consultas por similaridade na álgebra relacional e fornecer as maneiras para processar eficientemente as consultas por similaridade em cada uma das etapas (compilação, otimização e execução) do **processamento de consultas** do SGBD relacional.

Com este intuito, Barioni et al. (2006) propuseram um mecanismo, denominado SIREN (*SImilarity Retrieval ENgine*), que permite a realização de consultas por similaridade em SQL. O SIREN é um serviço, implementado entre a aplicação e o SGBD relacional,

que intercepta todo comando enviado pela aplicação, analisando e tratando todas as construções por similaridade e referências a elementos complexos. Possui, como seu principal componente, o compilador de consultas. No entanto, como não existe um otimizador de consultas em sua estrutura, o SIREN não processa eficientemente este tipo de consulta.

Para que o SIREN processe as consultas por similaridade de forma eficaz, neste trabalho de mestrado, o otimizador de consultas foi implementado no SIREN e, para que ele funcione de maneira semelhante ao otimizador do SGBD relacional, a álgebra por similaridade proposta neste trabalho foi nele incorporada. Assim, esta dissertação tem como objetivo apresentar a álgebra por similaridade e regras algébricas para processar as consultas unárias por similaridade e o otimizador de consultas do SIREN.

1.3 Organização do trabalho

Esta dissertação está organizada da seguinte maneira:

Capítulo 2 - Processamento de Consultas. Apresenta definições e conceitos sobre o processamento de consultas em SGBDs relacionais, dando enfoque principal para o otimizador de consultas.

Capítulo 3 - Consultas por Similaridade. São apresentados conceitos básicos de consultas por similaridade, formalizando os tipos de consultas por similaridade: similaridade por abrangência, similaridade aos k-vizinhos mais próximos, junção por abrangência, junção por proximidade e junção por vizinhança. Além disso, são apresentados os algoritmos para as consultas por similaridade e algumas álgebras que tratam a noção de similaridade.

Capítulo 4 - Álgebra por Similaridade. A Álgebra por Similaridade para os operadores unários proposta neste trabalho é apresentada. O modelo de dados, a álgebra por similaridade e as regras de transformação algébrica são definidos para os operadores unários por similaridade.

Capítulo 5 - Otimizador de Consultas por Similaridade. O otimizador de consultas por similaridade desenvolvido neste trabalho é apresentado, seu funcionamento é detalhado e um exemplo de utilização é mostrado.

Capítulo 6 - Conclusão. Encerra esta dissertação, apresentando as considerações finais, as principais contribuições e as propostas para trabalhos futuros.

Referências Bibliográficas.

Capítulo A - Apêndice. O mecanismo SIREN é descrito sucintamente.

Capítulo

2

Processamento de consultas

2.1 Introdução

O Sistema de Gerenciamento de Bases de Dados (SGBD) relacional usa a linguagem SQL (*Structured Query Language*) para representar as operações de armazenamento e recuperação de dados. Para um bom desempenho das operações de recuperação de dados, o SGBD precisa estar apto a produzir, automaticamente, alternativas de execução para uma determinada consulta e, dentre elas, escolher a que possui melhor desempenho. O processo de produção de alternativas de execução e escolha da melhor alternativa é realizado no processador de consultas do SGBD relacional.

O **processador de consultas** é o grupo de componentes do SGBD que transforma uma consulta SQL em uma seqüência de operações sobre o banco de dados, gera alternativas de execução para a consulta e executa a alternativa com o melhor desempenho entre elas, isto é, com o menor custo computacional. Como ilustrado na Figura 2.1, o processador de consultas é composto pelo compilador, otimizador e executor. O **compilador** analisa uma consulta expressa em SQL para saber se ela é válida; o **otimizador** gera as alternativas de execução, examina-as e escolhe o plano de consulta com o menor custo computacional previsto para ser executado; e o **executor** executa o plano de consulta escolhido e produz a resposta para a consulta SQL.

A importância do processador de consultas é um consenso entre os pesquisadores, mas os detalhes de cada camada e os módulos da arquitetura do processador de consultas não são. Portanto, a arquitetura para as etapas do processamento de consultas pode diferir quanto à nomenclatura dos módulos e à divisão das camadas.

A arquitetura do otimizador adotada neste trabalho de mestrado é a baseada em reescrita de consultas. Esse tipo de otimização é apoiada no fato de que diversas expressões

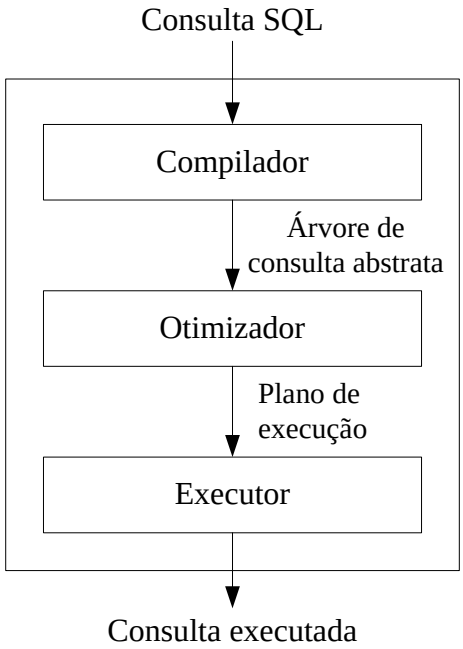

Figura 2.1: Arquitetura do processador de consultas de um SGBD relacional.

algébricas possuem resultados equivalentes, mas cada expressão tem um custo de execução diferente. O objetivo dessa forma de otimização é identificar uma expressão algébrica equivalente que pode ser executada com baixo custo computacional.

A arquitetura do processador de consultas de um SGBD relacional, bem como uma visão geral do seu funcionamento são apresentadas na Seção 2.2; em seguida, na Seção 2.3, é apresentada uma visão mais detalhada do funcionamento do otimizador de consultas do SGBD relacional; e na Seção 2.4 são apresentadas as considerações finais. As definições e propriedades apresentadas neste capítulo são baseadas nos trabalhos de Yu e Meng (2002), Garcia-Molina et al. (2000), Elmasri e Navathe (2005), Ioannidis (1996) e Arantes (2005).

2.2 Arquitetura de um SGBD relacional

A Figura 2.2 mostra o esquema relacional usado para exemplificar o funcionamento do processador de consultas de um SGBD relacional. Este esquema também será utilizado em todos os exemplos desta dissertação. Esta base de dados foi utilizada pois ela é, ao mesmo tempo, intuitiva (facilita o entendimento dos exemplos) e abrangente, permitindo que todos os exemplos a utilizem. Além disso, dados reais foram obtidos para carregá-la e, com isso, ela também pode ser usada para executar os exemplos aplicados em situações reais.

A relação `CidadeBR` contém 5507 tuplas com informações a respeito das posições geográficas das cidades brasileiras (IBGE, 2006). A Tabela 2.1 apresenta os atributos que compõem cada tupla da relação `CidadeBR`. Os atributos `latitude` e `longitude` são

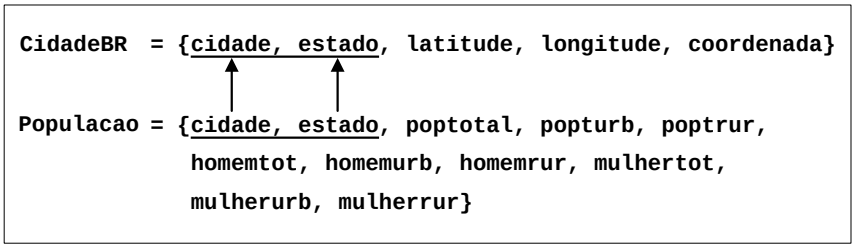

Figura 2.2: Esquema Relacional.

usados para calcular a distância entre cada par de elementos complexos ao aplicar a função Euclidiana (L_2) sobre suas coordenadas construindo, assim, o atributo complexo `coordenada`. Este atributo complexo será utilizado nas consultas por similaridade apresentadas a partir do Capítulo 3.

Atributo	Tipo	Descrição
`cidade`	Cadeia de caracteres	Nome da cidade
`estado`	Cadeia de caracteres	Sigla do Estado brasileiro
`latitude`	Numérico	Latitude
`longitude`	Numérico	Longitude
`coordenada`	Complexo	Posição geográfica da cidade

Tabela 2.1: Descrição dos atributos da relação `CidadeBR`.

A relação `Populacao` contém 5507 tuplas com informações a respeito da população das cidades brasileiras (IBGE, 2006). A Tabela 2.2 apresenta os atributos que compõem cada tupla da relação `Populacao`.

Atributo	Tipo	Descrição
`cidade`	Cadeia de caracteres	Nome da cidade
`estado`	Cadeia de caracteres	Sigla do Estado brasileiro
`poptotal`	Numérico	População total
`popturb`	Numérico	População total urbana
`poptrur`	Numérico	População total rural
`homemtot`	Numérico	População masculina total
`homemurb`	Numérico	População masculina urbana
`homemrur`	Numérico	População masculina rural
`mulhertot`	Numérico	População feminina total
`mulherurb`	Numérico	População feminina urbana
`mulherrur`	Numérico	População feminina rural

Tabela 2.2: Descrição dos atributos da relação `Populacao`.

Um exemplo de consulta utilizando o esquema relacional mostrado na Figura 2.2 é:

Q1: "*Selecione a cidade, a população masculina e feminina urbana e rural das cidades do estado de São Paulo com a latitude menor ou igual a -22.02, a longitude menor ou igual a 47.89 e a população total maior ou igual a 100000 habitantes.*"

Esta consulta pode ser expressa em SQL por:

```
SELECT cidade, homemurb, homemrur,
       mulherurb, mulherrur
  FROM CidadeBR, Populacao
 WHERE CidadeBR.cidade = Populacao.cidade
   AND CidadeBR.estado = Populacao.estado
   AND Populacao.estado = 'SP'
   AND latitude <= -22.02
   AND longitude <= 47.89
   AND poptotal >= 100000
```

Quando um SGBD relacional recebe uma consulta em SQL, esta é compilada, otimizada e, então, executada. Para que uma consulta tradicional, expressa em SQL, seja considerada válida, o **compilador** faz a análise léxica, sintática e semântica. Na **análise léxica**, o compilador verifica os itens léxicos (palavras-chave ou *tokens*) da linguagem SQL; na **análise sintática**, verifica a sintaxe da consulta para determinar se ela obedece às regras da gramática da linguagem; e, na **análise semântica**, verifica se as relações ou visões usadas pertencem ao esquema sobre o qual a consulta está sendo executada, verifica se os atributos usados pertencem a alguma relação do esquema e se eles possuem o tipo apropriado para o seu uso.

Após o compilador realizar todas as análises com êxito, a consulta é considerada válida e a **árvore de consulta abstrata** é gerada. A Figura 2.3 ilustra a árvore gerada para a Consulta **Q1**. Por se tratar de um exemplo, a gramática que gerou a árvore da Figura 2.3 foi omitida.

A árvore de consulta abstrata é utilizada como entrada no otimizador para que a **expressão canônica** em álgebra relacional possa ser gerada, representando o plano de consulta inicial ainda sem sofrer otimizações, conforme mostrado na Figura 2.4.

A expressão canônica, o plano de consulta inicial, os planos de consulta alternativos, os planos equivalentes e o plano de consulta ótimo são representados em forma de árvore de consulta, onde as relações de entrada são representadas como nós folha e as operações da álgebra relacional são representadas como nós internos. Um exemplo da representação de uma consulta em forma de árvore de consulta é apresentado na Figura 2.4.

Uma execução da árvore de consulta consiste na execução de uma operação do nó interno sempre que seus operandos estiverem disponíveis e depois da substituição do

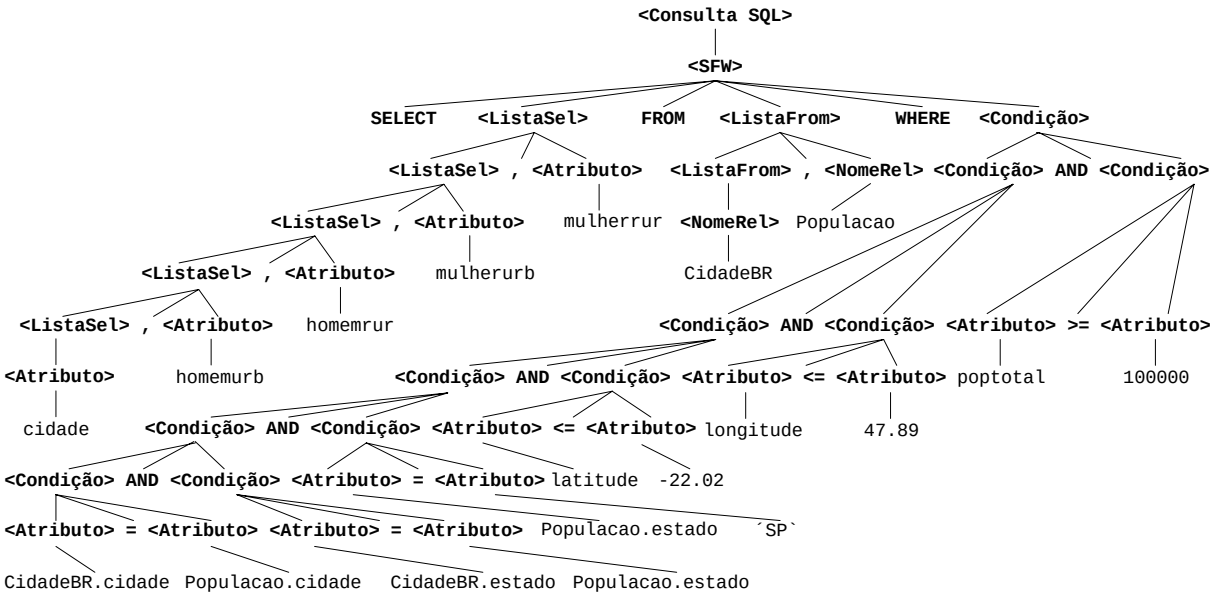

Figura 2.3: Árvore de consulta abstrata gerada a partir da Consulta **Q1**.

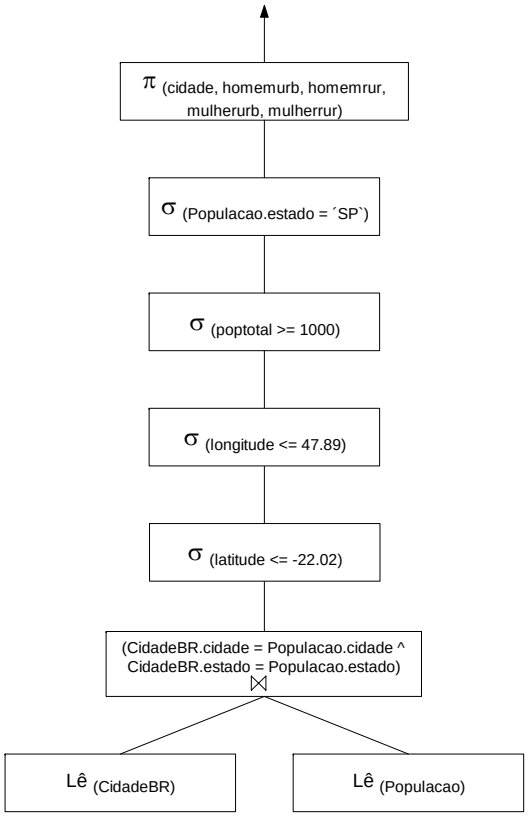

Figura 2.4: Plano de consulta inicial construído a partir da árvore de consulta abstrata da Figura 2.3 (expressão em forma canônica).

nó interno pela relação que resulta da execução da operação (execução *bottom-up*). A execução termina quando o nó raiz é executado e produz a relação resultado da consulta.

No **otimizador**, regras de equivalência entre expressões da álgebra relacional e heurísticas de otimização são aplicadas na árvore de consulta inicial para a geração de planos de consulta alternativos. Para cada plano de consulta inicial existem diferentes planos de consulta que produzem o mesmo resultado. Planos de consulta que sempre produzem o mesmo resultado são chamados de **equivalentes**. Porém, diferentes planos equivalentes são avaliados com custos distintos. O objetivo do otimizador é encontrar um plano de consulta, dentre todos os possíveis planos equivalentes, que pode ser executado com o menor custo computacional, ou seja, com o menor tempo de execução. Este plano de consulta é chamado de **plano de consulta ótimo** e é utilizado como entrada do executor.

Para ilustrar o método de geração de planos de consulta alternativos, o plano de consulta inicial mostrado na Figura 2.4 é repetido na Figura 2.5a. Regras de equivalência e heurísticas de otimização foram aplicadas sobre o plano de consulta inicial e um plano de consulta alternativo foi gerado, conforme mostrado na Figura 2.5b. Por se tratar de um exemplo ilustrativo, está mostrado somente um plano alternativo, apesar desta consulta gerar diversos planos equivalentes com diferentes custos computacionais. As regras de equivalência e as heurísticas de otimização utilizadas para a geração do plano alternativo da Figura 2.5b são apresentadas na Seção 2.3.

No **executor**, índices e tabelas são acessados e a resposta para a consulta tradicional expressa em SQL é retornada.

Na Seção 2.3, o otimizador de consultas é estudado mais profundamente, pois um dos resultados deste trabalho foi o desenvolvimento um otimizador de consultas por similaridade baseado no otimizador de consultas do SGBD relacional. O otimizador de consultas por similaridade desenvolvido é apresentado no Capítulo 5.

2.3 Otimizador de consultas

O **otimizador de consultas** é a camada do processador de consultas do SGBD relacional responsável por: gerar planos de consulta alternativos para uma determinada consulta que utiliza leis algébricas; calcular o custo de cada plano gerado; e escolher, dentre eles, o que possui o menor custo computacional (Ramakrishnan e Gehrke, 2003). Todos os planos de consulta alternativos gerados pelo otimizador são equivalentes se considerarmos a sua saída, mas variam em relação ao custo computacional (Ioannidis, 1996). Portanto, o objetivo do otimizador de consultas é encontrar um plano de consulta, dentre todos os possíveis planos equivalentes, que possui o melhor desempenho, isto é, o menor custo computacional. Tal plano é chamado de **plano de consulta ótimo**.

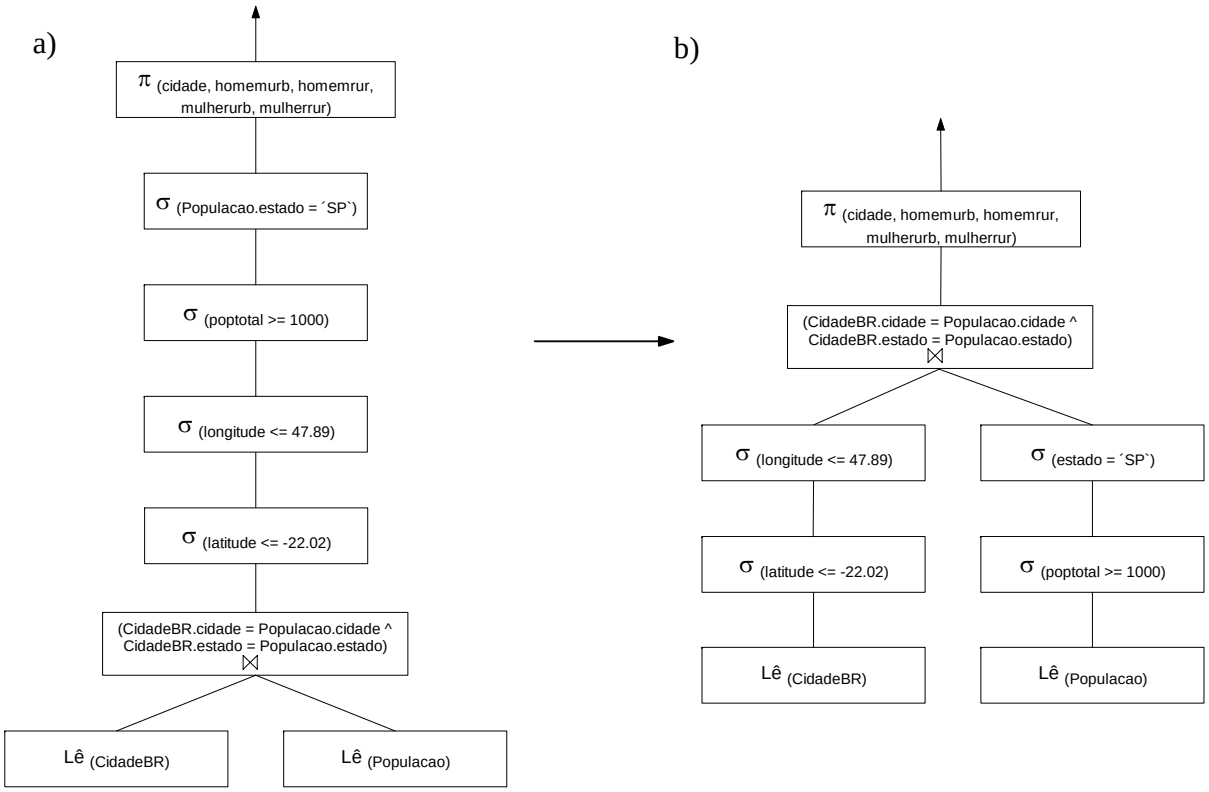

Figura 2.5: Plano de consulta alternativo antecipando as seleções na árvore de consulta. (a)Plano de consulta inicial; (b) plano de consulta alternativo.

Segundo Ioannidis (1996) e Garcia-Molina et al. (2000), o otimizador pode ser dividido em: Gerador de Planos Lógico e Gerador de Planos Físico, conforme mostrado na Figura 2.6.

O **Gerador de Planos Lógico** é o responsável pela aplicação das transformações na árvore de consulta inicial e a produção das árvores de consulta equivalentes com a intenção de serem mais eficientes. Para isto, o Gerador de Planos Lógico utiliza o Espaço Algébrico e o Espaço Estrutura-Método. O **Espaço Algébrico** determina a ordem das operações a ser considerada pelo Gerador de Planos Lógico para cada consulta e o **Espaço Estrutura-Método** escolhe a implementação existente para executar a ordem das operações especificada pelo Espaço Algébrico (Ioannidis, 1996).

Uma árvore de consulta define uma seqüência de passos que são utilizados pelo Gerador de Planos Físico para a avaliação de uma consulta. Cada passo da seqüência corresponde a uma operação relacional mais o método ou implementação existente para executar e avaliar esta operação. Para cada operação relacional, há um número de métodos que podem ser usados para avaliá-lo. Em geral, o número de planos de consulta equivalentes para uma dada consulta é determinado pelo número de operações na consulta e pelo número de métodos que podem ser usados para avaliar cada operação. Supondo h como o número de operações em uma consulta e que cada operação pode ser avaliada de g diferentes maneiras, então é possível obter $(h!) * g^h$ diferentes planos de consulta (Yu e

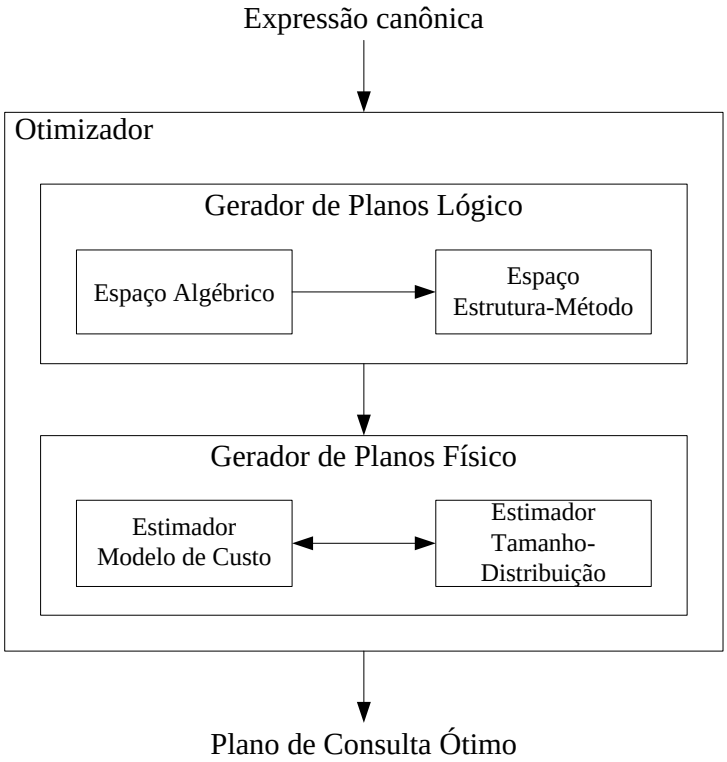

Figura 2.6: Arquitetura do otimizador de consultas segundo Ioannidis (1996) e Garcia-Molina et al. (2000).

Meng, 2002). Por exemplo: seja a consulta **Q1**, a operação de seleção pode ser avaliada por 4 métodos, a projeção pode ser avaliada por 2 métodos e a junção por 3 métodos, então temos $h = 3$ e $g = 9$ e é possível obter $(3!) * 9^3 = 4374$ diferentes planos de consulta. O conjunto de todos os planos de consulta equivalentes é o **espaço de busca** do otimizador de consultas.

O **Gerador de Planos Físico** emprega uma estratégia de busca que explora o espaço de busca determinado pelo Espaço Algébrico e pelo Espaço Estrutura-Método para cada plano de consulta produzido pelo Gerador de Planos Lógico. Ele compara estes planos baseado nas estimativas de seus custos derivadas do Estimador Modelo de Custo e do Estimador Tamanho-Distribuição e seleciona, dentre todos, a árvore de consulta com o menor custo computacional para ser usada na geração da resposta à consulta original. O **Estimador Modelo de Custo** especifica as fórmulas usadas para estimar os custos das árvores de consulta e o **Estimador Tamanho-Distribuição** estima o tamanho dos resultados das consultas (ou subconsultas) e a distribuição das freqüências de valores dos atributos destes resultados (estatísticas), os quais são utilizados pelo Estimador Modelo de Custo (Ioannidis, 1996).

Uma consulta SQL consiste em um conjunto de operações relacionais. A ordem das operações tem um impacto significativo no custo de avaliação da consulta. Há duas técnicas principais para determinar a ordem de execução das operações: a técnica baseada em álgebra, que usa um conjunto de regras heurísticas para ajudar na transformação de

um plano de consulta em outro; e a técnica baseada na avaliação do custo, que avalia, para cada consulta, o custo de todos os planos de consulta possíveis e escolhe o plano de consulta com o menor custo estimado. Ambas as técnicas utilizam um conjunto de regras que podem transformar um plano de consulta em outro equivalente, representados como uma expressão em álgebra relacional.

Na literatura, há diversas regras que transformam uma expressão de álgebra relacional em uma expressão equivalente, isto é, em uma expressão que produz o mesmo resultado. As regras de transformação mais comumente utilizadas na otimização de consultas são descritas a seguir. Como um dos resultados deste trabalho foi o desenvolvimento de um otimizador de consultas por similaridade, as regras de transformação aplicadas à seleção descritas nesta seção foram adaptadas, no Capítulo 4, para funcionar também com as consultas por similaridade.

Supondo que U, V e W são três relações, então:

1. **Seleções em cascata:** Seja c_1 e c_2 duas condições de seleção sobre U. Então,

$$\sigma_{c_1 \wedge c_2}(U) \;=\; \sigma_{c_1}(U) \cap \sigma_{c_2}(U) \;=\; \sigma_{c_1}(\sigma_{c_2}(U)) \;=\; \sigma_{c_2}(\sigma_{c_1}(U)). \tag{2.1}$$

2. **Comutatividade da seleção com a junção:** Se a condição c envolve somente atributos de U então,

$$\sigma_c(U \bowtie V) \;=\; \sigma_c(U) \bowtie V; \tag{2.2}$$

   se a condição c envolve somente atributos de V então,

$$\sigma_c(U \bowtie V) \;=\; U \bowtie \sigma_c(V); \tag{2.3}$$

   se a condição c_1 tem apenas atributos de U e a condição c_2 tem apenas atributos de V, as regra de transformação 2.1, 2.2 e 2.3 podem ser utilizadas para deduzir que:

$$\sigma_{(c_1 \wedge c_2)}(U \bowtie V) \;=\; \sigma_{c_1}(U) \bowtie \sigma_{c_2}(V). \tag{2.4}$$

   Esta regra, bem como as regras de transformação 2.5, 2.6, 2.7 e 2.8, também podem ser aplicadas ao produto cartesiano das relações U e V ($U \times V$).

3. **Comutatividade entre a projeção e a junção:** Se assumirmos que os atributos projetados são $AL = \{\alpha_1, \ldots, \alpha_n, \beta_1, \ldots, \beta_m\}$, onde $\alpha's$ são atributos de U e $\beta's$ são atributos de V, se a condição de junção c envolve atributos somente em AL, então

$$\pi_{AL}(U \overset{c}{\bowtie} V) \;=\; (\pi_{\alpha_1,\ldots,\alpha_n}(U)) \overset{c}{\bowtie} (\pi_{\beta_1,\ldots,\beta_m}(V)); \tag{2.5}$$

se, além dos atributos em AL, c também envolve os atributos $\alpha'_1, \ldots, \alpha'_u$ de U e os atributos $\beta'_1, \ldots, \beta'_v$ de V, então

$$\pi_{AL}(U \overset{c}{\bowtie} V) = \pi_{AL}((\pi_{\alpha_1,\ldots,\alpha_n,\alpha'_1,\ldots,\alpha'_u}(U)) \overset{c}{\bowtie} (\pi_{\beta_1,\ldots,\beta_m,\beta'_1,\ldots,\beta'_v}(V))). \tag{2.6}$$

4. **Associatividade da junção–θ e da junção natural:** As operações de junção–θ e junção natural não podem ser colocadas em uma mesma expressão, pois se representadas em uma mesma regra, retornam respostas incorretas, isto é, $U \overset{c}{\bowtie} (V \bowtie W) \neq (U \overset{c}{\bowtie} V) \bowtie W$. No entanto:

$$U \overset{c_1}{\bowtie} (V \overset{c_2}{\bowtie} W) = (U \overset{c_1}{\bowtie} V) \overset{c_2}{\bowtie} W; \tag{2.7}$$

$$U \bowtie (V \bowtie W) = (U \bowtie V) \bowtie W. \tag{2.8}$$

5. **Troca σ e $\times$ por $\bowtie$:** Se a condição de seleção c da forma "$U.\alpha\ op\ V.\beta$", no qual "α" é um atributo de U, "β" é um atributo de V e "op" é um operador de comparação ou relacional ($=$, $\neq$, $<$, $\leq$, $>$ ou $\geq$), for seguida pelo produto cartesiano de U e V então,

$$\sigma_c(U \times V) = U \overset{c}{\bowtie} V \tag{2.9}$$

A **técnica de otimização baseada em álgebra** tem como objetivo representar cada consulta relacional como uma expressão algébrica e, então, transformá-la em uma expressão algébrica equivalente, que pode ser executada de maneira mais eficiente. Esta transformação é guiada por regras heurísticas de otimização. Estas regras heurísticas usam as regras de transformação 2.1 a 2.9 para converter uma árvore de consulta inicial em uma árvore de consulta que, na maioria das vezes, é a mais eficiente para ser executada, ou seja, uma árvore de consulta otimizada. As heurísticas de otimização mais comumente utilizadas são:

1. Executar as seleções o mais cedo possível, pois elas podem reduzir consideravelmente o tamanho das relações. As seleções com condições conjuntivas, isto é, conjunções conectadas $\mathtt{E}(\wedge)$, podem ser quebradas em uma cascata de operações de seleção ao usar a regra de transformação 2.1, permitindo assim um grau maior de liberdade para distribuir operações de seleção para ramos diferentes e para níveis inferiores no plano de consulta, ou seja, as operações de seleção podem ser empurradas ou puxadas para baixo no plano de consulta. Estas seleções conjuntivas separadas em seleções individuais também podem ser distribuídas sobre junções e produtos cartesianos se forem usadas as regras de transformação 2.2, 2.3 e 2.4.

2. Trocar os produtos cartesianos por junções, sempre que possível. Um produto cartesiano entre duas relações é usualmente muito mais caro do que uma junção entre duas relações. Isto ocorre pois o primeiro executa exaustivamente a concatenação de pares de tuplas e pode gerar um resultado muito grande. A troca de produtos cartesianos por junções pode ser realizada se houver uma combinação de uma operação de seleção com uma operação de produto cartesiano subseqüentes no plano de consulta e se a condição de seleção representar uma condição de junção. A regra de transformação 2.9 pode ser usada para realizar esta troca.

3. Se existirem muitas junções, executar primeiro as mais restritivas. Uma junção é mais restritiva do que outra se resulta em um resultado menor, isto é, se produz uma relação com menor número de tuplas. Algebricamente, esta transformação pode ser realizada usando as regras de transformação 2.7 e 2.8.

4. Retirar os atributos desnecessários, o mais cedo possível. Se um atributo de uma relação não é necessário para futuras operações, pode ser removido se usar a operação de projeção com o objetivo de utilizar relações de entrada menores em operações futuras. Assim, somente aqueles atributos necessários no resultado da consulta e nas operações subseqüentes no plano de consulta serão mantidos após cada operação de projeção. Se utilizar as regras de transformação 2.5 e 2.6, quebram-se e transferem-se as listas de atributos da operação de projeção para níveis inferiores no plano de consulta, ou seja, as operações de projeção também podem ser distribuídas (empurradas ou puxadas para baixo) no plano de consulta.

A **técnica de otimização baseada em avaliação de custo** tem como objetivo escolher, dentre todos os possíveis planos de consulta, aquele que possui o menor custo estimado. Estimar o custo de avaliação de um plano de consulta em um SGBD relacional envolve a soma de dois componentes: o custo de acesso à memória secundária (custo de E/S) e o custo de computação (custo de CPU). O **custo de E/S** é determinado pela transferência de dados entre a memória principal e o armazenamento secundário. O **custo de CPU** é determinado pela execução de operações em memória sobre os dados durante a execução da consulta. Para muitas operações, incluindo as operações de seleção, projeção e junção, o custo de E/S é o custo dominante. Algumas estruturas de indexação, como a árvore B^+, são utilizadas para reduzir este custo.

Esta técnica de otimização funciona da seguinte maneira: (1) para cada consulta, enumere todos os possíveis planos de consulta; (2) para cada plano de consulta, estime o custo do plano de consulta; (3) escolha o plano de consulta com o menor custo estimado. Se o custo de cada plano de consulta for precisamente estimado, então um plano de consulta ótimo pode finalmente ser encontrado. Porém, ao utilizar esta técnica, duas dificuldades podem ser encontradas:

1. o número de possíveis planos de execução é uma função exponencial do número de relações referenciadas em uma consulta. Com isto, existem muitos planos de execução para serem enumerados;

2. estimar o custo de cada plano de execução de modo preciso pode ser difícil, pois é necessário estimar precisamente o tamanho dos resultados intermediários. Por exemplo, em um plano de execução mais elaborado, o resultado de uma operação (Op_1) pode ser usado como entrada para outra operação (Op_2). Para estimar o custo de Op_2, é necessário estimar o tamanho do resultado de Op_1 primeiro.

A primeira dificuldade pode ser tratada se enumerarmos apenas um subconjunto de todos os possíveis planos de execução ao invés de enumerar todos os possíveis, que é realizada por meio de heurísticas. Já a segunda dificuldade requer a utilização das informações sobre os dados armazenados, as chamadas **estatísticas**, para estimar a seletividade das diversas condições usadas para expressar uma consulta e o custo de cada operação de acesso à base de dados que precisa ser executada para responder à consulta. Os métodos propostos para tratá-la podem ser classificados em: (1) amostragem, a qual estima os tamanhos dos resultados intermediários baseada em informações coletadas de uma pequena fração de instâncias da relação; (2) histograma, o qual usa informações detalhadas pré-armazenadas sobre as relações para estimar os tamanhos dos resultados intermediários; (3) paramétrico, o qual usa técnicas analíticas e/ou estatísticas para estimar os tamanhos dos resultados intermediários, ao fazer suposições sobre a distribuição dos valores dos dados e sobre a correlação entre os valores de diferentes atributos.

2.4 Considerações finais

Neste capítulo, foi apresentado sucintamente o funcionamento do processador de consultas e com mais detalhes o funcionamento do otimizador de consultas de um SGBD relacional.

Toda consulta expressa em SQL passa por um processo automático que é feito pelo SGBD em etapas. As etapas do processamento de consultas são: Compilador, Otimizador e Executor.

A idéia geral é que, dada uma determinada consulta em SQL, ela é analisada pelo compilador e, então, uma árvore de consulta abstrata é gerada. Esta árvore de consulta abstrata é submetida ao Gerador de Planos Lógico do otimizador de consultas para produzir vários planos alternativos algebricamente equivalentes. Os planos equivalentes são submetidos ao Gerador de Planos Físico, onde seus custos são calculados e comparados. O plano com o menor custo computacional é enviado ao executor. Este plano é executado e a resposta à consulta em SQL é retornada.

O processo de geração de planos equivalentes e a escolha do melhor plano a ser executado é realizado no otimizador de consulta e, para realizar este processo, ele utiliza as regras algébricas e as heurísticas de otimização de consulta apresentadas na Seção 2.3.

Para dados tradicionais, o processamento de consultas está bem sedimentado, apesar de não haver um consenso entre os pesquisadores sobre os módulos da arquitetura do processador e os detalhes das camadas. No entanto, para que um SGBD consiga processar dados complexos, o otimizador de consultas deve estar adaptado a reescrever as consultas por similaridade. A álgebra por similaridade e as regras de transformação são uma contribuição deste trabalho de mestrado, como apresentado no Capítulo 4.

Capítulo

3

Consultas por Similaridade

3.1 Introdução

Em contraste com as consultas tradicionais que utilizam os operadores de igualdade relacionais para manipular os dados tradicionais em SGBDs relacionais, as consultas por similaridade procuram em um conjunto por elementos que, segundo algum critério de similaridade, sejam mais "parecidos com" ou mais "distintos de" um dado elemento de consulta; isto é, as consultas por similaridade comparam todos os elementos do conjunto a um elemento de consulta e selecionam apenas aqueles que atendam a um certo critério de similaridade.

Para determinar o quão similares os elementos são, eles podem ser comparados diretamente ou a partir de vetores de características deles extraídos. Nos dois casos, uma função d, que calcula a distância entre dois elementos deve ser definida. Essa função d é chamada de **função de dissimilaridade** ou **função de distância** e, por definição, d retorna um valor real sempre maior ou igual a zero: tem valores próximos de zero para elementos muito similares, valores iguais a zero para elementos idênticos e valores maiores para elementos menos similares. Na realidade, a função de distância indica a dissimilaridade entre dois elementos e convém colocar que a similaridade é medida pelo seu inverso. Muitos autores estabelecem que a função de distância indica a similaridade, já que esta pode ser obtida pela dissimilaridade. A função de distância utilizada para calcular o grau de similaridade entre os elementos usualmente é definida por um especialista no domínio em questão.

Na Seção 3.2 são detalhados os tipos de consultas por similaridade. Alguns métodos de acesso métricos (MAMs), isto é, algumas estruturas de indexação para dados em domínios métricos, são apresentados na Seção 3.3. Os algoritmos para responder às consultas por

similaridade são discutidos na Seção 3.4. Na Seção 3.5 são apresentadas algumas propostas de álgebras que lidam com consultas por similaridade e na Seção 3.6 são apresentadas as considerações finais.

3.2 Consultas por similaridade

Seja $\mathbb{S}$ um domínio de elementos. A função $d : \mathbb{S} \times \mathbb{S} \rightarrow \mathbb{R}^+$ corresponde à medida de **distância** ou **dissimilaridade** entre dois elementos, isto é, quanto menor a distância, mais próximos ou similares os elementos são. Se uma função de distância possuir as propriedades de:

- simetria: $d(s_1, s_2) = d(s_2, s_1)$,
- não-negatividade: $0 < d(s_1, s_2) < \infty$ se $s_1 \neq s_2$ e $d(s_1, s_1) = 0$,
- desigualdade triangular: $d(s_1, s_2) \leq d(s_1, s_3) + d(s_3, s_2)$, $\forall\, s_1, s_2, s_3 \in \mathbb{S}$,

onde $\{s_1, s_2, s_3\} \in \mathbb{S}$, o espaço M definido pelo par $< \mathbb{S}, d >$ é chamado de **Espaço Métrico** e a função de distância d também pode ser chamada de **Métrica** (Burkhard e Keller, 1973). O subconjunto finito T, no qual as consultas são efetuadas, está no espaço métrico se $T \subseteq \mathbb{S}$.

Tais como as consultas baseadas em ROT, as consultas por similaridade são baseadas em critérios de comparação, chamados de "critérios de similaridade" ou "condições de similaridade". Os principais critérios de similaridade usados são os unários e os binários.

Relações que incorporam atributos que podem ser avaliados para consultas por similaridade devem seguir as mesmas regras e definições das relações tradicionais. Portanto, utilizamos a seguinte notação para expressar relações que incluem atributos que podem ser avaliados por similaridade. Seja A um atributo qualquer amostrado em um domínio $\mathbb{A}$ que atende à ROT, S um atributo qualquer amostrado no domínio $\mathbb{S}$ de um espaço métrico e T uma relação que inclui qualquer número de atributos de qualquer domínio, isto é, dado o esquema de uma relação $\mathbb{T} = \{\mathbb{A}_1, \ldots, \mathbb{A}_j, \mathbb{S}_1, \ldots, \mathbb{S}_n\}$, uma relação $T = \{A_1, \ldots, A_j, S_1, \ldots, S_n\}$ tem cada tupla $t = \langle a_1, \ldots, a_j, s_1, \ldots, s_n \rangle$ com seus valores amostrados nos domínios dos respectivos atributos. Seja $t_i(atributo)$ o valor da i-ésima tupla t_i no atributo. Portanto, $t_i(S)$ é o valor da i-ésima tupla no atributo S, isto é, é o elemento s_i. Estas notações são utilizadas nas definições das consultas por similaridade unárias e binárias e, no Capítulo 4, para a definição da álgebra por similaridade.

3.2.1 Unários

Os operadores por similaridade unários são utilizados em operações de **seleção por similaridade** e comparam os elementos do conjunto de dados $T \subseteq \mathbb{S}$ com um ou mais

elementos $s_q \in \mathbb{S}$, em que s_q é chamado de elemento central de consulta ou elemento de consulta, dados como parte do predicado da consulta (Wang e Shasha, 1990). Então, uma operação de seleção por similaridade sobre um atributo complexo $S \in T$ pode ser representada, algebricamente, como:

$$\hat{\sigma}_{(S\ Op_s(d,lim)\ s_q)}\ T,$$

em que $\langle S\ Op_s(d, lim)\ s_q\rangle$ é o **predicado de seleção por similaridade**, S é um dos atributos da relação T amostrados em domínios $\mathbb{S}$ de espaço métricos, $s_q \in \mathbb{S}$ é o elemento de consulta e $\langle Op_s(d, lim)\rangle$ é operador unário de seleção por similaridade que utiliza a função de distância d e o limite (ou *threshold*) de dissimilaridade lim para obter o resultado da consulta de seleção por similaridade. Como as propriedades da seleção por similaridade não são as mesmas das operações de seleção da álgebra relacional (representada por σ), a seleção por similaridade é representada, nesta dissertação, pelo símbolo $\hat{\sigma}$.

Há dois tipos de operadores unários por similaridade que são descritos a seguir. Se o elemento de consulta s_q pertencer ao conjunto de valores presentes na relação T ($s_q \in T$), então s_q faz parte do subconjunto resposta T' ($s_q \in T'$); caso contrário, s_q não pertence ao subconjunto resposta T' ($s_q \notin T'$).

Similaridade por Abrangência (*Similarity Range Query*) - R_q: dados o atributo S sobre o qual é expressa a condição de similaridade, a função de distância d, o limite de dissimilaridade ξ e o elemento de consulta $s_q \in \mathbb{S}$, a consulta $\hat{\sigma}_{(S\ R_q(d,\xi)\ s_q)}T$ recupera todas as tuplas $t_i \in T$ tal que $d(t_i(S), s_q) \leq \xi$. O subconjunto resposta resultante é $T' = \{t_i \in T \mid d(t_i(S), s_q) \leq \xi\}$.

Um exemplo de uma consulta R_q sobre a relação `CidadeBR`, definida no Capítulo 2, é: "Selecione as cidades brasileiras que distam da cidade de "São Carlos-SP", cuja latitude é igual a -22.02 e a longitude é igual a 47.89, até no máximo 0.3 unidades de distância, considerando a distância Euclidiana L_2". A relação T é o conjunto das cidades brasileiras, s_q é a cidade de "São Carlos-SP" ($s_q \in T$), $\xi = 0.3$ é o limite de dissimilaridade, a métrica $d = L_2$ e o subconjunto resultante T' é o conjunto das cidades brasileiras que estão a uma distância menor ou igual a 0.3 unidades de distância da cidade de "São Carlos-SP". Essa consulta pode ser representada algebricamente por:

$$\hat{\sigma}_{(\texttt{coordenada}\ R_q(L_2,0.3)\ (-22.02,47.89))}\texttt{CidadeBR} \tag{3.1}$$

A Figura 3.1 ilustra o exemplo da consulta de seleção por abrangência representado pela Expressão Algébrica 3.1 com a função de distância Euclidiana L_2. As cidades brasileiras contidas na circunferência de centro na cidade de "São Carlos-SP" (quadrados pretos) pertencem ao subconjunto resposta T'. Como a cidade de "São

Carlos-SP" pertence ao conjunto de valores presentes na relação `CidadeBR`, então ela "São Carlos" também pertence ao conjunto resposta T'.

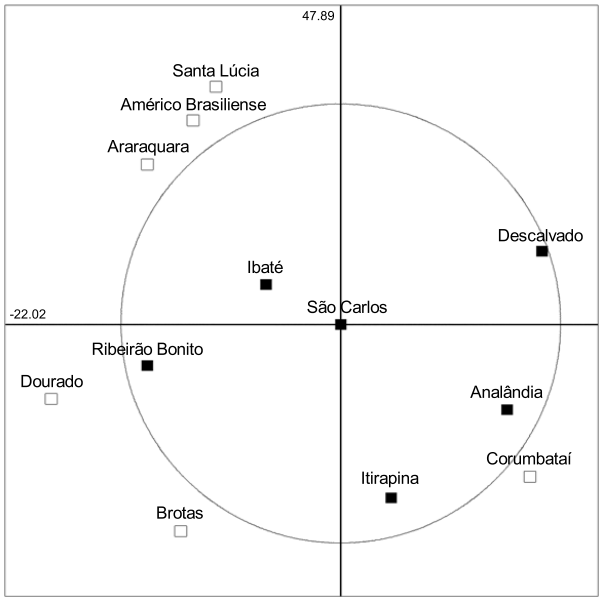

Figura 3.1: Exemplo de consulta por similaridade por abrangência (R_q), considerando $\xi = 0.3$, a função de distância L_2 e o elemento de consulta "São Carlos-SP". O conjunto resposta T' é representado pelos quadrados pretos (Expressão Algébrica 3.1).

Similaridade aos k-vizinhos mais próximos (k-*Nearest Neighbor Query*) - kNN: dados o atributo S sobre o qual é expressa a condição de similaridade, o limite de dissimilaridade $k \geq 1$ e o elemento de consulta $s_q \in \mathbb{S}$, a consulta $\hat{\sigma}_{(S\ kNN(d,k)\ s_q)}T$ recupera as k tuplas mais próximos do elemento de consulta s_q, de acordo com a função de distância d. O subconjunto resposta resultante é $T' = \{t_i \in T \mid \forall\, t_j \in [T - T'], |T'| = k, d(t_i(S), s_q) \leq d(t_j(S), s_q)\}$.

Um exemplo de consulta kNN sobre a relação `CidadeBR` é: "Selecione as 5 cidades brasileiras "mais próximas" da cidade de "São Carlos-SP", cuja latitude é igual a -22.02 e a longitude é igual a 47.89, considerando a distância Euclidiana L_2". A relação T é o conjunto das cidades brasileiras, s_q é a cidade de "São Carlos-SP", $k = 5$ é o número de elementos a ser recuperado, a métrica $d = L_2$ e o subconjunto resultante T' é o conjunto das 5 cidades brasileiras mais próximas da cidade de "São Carlos-SP". Essa consulta pode ser representada algebricamente por:

$$\hat{\sigma}_{(\texttt{coordenada}\ kNN(L_2,5)\ (-22.02,47.89))}\texttt{CidadeBR} \tag{3.2}$$

A Figura 3.2 ilustra o exemplo da consulta de seleção aos k-vizinhos mais próximos representado pela Expressão Algébrica 3.2 com a função de distância Euclidiana L_2. Os elementos (quadrados pretos) conectados por uma linha ao elemento de consulta

("São Carlos-SP") pertencem ao conjunto resposta T'. Como a cidade de "São Carlos-SP" pertence ao conjunto de valores presentes na relação `CidadeBR`, então ela "São Carlos" também pertence ao conjunto resposta T'.

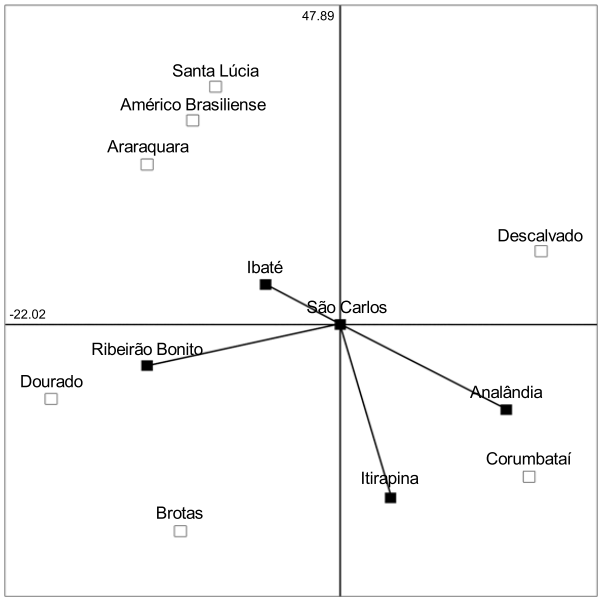

Figura 3.2: Exemplo de consulta por similaridade aos k-vizinhos mais próximos (kNN), considerando $k = 5$, a função de distância L_2 e o elemento de consulta "São Carlos-SP". O conjunto resposta T' é representado pelos quadrados pretos (Expressão Algébrica 3.2).

3.2.2 Binários

Os operadores por similaridade binários são utilizados em operações de **junção por similaridade**. Eles comparam pares de elementos de dois conjuntos de dados amostrados no mesmo domínio $\mathbb{S}$ e, nesse caso, não existem os elementos de consulta s_q. Então, uma operação de junção por similaridade sobre um atributo complexo S_1 de uma relação T_1 (chamada relação da esquerda da junção), $S_1 \in T_1$, e o atributo complexo S_2 da relação T_2 (chamada relação da direita da junção), $S_2 \in T_2$, pode ser representada, algebricamente, como:

$$T_1 \overset{S_1\ Op_j(d,lim)\ S_2}{\bowtie} T_2,$$

em que $\langle S_1\ Op_j(d, lim)\ S_2\rangle$ é o **predicado de junção por similaridade** e $\langle Op_j(d, lim)\rangle$ é o operador binário de junção por similaridade que utiliza a função de distância d e o limite (ou *threshold*) de dissimilaridade lim para obter o resultado da junção por similaridade, isto é, para obter os pares de elementos $\{< t_i, t_j > \mid\ t_i \in T_1 \text{ e } t_j \in T_2\}$ ordenados de acordo com as suas distâncias. Existem basicamente três tipos de junção por similaridade, como descrito a seguir (Böhm e Krebs, 2002).

Junção por abrangência (*Range Join*) - $\overset{R_q}{\bowtie}$: dados os atributos S_1 e S_2 sobre o qual é expressa a condição de similaridade, a função de distância d, o limite de dissimilaridade ξ, a consulta $T_1 \overset{S_1\ R_q(d,\ \xi)\ S_2}{\bowtie} T_2$ recupera os pares de elementos $\{< t_i,\ t_j >,\ t_i \in T_1,\ t_j \in T_2\}$ tal que $t_i(S_1)$ esteja à distância de no máximo ξ de $t_j(S_2)$. O subconjunto resposta resultante é $T' = \{< t_i, t_j > \in T_1 \times T_2 \mid d(t_i(S_1), t_j(S_2)) \leq \xi\}$.

Um exemplo de consulta $\overset{R_q}{\bowtie}$ sobre a relação `CidadeBR` é: "Selecione as cidades brasileiras que distam das capitais do sudeste (São Paulo-SP, Rio de Janeiro-RJ, Minas Gerais-MG e Espírito Santo-ES) até no máximo 0.15 unidades de distância, considerando a função de distância Euclidiana L_2". Neste caso, as relações T_1 e T_2 são o mesmo conjunto, o conjunto das cidades brasileiras, $\xi = 0.15$ é o limite de dissimilaridade, a métrica $d = L_2$ e o subconjunto resultante T' é o conjunto das cidades brasileiras na região das capitais do sudeste. Essa consulta pode ser representada algebricamente por:

$$(\sigma_{\texttt{((cidade='São Paulo' AND estado='SP') OR (cidade='Belo Horizonte' AND estado='MG') OR (cidade='Rio de Janeiro' AND estado='RJ') OR (cidade='Vitória' AND estado='ES'))}} (\texttt{CidadeBR})) \overset{\texttt{coordenada}\ R_q(L_2,0.15)\ \texttt{coordenada}}{\bowtie} \texttt{CidadeBR} \tag{3.3}$$

A Figura 3.3 ilustra o exemplo da consulta de junção por abrangência representado na Expressão Algébrica 3.3 com a função de distância Euclidiana L_2. Os elementos contidos nas circunferências (triângulos pretos) pertencem ao conjunto resposta T'.

Diversos autores consideram que existe apenas a junção por abrangência para as operações de junção por similaridade; nesse caso, ela é freqüentemente chamada simplesmente de "junção por similaridade" ou "junção baseada em distância" (Shim et al., 2002).

Junção por vizinhança (k-*Nearest Neighbor Join*) - $\overset{kNN}{\bowtie}$: dados os atributos S_1 e S_2 sobre o qual é expressa a condição de similaridade, o limite de dissimilaridade $k \geq 1$, a consulta $T_1 \overset{S_1\ kNN(d,\ k)\ S_2}{\bowtie} T_2$ recupera os $|T_1| * k$ pares de elementos $\{< t_i,\ t_j >, t_i \in T_1, t_j \in T_2\}$ tal que existam k pares para cada elemento pertencente a T_1 junto com os elementos de T_2 mais próximos. O subconjunto resposta resultante é $T' = \{< t_i, t_j > \in T_1 \times T_2 \mid |T'| = [|T_1| * k], \forall < t_i, t_j > \in T', \forall < t_i, t_n > \in [T_1 \times T_2 \backslash T'],\ d(t_i(S_1), t_j(S_2)) \leq d(t_i(S_1), t_n(S'))\}$, no qual o atributo $S' \in T'$.

Um exemplo de consulta $\overset{kNN}{\bowtie}$ sobre a relação `CidadeBR` é: "Selecione as 6 cidades brasileiras mais próximas a cada capital do sudeste, considerando a função de distância Euclidiana L_2". As relações T_1 e T_2 são o mesmo conjunto, o conjunto das

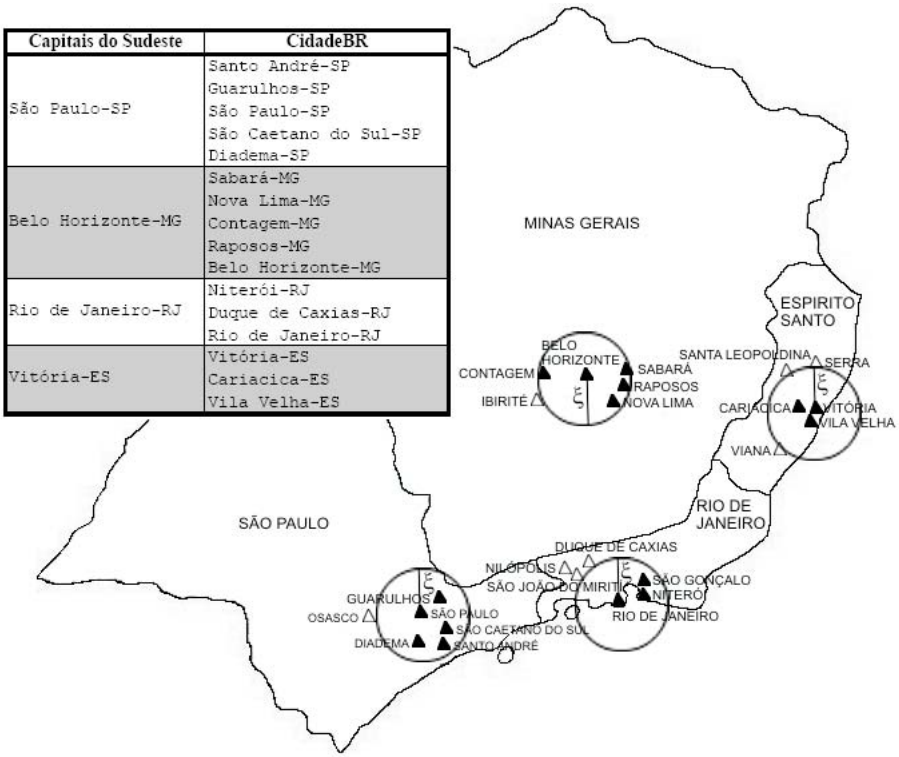

Figura 3.3: Exemplo de consulta de junção por abrangência ($\overset{R_q}{\bowtie}$), considerando $\xi = 0.15$ e a função de distância L_2. O conjunto resposta T' é representado pelos triângulos pretos (Expressão Algébrica 3.3).

cidades brasileiras, $k = 6$ é o número de vizinhos de cada capital, a métrica $d = L_2$ e o subconjunto resultante $T' \subseteq T_1 \times T_2$ é o conjunto das cidades brasileiras (6) mais próximas a cada capital da região sudeste ('SP', 'RJ', 'ES' e 'MG'). Essa consulta pode ser representada algebricamente por:

$$(\sigma_{\texttt{((cidade='São Paulo' AND estado='SP') OR (cidade='Belo Horizonte' AND estado='MG') OR (cidade='Rio de Janeiro' AND estado='RJ') OR (cidade='Vitória' AND estado='ES'))}} (\texttt{CidadeBR})) \overset{\texttt{coordenada } kNN[L_2,6] \texttt{ coordenada}}{\bowtie} \texttt{CidadeBR}. \quad (3.4)$$

A Figura 3.4 ilustra o exemplo da consulta de junção por vizinhança representado na Expressão Algébrica 3.4 com a função de distância Euclidiana L_2. Os elementos (triângulos pretos) conectados por uma linha pertencem ao conjunto resposta T'.

Junção por proximidade (k-*Closest Neighbor Join*) - $\overset{kCN}{\bowtie}$: dados os atributos S_1 e S_2 sobre o qual é expressa a condição de similaridade, o limite de dissimilaridade $k \geq 1$, a consulta $T_1 \overset{S_1\ kCN(d,\ k)\ S_2}{\bowtie} T_2$ recupera os k pares de elementos $\{< t_i,\ t_j >, t_i \in T_1,\ t_j \in T_2\}$ que estejam mais próximos entre si. O subconjunto resposta

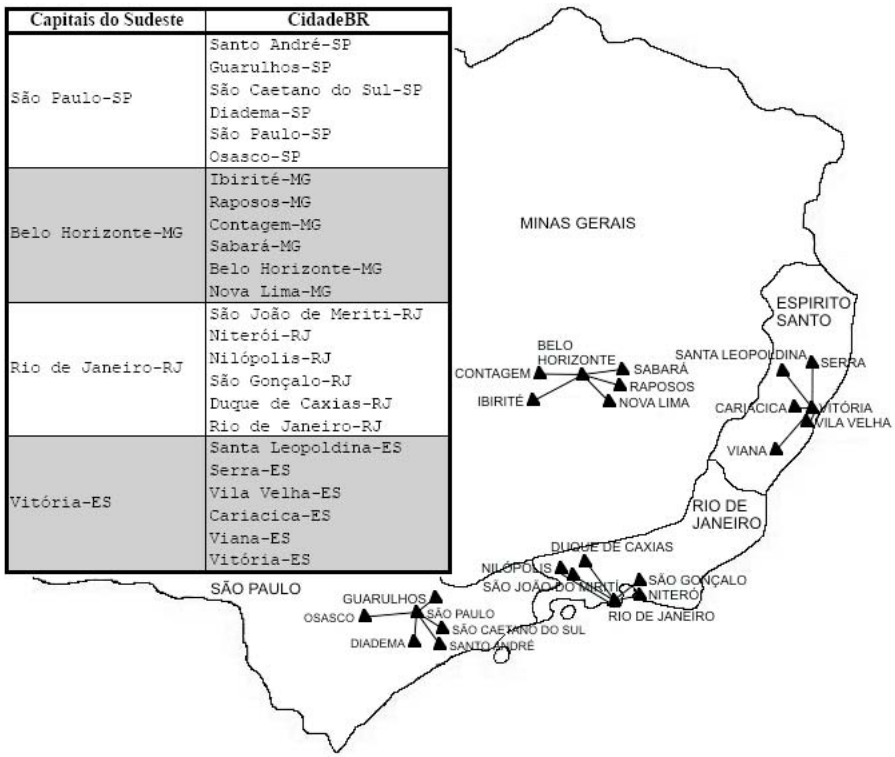

Figura 3.4: Exemplo de consulta de junção por vizinhança ($\overset{kNN}{\bowtie}$), considerando $k = 6$ e a função de distância L_2. O conjunto resposta T' é representado pelos triângulos pretos (Expressão Algébrica 3.4).

resultante $T' = \{< t_i, t_j > \in T_1 \times T_2 \mid |T'| = k \text{ e } \forall < t_i, t_j > \in T', \forall < t_m, t_n > \in [T_1 \times T_2 \backslash T'], d(t_i(S_1), t_j(S_2)) \leq d(t_m(S'), t_n(S'))\}$, no qual o atributo $S' \in T'$.

Um exemplo de consulta $\overset{kCN}{\bowtie}$ sobre a relação `CidadeBR` é: "Selecione os 6 pares de cidades brasileiras e capitais do sudeste mais próximos entre si, considerando a função de distância Euclidiana L_2". As relações T_1 e T_2 são o mesmo conjunto, o conjunto das cidades brasileiras, $k = 6$ é o número de elementos a ser retornado, a métrica $d = L_2$ e o subconjunto resultante T' é o conjunto de 6 pares de cidades brasileiras e capitais do sudeste mais próximos entre si. Essa consulta pode ser representada algebricamente por:

$$(\sigma_{\texttt{((cidade='São Paulo' AND estado='SP') OR (cidade='Belo Horizonte' AND estado='MG') OR (cidade='Rio de Janeiro' AND estado='RJ') OR (cidade='Vitória' AND estado='ES'))}}(\texttt{CidadeBR})) \overset{\texttt{coordenada } kCN[L_2,6] \texttt{ coordenada}}{\bowtie} \texttt{CidadeBR} \tag{3.5}$$

A Figura 3.5 ilustra o exemplo da consulta de junção por proximidade representado na Expressão Algébrica 3.5 com a função de distância Euclidiana L_2. Os elementos (triângulos) em preto e os elementos (triângulos pretos) conectados por uma linha pertencem ao conjunto resposta T'.

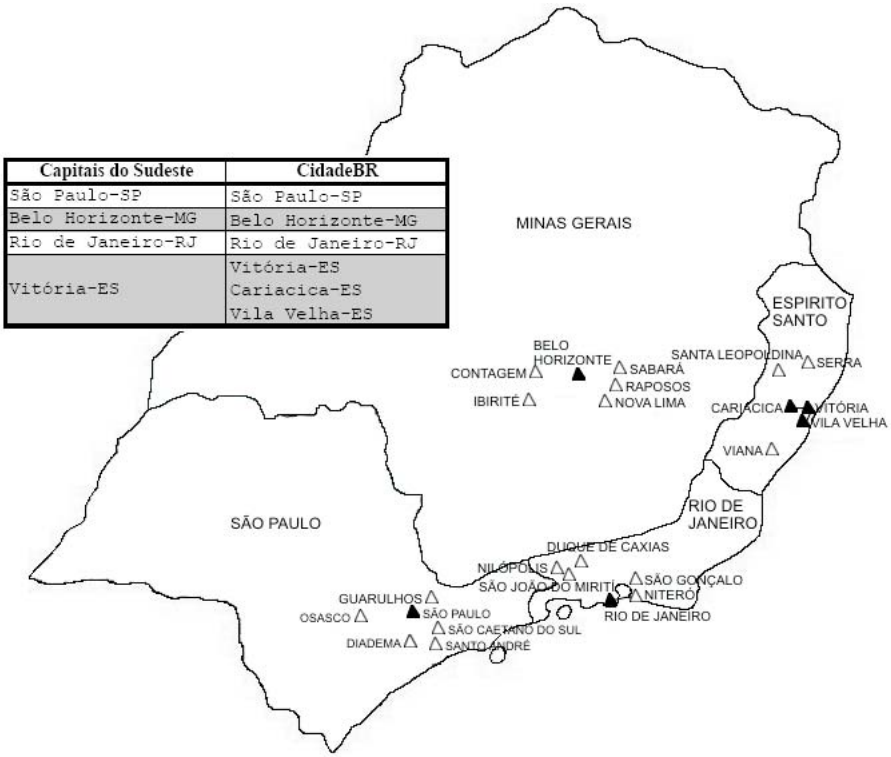

Figura 3.5: Exemplo de consulta de junção por proximidade ($\overset{kCN}{\bowtie}$), considerando $k = 6$ e a função de distância L_2. O conjunto resposta T' é representado pelos triângulos pretos (Expressão Algébrica 3.5).

Vale observar que uma consulta kNN(unária ou binária) ou kCN pode ou não prover uma lista de empate em seus resultados. Uma consulta kNN ou kCN sem lista de empate retorna qualquer combinação dos k elementos empatados à distância máxima, isto é, a consulta kNN ou kCN sem lista de empate pode nem sempre retornar os mesmos resultados (Arantes et al., 2004).

No caso de uma consulta kNN ou kCN com lista de empate, esta lista armazena todos os elementos que estão no limiar da condição de consulta, ou seja, pode haver vários elementos encontrados à distância máxima e, então, o número de elementos retornados pode ser maior do que k (Arantes et al., 2003). O custo computacional de uma consulta kNN ou kCN com lista de empate tende a não apresentar um aumento significativo, desde que o número de acessos a disco e o número de cálculos de distância continuem os mesmos. O tempo total é somente um pouco maior em conjuntos de dados com muitas listas de empate (Arantes et al., 2004).

As consultas por similaridade apresentadas podem ser respondidas examinando-se ou não todo o conjunto de dados T. Se não existir uma estrutura de indexação para os dados, então a busca seqüencial (*sequential scan*) é a maneira de responder às consultas. Essa estratégia não é a mais adequada em grandes bases de dados, pois o custo computacional envolvido é muito alto.

Se existir uma estrutura de indexação para os dados, pode-se alcançar um melhor desempenho na execução de consultas por similaridade porque a estrutura visa reduzir o número de cálculos de distância e acessos a disco necessários para executar estas consultas. Esta estrutura pode ser custosa em sua construção, mas minimiza cálculos de distâncias e acessos a disco quando as consultas são solicitadas (Chávez et al., 2001). Para conjuntos de dados que são caracterizados apenas por elementos e as distâncias entre eles, os métodos de acesso métricos (*Metric Access Method* - MAMs) são os mais adequados para a construção das estruturas de indexação.

3.3 Métodos de acesso métricos

Os **Métodos de Acesso Métricos (MAMs)** são estruturas de indexação que utilizam exclusivamente funções de distância que satisfaçam às três propriedades de um espaço métrico (simetria, não-negatividade e desigualdade triangular) para organizar o conjunto de dados. A objetivo de um MAM é minimizar o número de comparações (número de cálculos de distâncias) e o número de acessos a disco durante o processamento da consulta (Traina e Traina-Jr, 2003).

A idéia geral da maioria dos MAMs consiste em dividir o conjunto de dados em blocos ou nós e, para cada nó, selecionar um elemento que pode ser usado para indexar todos os elementos do nó e colocá-lo como elemento representante (s_{rep}) do conjunto. Os nós que armazenam os elementos originais são chamados de **nós folha** e os elementos representantes utilizados para indexar os demais são, por sua vez, agrupados em outros nós chamados de **nós interno** da árvore. De cada nó interno da árvore é escolhido também um elemento representante para indexá-lo, criando um novo nível de nós interno na árvore e, assim sucessivamente, novos níveis são acrescentados até que em um determinado nível apenas um nó seja necessário, chamado de **nó raiz**. Quando um novo elemento (s_i) é inserido, a distância entre ele e cada um dos representantes são calculadas e armazenadas junto com seus dados. Com isto, as distâncias entre todos os elementos armazenados e cada um dos representantes do conjunto são conhecidas e a propriedade de desigualdade triangular pode ser usada durante uma consulta por similaridade para eliminar os elementos que, com certeza, não fazem parte do conjunto resposta.

Um elemento $s_i \in T$ somente poderá ser podado se $d(s_{rep}, s_i) < d(s_{rep}, s_q) - \xi$ ou $d(s_{rep}, s_i) > d(s_{rep}, s_q) + \xi$, em que $s_{rep} \in T$ é o elemento representante, $s_q \in \mathbb{S}$ é o elemento de consulta e ξ é o limite de dissimilaridade da consulta. Dessa maneira, a quantidade de cálculos de distâncias necessárias para responder a uma consulta é reduzida, proporcionando um melhor desempenho. A Figura 3.6 mostra o funcionamento da poda por desigualdade triangular.

Na Figura 3.6, os elementos $\{s_1, s_2, s_3, s_4, s_5\} \in T$. A distância de s_i, com $1 \leq i \leq 5$, a s_{rep} é calculadas e armazenada com s_i na estrutura de indexação. Os elementos $\{s_2, s_3, s_5\}$

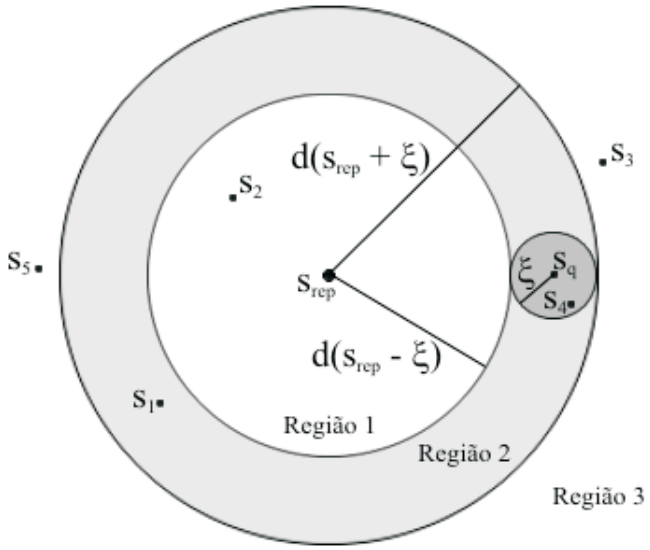

Figura 3.6: Pela propriedade da desigualdade triangular, é possível podar qualquer elemento que estiver na região 1 ou 3, sem mesmo calcular a distância entre os elementos contidos nestas regiões e o elemento de consulta. Os elementos que estão na região 2 não podem ser podados usando somente a propriedade da desigualdade triangular; é necessário calcular a distância entre o elemento de consulta (s_q) e os elementos pertencentes a esta região.

podem ser descartados usando a propriedade da desigualdade triangular, sem precisar de nenhum cálculo adicional. O elemento $\{s_1\}$ não pode ser descartado mas não faz parte do conjunto resposta, pois a distância entre ele e o elemento de consulta é maior do que o limite de dissimilaridade. O elemento $\{s_4\}$ não pode ser descartado pela propriedade da desigualdade triangular e faz parte do conjunto resposta, pois sua distância ao elemento de consulta é menor ou igual ao limite de dissimilaridade.

3.3.1 Principais métodos de acesso métrico

Existem vários trabalhos de pesquisa voltados ao estudo de como criar estruturas de indexação (MAM) que respondam a consultas por similaridade de forma eficiente. Os principais MAMs encontrados na literatura são apresentados, resumidamente, a seguir. Os trabalhos de Chávez et al. (2001), Böhm et al. (2001), Samet (2003) e de Hjaltason e Samet (2003) são boas referências para estudos mais detalhados e abrangentes sobre os MAMs.

As técnicas de divisão de um espaço métrico propostas por Burkhard e Keller (1973) foram o ponto de partida para o desenvolvimento da indexação de dados em domínios métricos. Eles apresentam **três técnicas de particionamento recursivo**, materializado por meio de árvores, que permitem a criação de MAM. A **primeira técnica** divide um conjunto de dados escolhendo um elemento como o representante do conjunto e agrupando os demais de acordo com as distâncias de cada elemento ao representante. Os elementos

que possuem uma mesma distância são agrupados em um mesmo grupo. A **segunda técnica** divide o conjunto original em uma quantidade pré-determinada de subconjuntos e seleciona um representante para cada subconjunto. Cada representante e a maior distância dele para qualquer elemento do subconjunto são mantidos na estrutura, para facilitar as consultas por similaridade posteriores. A **terceira técnica** é similar à segunda, apenas com o requisito adicional de que a distância máxima entre quaisquer dois elementos em um mesmo subconjunto não seja maior do que uma determinada constante "**c**", cujo valor é diferente para cada nível da estrutura. A escolha do valor de "**c**" tem que garantir que todos os elementos do espaço estejam em pelo menos um dos subconjuntos; um elemento pode aparecer em mais de um subconjunto. Os elementos representantes que aparecem nas técnicas de Burkhard e Keller (1973) são utilizados para descartar elementos e subárvores durante uma determinada consulta.

Após o trabalho de Burkhard e Keller (1973), diversos MAMs foram propostos e são encontrados na literatura, como *GH-tree* (*Generalized Hyper-plane tree* - Decomposição de Hiperplanos Generalizados) (Uhlmann, 1991), *Ball Decomposition* (Decomposição por bolas) (Uhlmann, 1991), *VP-tree* (*Vantage-Point tree*) (Yianilos, 1993), *MVP-tree* (*Multi-Vantage-Point tree*) (Bozkaya e Ozsoyoglu, 1997, 1999), *FQ-tree* (*Fixed Queries tree*) (Baeza-Yates et al., 1994), *GNAT* (*Geometric Near-Neighbor Access Tree*) (Brin, 1995), *SA-Tree* (*Spatial Approximation tree*) (Navarro, 2002) e *bu-Tree* (bottom-up index tree) (Liu et al., 2006). Todos esses MAMs constroem a estrutura de indexação utilizando todo o conjunto de dados disponível numa única operação, limitando sua aplicação em ambientes de bases de dados dinâmicas. Eles não permitem operações posteriores de inserção e remoção de elementos, pois não há a garantia de permanecerem balanceadas e requerem custosas reorganizações para evitar degradação de seu desempenho. Por isso, são denominados métodos estáticos.

Outros métodos, como a *M-tree* (Ciaccia et al., 1997), a *Slim-tree* (Traina et al., 2002; Traina-Jr. et al., 2000), a *Omni-family* (família Omni) (Santos-Filho et al., 2001; Traina-Jr. et al., 2006a), a *DF-tree* (*Distance Fields tree*) (Traina-Jr. et al., 2002), a DBM-tree (*Density-Based Metric tree*) (Vieira et al., 2004), a EGNAT (*Evolutionary Geometric Near-neighbor Access Tree*) (Uribe et al., 2006), a *DBM*-Tree* (Ocsa e Cuadros-Vargas, 2007) e a *MM-tree*(*Memory-Based Metric Tree*) (Pola et al., 2007), são MAMs dinâmicos que dispõem da operação de inserção a qualquer momento e não exigindo reestruturações periódicas, mas não da operação de remoção de elementos, isto é, a remoção apenas marca os elementos como removidos, sem efetivamente removê-los.

Os MAMs apresentados na literatura visam primordialmente responder a seleções por similaridade e não, a junções, apesar de estas operações também serem consideradas básicas em consultas por similaridade. A maioria dos trabalhos desenvolvidos para responderem a junções por similaridade concentra-se em efetuar as operações de junção baseadas no método de indexação espacial *R-tree* ou nas suas variantes, como por

exemplo, a *R-tree Spatial Join* - RSJ (Brinkhoff et al., 1993) e a BFRJ (Koudas e Sevcik, 1998). Um dos poucos trabalhos visando junções especificamente em espaços métricos está em Dohnal et al. (2003), que apresenta a *D-tree* e analisa três alternativas para a implementação dos algoritmos de junção por abrangência, que reduzem a complexidade normalmente quadrática sobre o número de tuplas das relações operadas: filtragem, particionamento e disjunção de seleções por abrangência; em Böhm e Krebs (2002) é apresentada a estrutura *multipage index* (MuX), especificamente desenvolvida para auxiliar a execução de junções por vizinhos mais próximos; e em Yang e Lin (2002) é apresentada a estrutura *b-Rdnn-tree* (*bichromatic Rdnn-tree*), para responder a operações de junção por proximidade, utilizando operações unárias de busca aos k-vizinhos mais próximos reversos (esse é um tipo de operação unária, que recupera as tuplas que têm o elemento de consulta como um de seus k-vizinhos mais próximos). Em Seraphim (2005) são apresentados algoritmos simples de junção por similaridade implementados em estruturas métricas, como a *Slim-tree* e a *M-tree*.

3.4 Algoritmos para consultas por similaridade

Os algoritmos para responder a consultas por similaridade que utilizam estruturas de indexação têm motivado muitas pesquisas. Algoritmos para responderem a **consultas por similaridade por abrangência** são implementados pelo uso de um limite de dissimilaridade ξ, conhecido desde o início da busca. Assim, um algoritmo para consultas por abrangência, chamado de $Range(s_q, \xi)$, calcula a distância do elemento s_q com os elementos $s_{i's}$ armazenados no conjunto de dados, e inclui no conjunto resposta todos aqueles que estão a uma distância menor ou igual a ξ, isto é, $\{t_i(S) \mid d(t_i(S), s_q) \leq \xi\}$ (Arantes, 2005). Árvores métricas (MAMs hierárquicos) são percorridas da raiz até as suas folhas e consideram o limite de dissimilaridade ξ e as propriedades inerentes aos MAM e ao domínio de dados para limitar o processo de busca. Podas, usando a propriedade de desigualdade triangular, são feitas para eliminar subárvores durante uma busca. Portanto, uma subárvore é percorrida apenas quando ela se qualifica como candidata a conter algum elemento da resposta para a consulta (Böhm e Kriegel, 2001; Ciaccia et al., 1997).

Já nos algoritmos para **consulta por similaridade aos k-vizinhos mais próximos** não há um limite de dissimilaridade conhecido desde o início da busca. Portanto, um algoritmo para consultas aos k-vizinhos mais próximos, chamado de $Nearest(s_q, k)$, utiliza um limite de dissimilaridade dinâmico iniciado com um valor maior do que a máxima distância entre qualquer par de elementos do conjunto de dados T, ou simplesmente infinito (∞). O algoritmo é executado ao calcular a distância do elemento s_q com os elementos $s_{i's}$ armazenados. Ao encontrar um elemento s_i com distância inferior ao limite de dissimilaridade dinâmico, ele é inserido na resposta. A partir de k elementos na resposta, o limite de dissimilaridade dinâmico vai sendo ajustado e recebe o valor da

distância do k-ésimo elemento (o mais distante encontrado até o momento no conjunto resposta). Quando um elemento s_i com distância inferior ao limite de dissimilaridade dinâmico é encontrado, após haver k elementos na resposta, esse elemento é inserido na resposta, o elemento mais distante é retirado e o limite de dissimilaridade dinâmico é ajustado adequadamente.

Ao contrário dos algoritmos $Range(s_q, \xi)$, em que a ordem de busca das subárvores não é relevante, pois não influencia no desempenho do algoritmo, em algoritmos $Nearest(s_q, k)$ a ordem de busca é muito importante pois, como o limite de dissimilaridade do k-ésimo elemento não é conhecido desde o início da busca para a consulta aos vizinhos mais próximos (limite de dissimilaridade dinâmico), ele é reduzido progressivamente à medida que elementos mais próximos são encontrados durante o processo de travessia da árvore. Escolher caminhos que levem a uma melhor capacidade de poda, desde o começo, é um dos objetivos mais perseguidos no desenvolvimento de novos algoritmos para consultas aos vizinhos mais próximos. Os algoritmos $Range(s_q, \xi)$ e $Nearest(s_q, k)$ descritos são chamados de algoritmos básicos para as consultas por similaridade (Arantes, 2005).

Com o objetivo de melhorar o desempenho de consultas aos vizinhos mais próximos, vários algoritmos que usam diferentes abordagens têm sido propostos por pesquisadores. Entre as abordagens existentes, a mais comum é a *branch-and-bound*. Nessa abordagem, uma estrutura de indexação é percorrida da raiz e, em cada passo, uma heurística é utilizada para determinar quais nós podem ser podados da consulta e quais devem ser percorridos em seguida. Em espaços métricos, a desigualdade triangular é a principal propriedade utilizada para podar nós. Roussopoulos et al. (1995) propuseram um algoritmo para encontrar os k-vizinhos mais próximos em conjuntos de dados multidimensionais armazenados na *R-tree*. Outros algoritmos baseados nesta abordagem são apresentados em Benetis et al. (2002), Cheung e Fu (1998) e Ciaccia et al. (1997) para conjuntos de dados armazenados na *M-tree* e em Traina-Jr. et al. (2002) para conjuntos de dados armazenados na *Slim-tree*.

Outra abordagem utilizada para melhorar o desempenho de consultas aos vizinhos mais próximos baseia-se no uso de algoritmos incrementais (Hjaltason e Samet, 1999, 2003; Park e Kim, 2003). A idéia geral dessa abordagem é: dado que os k-vizinhos mais próximos já foram calculados, os $(k+1)$-vizinhos mais próximos podem ser obtidos de maneira eficiente sem que os seus k-vizinhos mais próximos precisem ser recalculados. Para isto, é utilizada uma lista de prioridade global e são definidas heurísticas que garantem que os k elementos mais próximos estão sempre no início da lista e, para obter o $(k+1)$-ésimo elemento, não é necessário reiniciar a busca, pois basta continuar a percorrer a lista de prioridade a partir do passo anterior em que, nesse caso, foi obtido o k-ésimo elemento. Outros algoritmos para responderem a consultas aos vizinhos mais próximos são os algoritmos *multi-step* (Korn et al., 1996; Seidl e Kriegel, 1998) e o algoritmo paralelo proposto em Berchtold et al. (1997). O trabalho de Vieira et al. (2007) apresenta o algoritmo chamado

$k - NNF(s_q, k)$, que usa técnicas para estimar o limite de dissimilaridade final de uma consulta aos vizinhos mais próximos e utiliza a dimensão intrínseca do conjunto de dados, medida pela sua dimensão de correlação fractal, para acelerar estas consultas.

Todos os algoritmos descritos até aqui consideram apenas um predicado por similaridade, isto é, consideram apenas o predicado *range* ou o predicado *k-nearest*.

Nos trabalhos de Chaudhuri e Gravano (1996), Fagin (1996), Ciaccia et al. (1998), Böhm et al. (2001), Arantes et al. (2003), Arantes et al. (2004), Arantes (2005) e Traina-Jr. et al. (2006b) são descritos algoritmos para consultas por similaridade complexas, compostas por mais de um predicado. Em Chaudhuri e Gravano (1996) e Fagin (1996), a idéia básica é que a avaliação de predicados por similaridade não pode ser realizada de maneira independente, uma vez que o custo final da consulta depende dos custos combinados da execução de cada predicado isoladamente, avaliados por módulos independentes entre si. O algoritmo A_0 (Fagin, 1996), para responder a consultas por similaridade complexas sobre múltiplos atributos e múltiplas condições de consulta, retorna os k-vizinhos que atendem da maneira mais apropriada à consulta como um todo. O trabalho apresentado por Chaudhuri e Gravano (1996) é similar àquele publicado em Fagin (1996), mas os autores propuseram uma estratégia para transformar consultas aos k-vizinhos mais próximos em uma conjunção de consultas por abrangência (que são mais simples de serem executadas), usando métricas previamente levantadas sobre a distribuição de distâncias entre os elementos de dados.

Em Ciaccia et al. (1998) estuda-se o caso em que todos os critérios de similaridade se referem a uma única característica. Nesse trabalho, os autores propuseram um algoritmo mais eficiente do que o algoritmo A_0, usando um MAM. Em Böhm et al. (2001), é apresentada uma técnica para estimar o custo de consultas por similaridade complexas sobre múltiplos centros de consulta e múltiplos atributos. A técnica chamada GeVAS (*Generalized VA-File-based Search*) consiste em uma coleção de algoritmos para construir estruturas de indexação do tipo VA-File (*Vector Approximation File*) (Weber et al., 1998) para executar consultas sobre vários atributos e incorpora a idéia da execução em paralelo de diversas consultas simples propostas em Ciaccia et al. (1998). Todos os algoritmos citados usam algum tipo de função de pontuação para avaliar os pontos obtidos pelos elementos em cada predicado por similaridade. Essa avaliação determina a pertinência de um elemento à resposta da consulta por similaridade complexa envolvendo mais de um predicado.

Nos trabalhos propostos por Arantes et al. (2003), Arantes et al. (2004), Arantes (2005), Traina-Jr. et al. (2006b) são descritos algoritmos para executar consultas por similaridade complexas entre os predicados *range* e *k-nearest* usando os operadores Booleanos E ($\wedge$), OU($\vee$) e Negação ($\neg$) como, por exemplo, o algoritmo *kAndRange*, que executa a conjunção dos predicados *range* e *k-nearest* com um mesmo elemento central.

3.5 Algumas propostas de álgebras

Na literatura, há algumas propostas de álgebra que lidam com as consultas por similaridade. No entanto, nenhuma destas propostas trata adequadamente as consultas por similaridade no SGBD relacional. A primeira álgebra a considerar as consultas por similaridade foi a *multi-similarity algebra* (MSA) apresentada por Adali et al. (1998). Ela é baseada em abstrações que não são plenamente consistentes com o modelo relacional (Atnafu et al., 2001).

Outros trabalhos entendem a similaridade como incerta ou imprecisa e baseiam suas propostas na lógica *fuzzy* (Belohlávek et al., 2007; Ciaccia et al., 2000; Penzo, 2005). Porém, o fato de manipular dados complexos requer, em geral, avaliar sua similaridade e isto não significa que exista incerteza ou imprecisão nestes dados mas, somente que comparações por igualdade não têm utilidade nestes domínios.

Há também abordagens baseadas em ordem (*rank*), as quais são consistentes com o modelo relacional e aplicáveis para similaridade, por mapearem as funções de distância como um critério de ordenação ou critério de *ranking* (Adali et al., 2004; Li et al., 2005).

3.6 Considerações finais

Neste capítulo, foram apresentadas as consultas por similaridade que consistem em procurar por elementos de um conjunto de dados que, segundo algum critério de similaridade, sejam mais similares a um determinado elemento de consulta. Os critérios de similaridade utilizados nestes tipos de consultas são: os unários e os binários. Foram apresentados dois tipos de consultas por similaridade com os operadores unários: a seleção por abrangência e a seleção aos k-vizinhos mais próximos; e três consultas com os operadores binários: a junção por abrangência, a junção por vizinhança e a junção por proximidade.

Para responder a estas consultas, e minimizar os cálculos de distância e os acessos a disco, são utilizadas as estruturas de indexação chamadas de Métodos de Acesso Métricos (MAMs) capazes de indexar dados que são caracterizados apenas por elementos e as distâncias entre eles.

Os conceitos de consultas por similaridade apresentados neste capítulo, principalmente as consultas que utilizam os operadores unários, isto é, as consultas de seleção por similaridade, foram usados durante todo o desenvolvimento deste trabalho de mestrado, de modo que elas foram incorporadas a álgebra relacional desenvolvendo, assim, a álgebra por similaridade, apresentada no Capítulo 4.

Capítulo

4

Álgebra por Similaridade

4.1 Introdução

Para que um SGBD relacional dê suporte a dados complexos, ele precisa responder a consultas por similaridade incorporando estas consultas seguindo os padrões já existentes no processamento de consultas tradicionais. Portanto, o processador de consultas deverá prover maneiras de interpretar as consultas por similaridade expressas em SQL, traduzir as consultas por similaridade em expressões equivalentes em álgebra relacional, avaliar as possibilidades de otimização e executar as consultas. Uma maneira natural para isso é fornecer suporte à consultas por similaridade na álgebra relacional que representa a base das operações nos SGBDs relacionais e nas linguagens de consultas, tais como a SQL.

Este capítulo apresenta a extensão da álgebra relacional para responder as consultas por similaridade desenvolvida neste trabalho necessária para dar suporte às consultas por similaridade unárias no processamento de consultas em um SGBD relacional.

A álgebra por similaridade proposta neste trabalho de mestrado baseia-se na abordagem relacional baseada em *rank* introduzindo explicitamente o suporte para consultas por similaridade. Na Seção 4.2 são apresentados o modelo e a álgebra por similaridade e na Seção 4.3 são apresentadas as considerações finais.

4.2 Álgebra por similaridade

Nesta seção são descritos os fundamentos do modelo de dados e da **Álgebra por Similaridade**, propostos neste trabalho para as consultas por similaridade. Na Subseção 4.2.1, o modelo de dados proposto é apresentado; na Subseção 4.2.2, as operações de consultas

por similaridade unárias são definidas sobre o modelo proposto; e na Subseção 4.2.3, as regras algébricas para os operadores por similaridade são definidas.

4.2.1 Modelo de dados

Conforme foi mostrado no Capítulo 3, dados pertencentes ao um domínio $\mathbb{S}$ são vistos como elementos pertencentes a um espaço métrico $M =< \mathbb{S}, d >$. Já que os espaços métricos não satisfazem a propriedade de relação de ordem total, para que os elementos pertencentes a M estabeleçam alguma ordenação entre eles, podemos usar a abordagem baseada em ordem (*rank*) para ordenar uma relação T, utilizando as funções de distância como critério de *ranking*. Assim, a relação $T = \{A_1, \ldots, A_j, S_1, \ldots, S_n\}$ agora possui um atributo implícito D, que armazena as distâncias dos elementos da relação T durante uma consulta. Para cada predicado da consulta, existe um atributo implícito D que armazena sua distância. A idéia principal de D é possibilitar que os resultados intermediários de uma consulta kNN possam ser usados como entrada de outros operadores.

Portanto, as operações de consultas por similaridade unárias, apresentadas no Capítulo 3 foram redefinidas para que, a partir de agora, elas possuam o atributo implícito de distância D. Note que, nas consultas por similaridade as distâncias com menores valores correspondem aos melhores resultados, diferente da notação usual de *rank*.

4.2.2 Operações de consultas por similaridade unárias

As definições formais das operações de consultas por similaridade unárias são apresentadas nesta subseção. A primeira definição apresentada é da similaridade por abrangência.

Definição 4.1. ***Similaridade por abrangência (Range query)*** **-** *R_q: dados o atributo S sobre o qual é expressa a condição de similaridade, a função de distância d, o limite de dissimilaridade ξ e o elemento de consulta $s_q \in \mathbb{S}$, a consulta $\hat{\sigma}_{(S\ R_q(d,\xi)\ s_q)}T$ recupera todas as tuplas $t_i \in T$ tal que $d(t_i(S), s_q) \leq \xi$ e preserva a ordem induzida por D. O subconjunto resposta resultante é $T' = \{t_i \in T \mid d(t_i(S), s_q) \leq \xi\}$. É importante notar que a similaridade por abrangência gera um atributo de distância $D = d(t_i(S), s_q)$, para toda tupla $t_i \in T$.*

Existe uma variação particular da consulta por abrangência, chamada **consulta pontual**, cujo limite de dissimilaridade $\xi = \mathbf{0}$. O objetivo desta consulta é verificar se o elemento de consulta $s_q \in T$.

A operação complementar da seleção por similaridade por abrangência é chamada **Seleção por Similaridade por Abrangência Inversa** (*Reversed Range Query* - R_q^{-1}).

Definição 4.2. ***Similaridade por abrangência inversa (Reversed Range query)*** **-** *R_q^{-1}: dados o atributo S sobre o qual é expressa a condição de similaridade, a função*

de distância d, o limite de dissimilaridade ξ e o elemento de consulta $s_q \in \mathbb{S}$, a consulta $\hat{\sigma}_{(S\ R_q^{-1}(d,\xi)\ s_q)}T$ recupera todas as tuplas $t_i \in T$ tal que $d(t_i(S), s_q) > \xi$ e preserva a ordem induzida por D. O subconjunto resposta resultante é $T' = \{t_i \in T \mid d(t_i(S), s_q) > \xi\}$. É importante notar que a similaridade por abrangência gera um atributo de distância $D = d(t_i(S), s_q)$, para toda tupla $t_i \in T$.

A operação de similaridade aos k-vizinhos mais próximos é definida a seguir.

Definição 4.3. ***Similaridade aos k-vizinhos mais próximos (k-Nearest Neighbor query) -*** *kNN: dados o atributo S sobre o qual é expressa a condição de similaridade, o limite de dissimilaridade $k \geq 1$ e o elemento de consulta $s_q \in \mathbb{S}$, a consulta $\hat{\sigma}_{(S\ kNN(d,k)\ s_q)}T$ recupera as k tuplas mais próximos do elemento de consulta s_q, de acordo com a função de distância d. O subconjunto resposta resultante é $T' = \{t_i \in T \mid \forall\, t_j \in [T - T'], |T'| = k, d(t_i(S), s_q) \leq d(t_j(S), s_q)\}$ e produz uma nova ordem induzida por D. É importante notar que a similaridade aos k-vizinhos mais próximos gera um atributo de distância $D = d(t_i(S), s_q)$, para toda tupla $t_i \in T$, e que este recebe um valor numérico que indica a posição da tupla t_i no resultado da consulta.*

Como uma nova ordem induzida para cada consulta kNN é gerada pelo atributo D, a seleção por similaridade aos k-vizinhos mais próximos, a partir de agora, passou a ser denotada por $\ddot{\sigma}$ ao invés de $\hat{\sigma}$ como era denotada anteriormente. Portanto, a seleção por similaridade aos k-vizinhos mais próximos é denotada por:

$$\ddot{\sigma}_{(S\ kNN(d,k)\ s_q)}T. \tag{4.1}$$

Formalmente, esta seleção é denotada por:

$$kNN = \bigcup_{1 \leq j \leq k} \{t_j\},$$

tal que cada $\{t_j\}$ é dado por:

$$\begin{aligned}
\{t_1\} &= \{t_i \in T \mid \forall\, t_j \in T - \{t_i\},\ d(t_i(S), s_q) \leq d(t_j(S), s_q)\},\\
\{t_2\} &= \{t_i \in T - \{t_1\} \mid \forall\, t_j \in T - \{t_1, t_i\},\ d(t_i(S), s_q) \leq d(t_j(S), s_q)\},\\
&\vdots\\
\{t_k\} &= \{t_i \in T - \{t_1, \ldots, t_{k-1}\} \mid \forall\, t_j \in T - \{t_1, \ldots, t_{k-1}, t_i\},\ d(t_i(S), s_q) \leq d(t_j(S), s_q)\}.
\end{aligned} \tag{4.2}$$

A operação complementar da seleção por similaridade aos k-vizinhos mais próximos é a **Seleção por similaridade aos k-vizinhos mais distantes** (*k-Farthest Neighbor query - kFN*).

Definição 4.4. ***Similaridade aos vizinhos mais distantes*** *(k-Farthest Neighbor query) - kFN: dados o atributo S sobre o qual é expressa a condição de similaridade, o*

limite de dissimilaridade $k \geq 1$ e o elemento de consulta $s_q \in \mathbb{S}$, a consulta $\hat{\sigma}_{(S\ kFN(d,k)\ s_q)}T$ recupera as k tuplas mais distantes do elemento de consulta s_q, de acordo com a função de distância d. O subconjunto resposta resultante é $T' = \{t_i \in T \mid \forall\, t_j \in [T - T'], |T'| = k, d(t_i(S), s_q) > d(t_j(S), s_q)\}$ e produz uma nova ordem induzida por D. É importante notar que a similaridade aos vizinhos mais próximos gera um atributo de distância $D = d(t_i(S), s_q)$, para toda tupla $t_i \in T$, e que este recebe um valor numérico que indica a posição da tupla t_i no resultado da consulta.

As consultas kNN e kFN podem ou não prover uma lista de empate em seus resultados. Uma consulta kNN ou kFN sem lista de empate retorna qualquer combinação dos k elementos empatados, isto é, ela escolhe o número necessário de elementos empatados para completar k. No caso de uma consulta kNN ou kFN com lista de empate, esta lista inclui todos os elementos que estão no limiar da condição de consulta no resultado; então, o número de elementos retornados pode ser maior do que k.

4.2.3 Regras algébricas

Na álgebra relacional, algumas regras algébricas são usadas para transformar uma expressão em álgebra relacional em outras expressões equivalentes. Esta transformação é realizada no módulo de Reescrita de Planos do otimizador de consultas do SGBD relacional. Para o otimizador, as seleções são operações importantes pois elas tendem a reduzir o tamanho das relações. Nesta subseção são apresentadas as regras algébricas que envolvem os operadores de seleção por similaridade por abrangência $\hat{\sigma}$ e de seleção por similaridade por vizinhança $\ddot{\sigma}$.

Seleção por similaridade por abrangência - $\hat{\sigma}$

Nas definições e provas das regras apresentadas nesta subseção é utilizado o operador de similaridade por abrangência R_q. As definições e provas das regras para o operador de similaridade por abrangência inversa R_q^{-1} são realizadas de maneira análoga.

A Regra 4.1 é válida para as condições conjuntivas:

Regra 4.1. *A operação de seleção por similaridade R_q divide condições conectadas por* `AND` *ou $\wedge$ em uma seqüência de condições conectadas por $\cap$ ou em uma cascata de operações $\hat{\sigma}$ individuais, isto é,*

$$\begin{aligned}
&\hat{\sigma}_{(S_1\ R_q(d_1,\xi_1)\ s_{q1})\ AND\ (S_2\ R_q(d_2,\xi_2)\ s_{q2})}(T) \\
&\overset{1}{=} \hat{\sigma}_{(S_1\ R_q(d_1,\xi_1)\ s_{q1})}(T)\ \cap \hat{\sigma}_{(S_2\ R_q(d_2,\xi_2)\ s_{q2})}(T) \\
&\overset{2}{=} \hat{\sigma}_{(S_1\ R_q(d_1,\xi_1)\ s_{q1})}(\hat{\sigma}_{(S_2\ R_q(d_2,\xi_2)\ s_{q2})}(T)).
\end{aligned} \tag{4.3}$$

Demostração. Usando a Definição 4.1, mostramos que:

$$\begin{aligned}
&\hat{\sigma}_{(S_1\ R_q(d_1,\xi_1)\ s_{q1})\ \text{AND}\ (S_2\ R_q(d_2,\xi_2)\ s_{q2})}(T)\\
&= \{t_i \in T |\ d(t_i(S_1), s_{q1}) \leq \xi_1 \text{ and } d(t_i(S_2), s_{q2}) \leq \xi_2\}\\
&= \{t_{i1} \in T |\ d(t_i(S_1), s_{q1}) \leq \xi_1\} \cap \{t_{i2} \in T |\ d(t_i(S_2), s_{q2}) \leq \xi_2\}\\
&\overset{1}{=} \hat{\sigma}_{(S_1\ R_q(d_1,\xi_1)\ s_{q1})}(T) \cap \hat{\sigma}_{(S_2\ R_q(d_2,\xi_2)\ s_{q2})}(T).
\end{aligned}$$

Por outro lado,

$$\begin{aligned}
&= \{t_{i1} \in T |\ d(t_i(S_1), s_{q1}) \leq \xi_1\} \cap \{t_{i2} \in T |\ d(t_i(S_2), s_{q2}) \leq \xi_2\}\\
&= \{t_{i2} \in \{t_{i2} \in T |\ d(t_i(S_2), s_{q2}) \leq \xi_2\} | d(t_i(S_1), s_{q1}) \leq \xi_1\}\\
&\overset{2}{=} \hat{\sigma}_{(S_1\ R_q(d_1,\xi_1)\ s_{q1})}(\hat{\sigma}_{(S_2\ R_q(d_2,\xi_2)\ s_{q2})}(T)).
\end{aligned}$$

□

A Regra 4.2 é válida para as condições disjuntivas:

Regra 4.2. *Se as condições de $\hat{\sigma}$ estão conectadas por* `OR` *ou* $\vee$*, a operação de seleção por similaridade R_q divide as condições em suas partes constituintes e depois realiza a união destas partes. Esta regra funciona se, e somente se, a relação T é um conjunto pois, desta maneira, as duplicações serão eliminadas corretamente, isto é,*

$$\begin{aligned}
&\hat{\sigma}_{(S_1\ R_q(d_1,\xi_1)\ s_{q1})\ OR\ (S_2\ R_q(d_2,\xi_2)\ s_{q2})}(T) =\\
&(\hat{\sigma}_{(S_1\ R_q(d_1,\xi_1)\ s_{q1})}(T)) \cup (\hat{\sigma}_{(S_2\ R_q(d_2,\xi_2)\ s_{q2})}(T)).
\end{aligned} \tag{4.4}$$

Demostração.

$$\begin{aligned}
&\hat{\sigma}_{(S_1\ R_q(d_1,\xi_1)\ s_{q1})\ \text{OR}\ (S_2\ R_q(d_2,\xi_2)\ s_{q2})}(T)\\
&= \{t_i \in T |\ d(t_i(S_1), s_{q1}) \leq \xi_1 \text{ or } d(t_i(S_2), s_{q2}) \leq \xi_2\}\\
&= \{t_{i1} \in T |\ d(t_i(S_1), s_{q1}) \leq \xi_1\} \cup \{t_{i1} \in T |\ d(t_i(S_2), x_{q2}) \leq \xi_2\}\\
&= (\hat{\sigma}_{(S_1\ R_q(d_1,\xi_1)\ s_{q1})}(T)) \cup (\hat{\sigma}_{(S_2\ R_q(d_2,\xi_2)\ s_{q2})}(T)).
\end{aligned}$$

□

As Regras 4.3 e 4.4 mostram a comutatividade do operador $\hat{\sigma}$ com o operador de seleção tradicional e com ele mesmo.

Regra 4.3. *A operação de seleção por similaridade R_q comuta sob sua composição, isto é,*

$$\hat{\sigma}_{(S_1\ R_q(d_1,\xi_1)\ s_{q1})}(\hat{\sigma}_{(S_2\ R_q(d_2,\xi_2)\ s_{q2})}(T)) = \hat{\sigma}_{(S_2\ R_q(d_2,\xi_2)\ s_{q2})}(\hat{\sigma}_{(S_1\ R_q(d_1,\xi_1)\ s_{q1})}(T)). \tag{4.5}$$

Demostração. Usando a Definição 4.1, tem-se que esta prova é obtida diretamente pela Regra 4.1.

□

Regra 4.4. *A operação de seleção por similaridade R_q e a operação de seleção tradicional comutam sob suas composições, isto é,*

$$\hat{\sigma}_{(S\ R_q(d,\xi)\ s_q)}(\sigma_{(A\,\theta\,a)}(T)) = \sigma_{(A\,\theta\,a)}(\hat{\sigma}_{(S\ R_q(d,\xi)\ s_q)}(T)), \tag{4.6}$$

no qual 'A'é um atributo tradicional, θ é um dos operadores $<,\leq,>,\geq,=$ ou $\neq$ e 'a' é uma constante.

Demostração. Usando a Definição 4.1, mostramos que:

$$\begin{aligned}
&\hat{\sigma}_{(S\ R_q(d,\xi)\ s_q)}(\sigma_{(A=a)}(T)) \\
&= \hat{\sigma}_{(S\ R_q(d,\xi)\ s_q)}\{t_i \in T | t_i(A)\ \theta\ a\} \\
&= \{t_i \in \{t_i \in T | t_i(A)\ \theta\ a\} | d(t_i(S), s_q) \leq \xi\} \\
&= \{t_i \in T |\ t_i(A)\ \theta\ a \text{ and } d(t_i(S), s_q) \leq \xi\} \\
&= \{t_i \in \{t_i \in T | d(t_i(S), s_q) \leq \xi\} | t_i(A)\ \theta\ a\} \\
&= \sigma_{(A\,\theta\,a)}(\hat{\sigma}_{(S\ R_q(d,\xi)\ s_q)}(T)).
\end{aligned}$$

□

As Regras 4.5 e 4.6 permitem que o operador $\hat{\sigma}$ seja distribuído em diferentes ramos do plano de consulta. Estas regras são utilizadas com os operadores binários: produto cartesiano ($\times$), união ($\cup$), interseção ($\cap$), diferença ($-$) e junção ($\bowtie$).

A Regra 4.5 mostra como o operador $\hat{\sigma}$ pode ser distribuído sobre os operadores de conjunto $\cup$, $\cap$ e $-$. Esta regra requer que as relações T_1 e T_2 sejam compatíveis em domínio.

Regra 4.5. *O $\hat{\sigma}$ é distributivo sobre os operadores binários $\cup$, $\cap$ e $-$, isto é, se $\theta \in \{\cup, \cap, -\}$, então,*

$$\hat{\sigma}_{(S\ R_q(d,\xi)\ s_q)}(T_1\ \theta\ T_2) = \hat{\sigma}_{(S\ R_q(d,\xi)\ s_q)}(T_1)\ \theta\ \hat{\sigma}_{(S\ R_q(d,\xi)\ s_q)}(T_2), \tag{4.7}$$

Para a união (∪), o $\hat{\sigma}$ tem que ser distribuído para ambos os ramos do plano de consulta; para a diferença (−), o $\hat{\sigma}$ deve ser distribuído para o ramo esquerdo e, opcionalmente, para o ramo direito do plano de consulta; e para a interseção (∩), o $\hat{\sigma}$ deve ser distribuído para um ramo do plano de consulta e, opcionalmente, para ambos.

Demostração. Seja $\theta = \cup$ e usando a Definição 4.1, mostramos que:

$$\begin{aligned}
&\hat{\sigma}_{(S\, R_q(d,\xi)\, s_q)}(T_1 \cup T_2) \\
&= \{t_i \in T_1 \cup T_2 |\ d(t_i(S), s_q) \leq \xi\} \\
&= \{t_i \in T_1 \mid (d(t_i(s), s_q) \leq \xi)\} \cup \{t_i \in T_2 \mid d(t_i(S), s_q) \leq \xi) \\
&= \hat{\sigma}_{(S\, R_q(d,\xi)\, s_q)}(T_1) \cup \hat{\sigma}_{(S\, R_q(d,\xi)\, s_q)}(T_2).
\end{aligned}$$

Para $\theta = \cap$ e $-$, a prova é realizada de maneira análoga.

□

Para os outros operadores binários, tais como a junção ($\bowtie$) e o produto cartesiano ($\times$), a operação $\hat{\sigma}$ deve ser distribuída para o ramo do plano de consulta que tem o atributo complexo mencionado na condição de $\hat{\sigma}$. Isto é representado na Regra 4.6.

Regra 4.6. *O operador $\hat{\sigma}$ pode somente ser distribuído sobre uma junção ou um produto cartesiano se todos os atributos complexos mencionados no predicado range pertencerem a esta relação. Supondo que a relação T_1 tem o atributo complexo S, então*

$$\hat{\sigma}_{(S\, R_q(d,\xi)\, s_q)}(T_1\, \theta\, T_2) = (\hat{\sigma}_{(S\, R_q(d,\xi)\, s_q)} T_1)\, \theta\, T_2. \tag{4.8}$$

Se a relação T_2 tem o atributo complexo S, então

$$\hat{\sigma}_{(S\, R_q(d,\xi)\, s_q)}(T_1\, \theta\, T_2) = T_1\, \theta\, \hat{\sigma}_{(S\, R_q(d,\xi)\, s_q)}(T_2). \tag{4.9}$$

para $\theta = \bowtie$ ou $\times$.

Demostração. Seja $\theta = \times$ e usando a Definição 4.1, mostramos que:

$$\begin{aligned}
&\hat{\sigma}_{(S\, R_q(d,\xi)\, s_q)}(T_1 \times T_2) \\
&= \{(t_i t_j) \in T_1 \times T_2 |\ d(t_i(S), x_q) \leq \xi\} \\
&= \{t_i \in T_1 |\ d(t_i(S), s_q) \leq \xi\} \times T_2 \\
&= \hat{\sigma}_{(S\, R_q(d,\xi)\, s_q)}(T_1) \times T_2.
\end{aligned}$$

A mesma prova pode ser aplicada se S é um atributo complexo da relação T. Para $\theta = \bowtie$, a prova é realizada de maneira análoga.

□

No caso da condição da seleção por similaridade R_q ser conjuntivas (ou disjuntivas), em que S_1 é um atributo complexo de T_1 e S_2 é um atributo complexo de T_2, as Regras 4.1 e 4.6 são usadas para deduzir e provar que,

$$\begin{aligned}&\hat{\sigma}_{((S_1\ R_q(d,\xi)\ s_q)\ and\ (S_2\ R_q(d,\xi)\ s_q))}(T_1\ \theta\ T_2)\\&= \hat{\sigma}_{(S_1\ R_q(d,\xi)\ s_q)}(T_1)\ \theta\ \hat{\sigma}_{(S_2\ R_q(d,\xi)\ s_q)}(T_2),\end{aligned} \tag{4.10}$$

para $\theta = \times$ ou $\bowtie$, isto é, mostramos que a propriedade distributiva, válida para a seleção tradicional, também vale para a seleção por abrangência. Então, a Equivalência 4.10 completa as Equivalências 4.8 e 4.9 da Regra 4.6.

As Regras 4.1, 4.2, 4.3, 4.5 e 4.6 mostram que a seleção por similaridade por abrangência tem as mesmas regras de transformação algébrica que a seleção tradicional. A Regra 4.4 mostra que a seleção por similaridade por abrangência é também comutativa com a seleção tradicional. Assim, as seleções por similaridade por abrangência podem ser tratadas como as seleções tradicionais nos algoritmos de otimização de consultas do módulo de Reescrita de Consultas do otimizador de consultas.

Seleção por similaridade aos vizinhos mais próximos - $\ddot{\sigma}$

Nas definições e provas das regras apresentadas nesta subseção é utilizado a operação de similaridade aos k-vizinhos mais próximos `kNN`. As definições e provas das regras para a operação de similaridade aos k-vizinhos mais distantes `kFN` são realizadas de maneira análoga.

As regras de transformação válidas para as consultas por vizinhança são provadas algebricamente e as regras inválidas são provadas por contradição. Nos exemplos são usados dois sub-conjuntos da relação `CidadeBR` (Capítulo 2), chamados `CidadeSaoCarlos` e `CidadeAraraquara`, os quais possuem informações geográficas a respeito de 12 cidades da região de "São Carlos-SP" (mostradas na Figura 4.1a) e 17 cidades da região de "Araraquara-SP" (mostradas na Figura 4.1b), respectivamente. Em todos os exemplos dados a seguir, os elementos (quadrados pretos) conectados por uma linha preta ao elemento de consulta pertencem ao conjunto resposta.

Considerando a álgebra relacional, a seguinte expressão é considerada válida:

$$\sigma_{(c_1 \wedge c_2)}(T) = \sigma_{c_1}(T)\ \cap \sigma_{c_2}(T) = \sigma_{c_1}(\sigma_{c_2})(T), \tag{4.11}$$

no entanto, se a seleção σ é substituída pela seleção aos k-vizinhos mais próximos $\ddot{\sigma}$, a expressão 4.11 deixa de ser válida, originando a Regra 4.7 a seguir:

Regra 4.7. *A operação de seleção por similaridade* `kNN` *com condições conjuntivas não pode dividir as condições em uma seqüência de condições conectadas por* $\cap$ *ou em uma*

Figura 4.1: Relações `CidadeSaoCarlos` e `CidadeAraraquara`. a) Relação `CidadeSaoCarlos`: 12 cidades da região de "São Carlos-SP". b) Relação `CidadeAraraquara`: 17 cidades da região de "Araraquara-SP".

cascata de operações $\ddot{\sigma}$ individuais, isto é,

$$\begin{aligned}
&\ddot{\sigma}_{(S_1\ kNN(d_1,k_1)\ s_{q1}) \wedge (S_2\ kNN(d_2,k_2)\ s_{q2})}(T) \\
&\neq \ddot{\sigma}_{(S_1\ kNN(d_1,k_1)\ s_{q1})}(T)\ \cap \ddot{\sigma}_{(S_2\ kNN(d_2,k_2)\ s_{q2})}(T) \\
&\neq \ddot{\sigma}_{(S_1\ kNN(d_1,k_1)\ s_{q1})}(\ddot{\sigma}_{(S_2\ kNN(d_2,k_2)\ s_{q2})}(T)).
\end{aligned} \tag{4.12}$$

Provamos esta regra com o seguinte contra-exemplo.

Exemplo: *"Selecione as 5 cidades mais próximas da cidade de São Carlos-SP, cuja latitude* $=-22.02$ *e longitude* $=47.89$*, e mais próximas da cidade de Araraquara-SP, cuja latitude* $=-21.79$ *e a longitude* $=48.18$*, considerando a distância Euclidiana* L_2*".*

$$\begin{aligned}
&\ddot{\sigma}_{(coordenada\ kNN(L_2,5)\ (-22.02,47.89))\ AND} \\
&\quad \underbrace{(coordenada\ kNN(L_2,5)\ (-21.79,48.18))(\texttt{CidadeSaoCarlos})}_{1} \\
&= \ddot{\sigma}_{(coordenada\ kNN(L_2,5)\ (-22.02,47.89))}(\texttt{CidadeSaoCarlos}) \cap \\
&\quad \underbrace{\ddot{\sigma}_{(coordenada\ kNN(L_2,5)\ (-21.79,48.18))}(\texttt{CidadeSaoCarlos})}_{2} \\
&= \underbrace{\ddot{\sigma}_{(coordenada\ kNN(L_2,5)\ (-22.02,47.89))}(\ddot{\sigma}_{(coordenada\ kNN(L_2,5)\ (-21.79,48.18))}(\texttt{CidadeSaoCarlos}))}_{3}.
\end{aligned} \tag{4.13}$$

A execução Expressão Algébrica 1 da Equivalência 4.13 retorna 5 tuplas, sendo elas: {`São Carlos, Analândia, Itirapina, Ribeirão Bonito, Ibaté`}; a execução da Expressão Algébrica 2 retorna 2 tuplas, sendo elas: {`Ribeirão Bonito, Ibaté`}; e a

execução da Expressão Algébrica 3 também retorna 5 tuplas, porém elas são diferentes das retornadas pela Expressão Algébrica 1. São elas: {`Ribeirão Bonito, Ibaté, Araraquara, Américo Brasiliense, Santa Lúcia`}. Isto prova que a Regra 4.7 é válida.

Existe um caso especial, apresentado no trabalho de Traina-Jr. et al. (2006b), em que os autores mostram que: se a condição da seleção por similaridade aos k-vizinhos mais próximos for conjuntiva e esta seleção for realizada sobre o mesmo elemento de consulta, isto é, quando $s_{q1} = s_{q2} = s_q$, então pode ser realizada a seleção com um único predicado *kNN* com o menor valor de k entre aqueles exigidos na condição conjuntiva, isto é,

$$\begin{aligned} &\ddot{\sigma}_{(S\ kNN(d,k_1)\ s_q)}(T) \cap \ddot{\sigma}_{(S\ kNN(d,k_2)\ s_q)}(T) \Leftrightarrow \\ &\ddot{\sigma}_{(S\ kNN(d,k_1)\ s_q)\ \text{and}\ (S\ kNN(d,k_2)\ s_q)}(T) \Leftrightarrow \\ &\ddot{\sigma}_{(S\ kNN(d,min(k_1,k_2))\ s_q)}(T) \end{aligned} \tag{4.14}$$

Para as condições disjuntivas, a álgebra relacional considera válida a seguinte expressão:

$$\sigma_{(c_1 \vee c_2)}(T) = \sigma_{c_1}(T) \cup \sigma_{c_2}(T). \tag{4.15}$$

Porém, trocando o σ por $\ddot{\sigma}$, a expressão 4.15 deixa de ser válida, originando a Regra 4.8.

Regra 4.8. *A operação de seleção por similaridade* **`kNN`** *com condições disjuntivas não pode dividir as condições em suas partes constituintes e realizar a união delas, isto é,*

$$\begin{aligned} &\ddot{\sigma}_{(S_1\ kNN(d_1,k_1)\ s_{q1} \vee S_2\ kNN(d_2,k_2)\ s_{q2})}(T) \\ &\neq \ddot{\sigma}_{(S_1\ kNN(d_1,k_1)\ s_{q1})}(T) \cup \ddot{\sigma}_{(S_2\ kNN(d_2,k_2)\ s_{q2})}(T). \end{aligned} \tag{4.16}$$

Provamos esta regra usando o seguinte contra-exemplo.

Exemplo: *"Selecione as 5 cidades mais próximas da cidade de São Carlos-SP, cuja latitude* $= -22.02$ *e longitude* $= 47.89$*, ou mais próximas da cidade de Araraquara-SP, cuja latitude* $= -21.79$ *e a longitude* $= 48.18$*, considerando a distância Euclidiana* L_2*".*

$$\begin{aligned} &\ddot{\sigma}_{(coordenada\ kNN(L_2,5)\ (-22.02,47.89))\ OR} \\ &\underbrace{(coordenada\ kNN(L_2,5)\ (-21.79,48.18))(\texttt{CidadeSaoCarlos})}_{1} \\ =&\ddot{\sigma}_{(coordenada\ kNN(L_2,5)\ (-22.02,47.89))}(\texttt{CidadeSaoCarlos}) \cup \\ &\underbrace{\ddot{\sigma}_{(coordenada\ kNN(L_2,5)\ (-21.79,48.18))}(\texttt{CidadeSaoCarlos})}_{2}. \end{aligned} \tag{4.17}$$

A execução Expressão Algébrica 1 da Equivalência 4.17 retorna 5 tuplas, sendo elas: {`São Carlos, Analândia, Itirapina, Ribeirão Bonito, Ibaté`}; e a execução

da Expressão Algébrica 2 retorna 8 tuplas, sendo elas: {`Ribeirão Bonito, Ibaté, São Carlos, Analândia, Araraquara, Itirapina, Américo Brasiliense, Santa Lúcia`}. Isto prova que a Regra 4.8 é válida.

Existe um caso especial, apresentado no trabalho de Traina-Jr. et al. (2006b), em que os autores mostram que: se a condição da seleção por similaridade aos k-vizinhos mais próximos for disjuntiva e esta seleção for realizada sobre o mesmo elemento de consulta, isto é, quando $s_{q1} = s_{q2} = s_q$, então pode ser realizada a seleção com um único predicado *kNN* com o maior valor de k entre aqueles exigidos na condição disjuntiva, isto é,

$$\begin{gathered}\ddot{\sigma}_{(S\ kNN(d,k_1)\ s_q)}(T) \cup \ddot{\sigma}_{(S\ kNN(d,k_2)\ s_q)}(T) \Leftrightarrow \\ \ddot{\sigma}_{(S\ kNN(d,k_1)\ s_q)\ \text{or}\ (S\ kNN(d,k_2)\ s_q)}(T) \Leftrightarrow \\ \ddot{\sigma}_{(S\ kNN(d,max(k_1,k_2))\ s_q)}(T)\end{gathered} \tag{4.18}$$

Em álgebra relacional, a comutatividade de σ, dada pela seguinte expressão, é valida.

$$\sigma_{c1}(\sigma_{c2}(T)) = \sigma_{c_2}(\sigma_{c_1}(T)). \tag{4.19}$$

Entretanto, se trocarmos o σ pelo $\ddot{\sigma}$, a Expressão 4.19 torna-se inválida. As Regras 4.9, 4.10 e 4.11 apresenta a não-comutatividade da operação de seleção por similaridade aos k-vizinhos mais próximos com a seleção tradicional, a seleção por similaridade por abrangência e com ela própria.

A Regra 4.9 mostra que a operação $\ddot{\sigma}$ não comuta entre si.

Regra 4.9. *A operação de seleção por similaridade* **`kNN`** *não comuta sobre sua composição, isto é,*

$$\begin{gathered}\ddot{\sigma}_{(S_1\ kNN(d_1,k_1)\ s_{q1})}(\ddot{\sigma}_{(S_2\ kNN(d_2,k_2)\ s_{q2})}(T)) \\ \neq \ddot{\sigma}_{(S_2\ kNN(d_2,k_2)\ s_{q2})}(\ddot{\sigma}_{(S_1\ kNN(d_1,k_1)\ s_{q1})}(T)).\end{gathered} \tag{4.20}$$

Mostramos esta regra com o seguinte contra-exemplo.

Exemplo: *"Selecione as 5 cidades mais próximas da cidade de São Carlos-SP, cuja latitude* $=-22.02$ *e longitude* $=47.89$*, e mais próximas da cidade de Araraquara-SP, cuja latitude* $=-21.79$ *e a longitude* $=48.18$*, considerando a distância Euclidiana* L_2*".*

$$\begin{gathered}\ddot{\sigma}_{(coordenada\ kNN(L_2,5)\ (-22.02,47.89))}(\ddot{\sigma}_{(coordenada\ kNN(L_2,5)\ (-21.79,48.18))}(\texttt{CidadeSaoCarlos})) \\ = \ddot{\sigma}_{(coordenada\ kNN(L_2,5)\ (-21.79,48.18))}(\ddot{\sigma}_{(coordenada\ kNN(L_2,5)\ (-22.02,47.89))}(\texttt{CidadeSaoCarlos})).\end{gathered} \tag{4.21}$$

Porém, após a execução das expressões algébricas pertencentes à Equivalência 4.21, o resultado delas não é o mesmo, conforme observado na Figura 4.2.

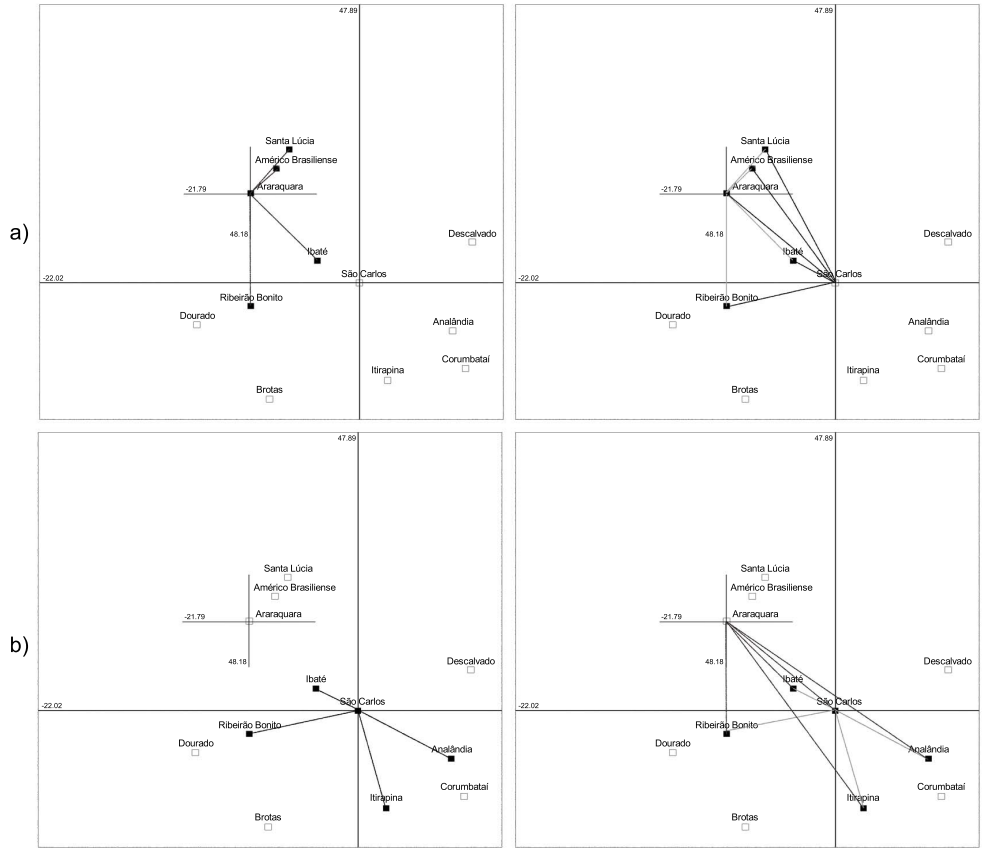

Figura 4.2: Ilustração da execução da consulta apresentada pela Equivalência 4.21. a) Consulta em que se executa primeiro o predicado por vizinhança com $s_{q1}=$ "Araraquara" e depois o predicado por vizinhança com $s_{q2}=$ "São Carlos" na relação `CidadeSaoCarlos`. b) Consulta em que se executa primeiro o predicado por vizinhança com $s_{q2}=$ "São Carlos" e depois o predicado por vizinhança com $s_{q1}=$ "Araraquara" na relação `CidadeSaoCarlos`.

Na Figura 4.2a, é executada primeiro a seleção por vizinhança com $s_{q1}=$ "Araraquara" na relação `CidadeSaoCarlos` e, sobre o resultado desta, a seleção por vizinhança com $s_{q2}=$ "São Carlos". O resultado desta consulta é {`Ribeirão Bonito, Ibaté, Araraquara, Américo Brasiliense, Santa Lúcia`}. Na Figura 4.2b, é executada primeiro a seleção por vizinhança com $s_{q2}=$ "São Carlos" na relação `CidadeSaoCarlos` e, em seguida, a seleção por vizinhança com $s_{q1}=$ "Araraquara". O resultado da última consulta é {`São Carlos, Analândia, Itirapina, Ribeirão Bonito, Ibaté`}.

A Regra 4.10 mostra a não comutatividade da operação $\ddot{\sigma}$ com σ.

Regra 4.10. *A operação de seleção por similaridade* **`kNN`** *e a operação de seleção tradicional não comutam entre si, isto é,*

$$\ddot{\sigma}_{(S\ kNN(d,k)\ s_q)}(\sigma_{(A\ \theta\ a)}(T)) \neq \sigma_{(A\ \theta\ a)}(\ddot{\sigma}_{(S\ kNN(d,k)\ s_q)}(T)), \tag{4.22}$$

no qual θ é um dos operadores $<,\leq,>,\geq,=$ ou $\neq$ e 'a' é uma constante.

Para mostrarmos a Regra 4.22 utilizamos o seguinte contra-exemplo.

Exemplo: *"Selecione as 5 cidades mais próximas da cidade de "São Carlos-SP", cuja latitude* $= -22.02$ *e longitude* $= 47.89$*, que possuem a latitude menor do que a latitude da cidade de "São Carlos-SP", considerando a distância Euclidiana* L_2*".*

$$\ddot{\sigma}_{(coordenada\ kNN(L_2,5)\ (-22.02,47.89))}(\sigma_{(lat<-22.02)}(\texttt{CidadeSaoCarlos})) = \sigma_{(lat<-22.02)}(\ddot{\sigma}_{(coordenada\ kNN(L_2,5)\ (-22.02,47.89))}(\texttt{CidadeSaoCarlos})). \quad (4.23)$$

Porém, após a execução das expressões algébricas pertencentes à Equivalência 4.23, o resultado delas não é o mesmo, conforme observado na Figura 4.3.

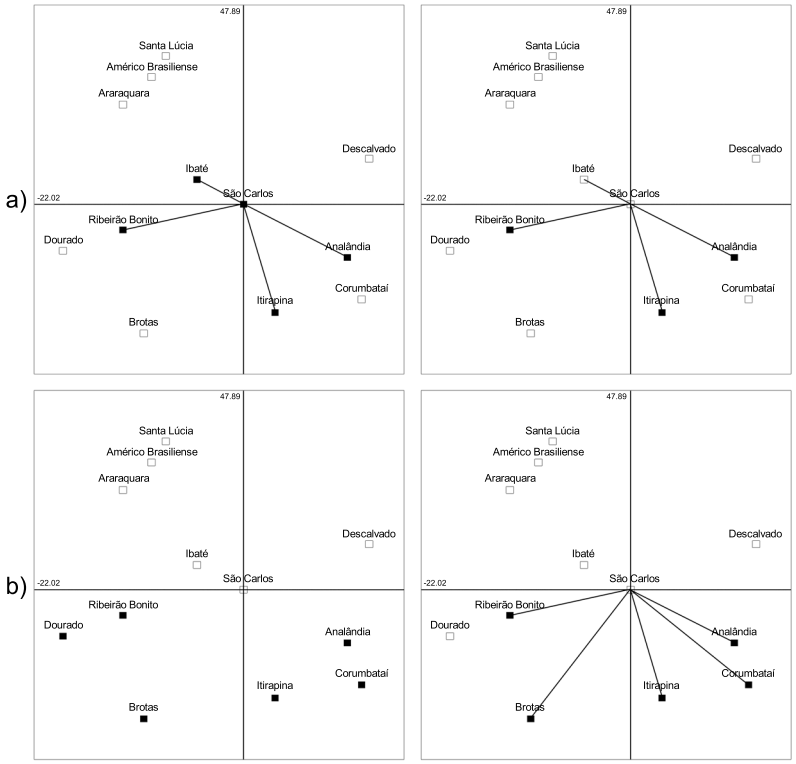

Figura 4.3: Ilustração da execução da consulta apresentada pela Equivalência 4.23. a) Consulta em que se executa primeiro o predicado por vizinhança e depois o tradicional na relação `CidadeSaoCarlos`. b) Consulta em que se executa primeiro o predicado tradicional e depois o predicado por vizinhança na relação `CidadeSaoCarlos`.

Na Figura 4.3a, é executada primeiro a seleção por vizinhança na relação `CidadeSaoCarlos` e, depois, a seleção tradicional. O resultado desta consulta é {`Itirapina, Analândia, Ribeirão Bonito`}. Na Figura 4.3b, é executada primeiro a seleção tradicional na relação `CidadeSaoCarlos` e, em seguida, a seleção por vizinhança. O resultado da última consulta é {`Corumbataí, Analândia, Ribeirão Bonito, Itirapina, Brotas`}.

A Regra 4.11 apresenta a não comutatividade da seleção por similaridade kNN com a seleção por similaridade R_q.

Regra 4.11. *A operação de seleção por similaridade kNN e a operação de seleção por similaridade R_q não comutam entre si, isto é,*

$$\begin{aligned}&\ddot{\sigma}_{(S_1\ kNN(d_1,k_1)\ s_{q1})}(\hat{\sigma}_{(S_2\ R_q(d_2,\xi_2)\ s_{q2})}(T))\\ &\neq \hat{\sigma}_{(S_2\ R_q(d_2,\xi_2)\ s_{q2})}(\ddot{\sigma}_{(S_1\ kNN(d_1,k_1)\ s_{q1})}(T)).\end{aligned} \tag{4.24}$$

Utilizamos o seguinte contra-exemplo para mostrar esta regra.

Exemplo: *"Selecione as 5 cidades mais próximas da cidade de São Carlos-SP, cuja latitude* $=-22.02$ *e longitude* $=47.89$*, que distam da cidade de "Araraquara", cuja latitude* $=-21.79$ *e a longitude* $=48.18$*, até, no máximo,* 0.3 *unidades de distância, considerando a distância Euclidiana* L_2*".*

$$\begin{aligned}&\ddot{\sigma}_{(coordenada\ kNN(L_2,5)\ (-22.02,47.89))}(\hat{\sigma}_{(coordenada\ R_q(L_2,0.3)\ (-21.79,48.18))}(\texttt{CidadeSaoCarlos}))\\ &=\hat{\sigma}_{(coordenada\ R_q(L_2,0.3)\ (-21.79,48.18))}(\ddot{\sigma}_{(coordenada\ kNN(L_2,5)\ (-22.02,47.89))}(\texttt{CidadeSaoCarlos})).\end{aligned} \tag{4.25}$$

As expressões algébricas pertencentes à Equivalência 4.25 foram executadas e o resultado apresentado por elas não é o mesmo, conforme observado na Figura 4.4.

Na Figura 4.4a, é executada primeiro a seleção por abrangência com s_{q1} = "Araraquara" na relação `CidadeSaoCarlos` e, depois, a seleção por vizinhança com s_{q2} = "São Carlos". O resultado desta consulta é {`Ribeirão Bonito, Ibaté, Araraquara, Américo Brasiliense, Santa Lúcia`}. Na Figura 4.4b, é executada primeiro a seleção por vizinhança com s_{q2} = "São Carlos" na relação `CidadeSaoCarlos` e, em seguida, a seleção por abrangência com s_{q1} = "Araraquara" sobre o resultado da primeira seleção. O resultado da última consulta é {`Ribeirão Bonito, Ibaté`}.

Em resumo, não existe comutatividade entre o predicado por vizinhança e qualquer outro tipo de predicado, inclusive ele próprio. Isto é, a execução em ordens distintas dos predicados de uma consulta complexa por vizinhança resulta em respostas diferentes. Portanto, para este tipo de consulta é necessário executar cada um dos predicados separadamente e retornar a interseção ou união dos resultados como resposta à consulta complexa. Algebricamente, a não-comutatividade da seleção por vizinhança pode ser expressa de duas maneiras, dependendo de como as condições complexas estão conectadas. Para condições complexas conectadas por `AND` ($\wedge$) temos:

$$\begin{array}{ccccc}\ddot{\sigma}_{c1}(T)\cap\ddot{\sigma}_{c2}(T) & \overset{1}{\nLeftrightarrow} & \ddot{\sigma}_{c1}(\ddot{\sigma}_{c2}(T)) & \overset{2}{\nLeftrightarrow} & \ddot{\sigma}_{c1\wedge c2}(T)\\ \Updownarrow_3 & & \not\Updownarrow_4 & & \not\Updownarrow_5\\ \ddot{\sigma}_{c2}(T)\cap\ddot{\sigma}_{c1}(T) & \overset{6}{\nLeftrightarrow} & \ddot{\sigma}_{c2}(\ddot{\sigma}_{c1}(T)) & \overset{7}{\nLeftrightarrow} & \ddot{\sigma}_{c2\wedge c1}(T),\end{array}$$

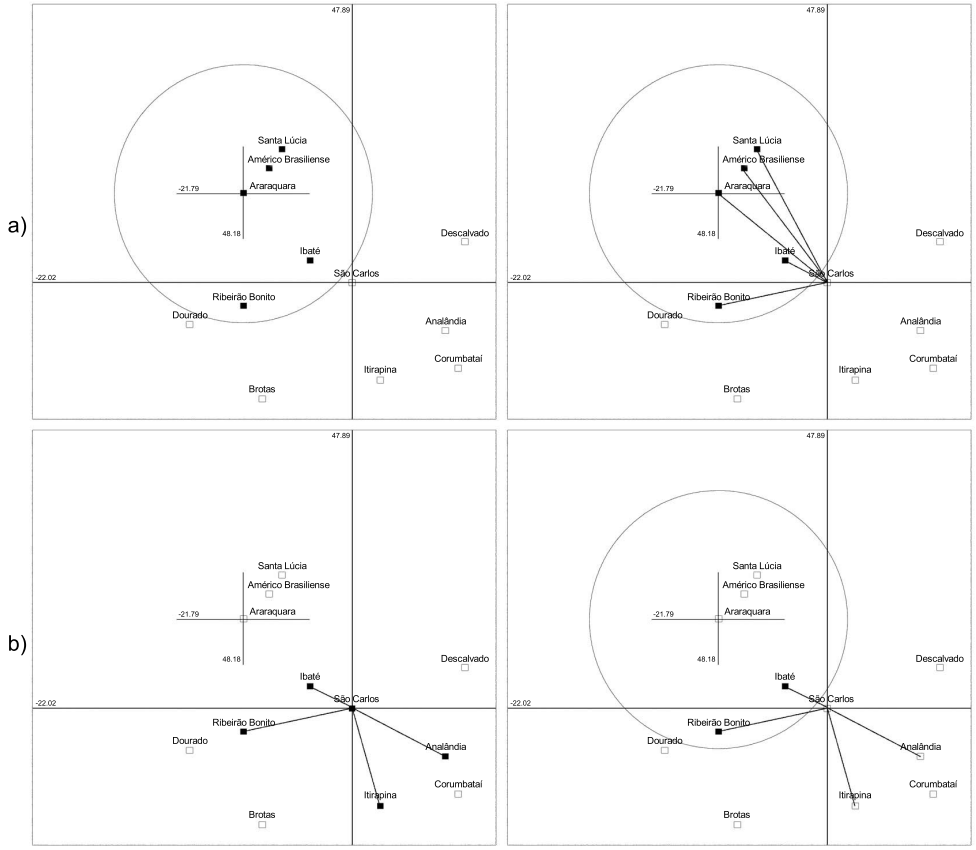

Figura 4.4: Ilustração da execução da consulta apresentada pela Equivalência 4.25. a) Consulta em que se executa primeiro o predicado por abrangência com $s_{q1} =$ "Araraquara" e depois o predicado por vizinhança com $s_{q2} =$ "São Carlos" na relação `CidadeSaoCarlos`. b) Consulta em que se executa primeiro o predicado por vizinhança com $s_{q2} =$ "São Carlos" e depois o predicado por abrangência com $s_{q1} =$ "Araraquara" na relação `CidadeSaoCarlos`.

e para as condições complexas conectadas por `OR` ($\vee$) temos:

$$\begin{array}{ccc} \ddot{\sigma}_{c1}(T) \cup \ddot{\sigma}_{c2}(T) & \overset{1}{\Leftrightarrow} & \ddot{\sigma}_{c1 \vee c2}(T) \\ \Updownarrow_2 & & \not\Updownarrow_3 \\ \ddot{\sigma}_{c2}(T) \cup \ddot{\sigma}_{c1}(T) & \overset{4}{\Leftrightarrow} & \ddot{\sigma}_{c2 \vee c1}(T), \end{array}$$

Com isto, a Regra 4.12 é proposta.

Regra 4.12. *O operador $\ddot{\sigma}$ não é comutativo com ele próprio nem com nenhum outro operador de seleção. Para condições complexas, deve-se executar cada um dos operadores separadamente e retornar a interseção, para condições conectadas por* `AND` *($\wedge$), ou união, para condições conectadas por* `OR` *($\vee$), dos resultados como resposta à consulta complexa,*

isto é, para condições conjuntivas, temos:

$$\begin{aligned}&\ddot{\sigma}_{(S_1\ kNN(d_1,k_1)\ s_{q1})}(T)\ \cap\ \ddot{\sigma}_{(S_2\ kNN(d_2,k_2)\ s_{q2})}(T)\\ =&\ddot{\sigma}_{(S_2\ kNN(d_2,k_2)\ s_{q2})}(T)\ \cap\ \ddot{\sigma}_{(S_1\ kNN(d_1,k_1)\ s_{q1})}(T),\end{aligned} \tag{4.26}$$

e para condições disjuntivas, temos:

$$\begin{aligned}&\ddot{\sigma}_{(S_1\ kNN(d_1,k_1)\ s_{q1})}(T)\ \cup\ \ddot{\sigma}_{(S_2\ kNN(d_2,k_2)\ s_{q2})}(T)\\ =&\ddot{\sigma}_{(S_2\ kNN(d_2,k_2)\ s_{q2})}(T)\ \cup\ \ddot{\sigma}_{(S_1\ kNN(d_1,k_1)\ s_{q1})}(T),\end{aligned} \tag{4.27}$$

Demostração. Usando a Definição 4.3, mostramos que:

$$\begin{aligned}&\ddot{\sigma}_{(S_1\ kNN(d_1,k_1)\ s_{q1})}(T)\ \cap\ \ddot{\sigma}_{(S_2\ kNN(d_2,k_2)\ s_{q2})}(T)\\ =&\{t_i \in T \mid \forall\, t_{i1} \in [T - T_1'], |T_1'| = k, d(t_i(S), s_{q1}) \leq d(t_{i1}(S), s_{q1})\} \cap\\ &\{t_i \in T \mid \forall\, t_{i2} \in [T - T_2'], |T_2'| = k, d(t_i(S), s_{q2}) \leq d(t_{i2}(S), s_{q2})\}\\ =&\{t_i \in T \mid \forall\, t_{i1} \in [T - T_1'], |T_1'| = k, d(t_i(S), s_{q1}) \leq d(t_{i1}(S), s_{q1}) \text{ and}\\ &\qquad \forall\, t_{i2} \in [T - T_2'], |T_2'| = k, d(t_i(S), s_{q2}) \leq d(t_{i2}(S), s_{q2})\}\\ =&\{t_i \in T \mid \forall\, t_{i2} \in [T - T_2'], |T_2'| = k, d(t_i(S), s_{q2}) \leq d(t_{i2}(S), s_{q2}) \text{ and}\\ &\qquad \forall\, t_{i1} \in [T - T_1'], |T_1'| = k, d(t_i(S), s_{q1}) \leq d(t_{i1}(S), s_{q1})\}\\ =&\{t_i \in T \mid \forall\, t_{i2} \in [T - T_2'], |T_2'| = k, d(t_i(S), s_{q2}) \leq d(t_{i2}(S), s_{q2})\} \cap\\ &\{t_i \in T \mid \forall\, t_{i1} \in [T - T_1'], |T_1'| = k, d(t_i(S), s_{q1}) \leq d(t_{i1}(S), s_{q1})\}\\ =&\ddot{\sigma}_{(S_2\ kNN(d_2,k_2)\ s_{q2})}(T)\ \cap\ \ddot{\sigma}_{(S_1\ kNN(d_1,k_1)\ s_{q1})}(T)\end{aligned}$$

A prova para as condições disjuntivas é realizada de maneira análoga.

□

Existe um caso especial, apresentado no trabalho de Traina-Jr. et al. (2006b), em que a seleção por vizinhança é comutativa com a seleção por abrangência. Isto ocorre quando as seleções são executadas sobre o mesmo elemento de consulta, isto é, quando $s_{q1} = s_{q2} = s_q$. Os autores explicam que realizar a conjunção dos predicados *range* e *kNN* é equivalente a interseção dos resultados obtidos executando a seleção por abrangência $\hat{\sigma}$ e aos vizinhos mais próximos $\ddot{\sigma}$ e também é equivalente à execução do algoritmo *kAndRange*(θ, s_q, k, ξ), isto é,

$$\begin{aligned}&\hat{\sigma}_{(S\ R_q(d,\xi)\ s_q)}(T) \cap \ddot{\sigma}_{(S\ kNN(d,k)\ s_q)}(T) \Leftrightarrow\\ &\hat{\sigma}_{(S\ R_q(d,\xi)\ s_q)}(\ddot{\sigma}_{(S\ kNN(d,k)\ s_q)})(T) \Leftrightarrow kAndRange(\theta, s_q, k, \xi);\end{aligned} \tag{4.28}$$

e que realizar a disjunção dos predicados *range* e *kNN* é equivalente a união dos dos resultados obtidos executando a seleção por abrangência e por vizinhança e também é

equivalente à execução do algoritmo *kOrRange*(θ, s_q, k, ξ) isto é,

$$\hat{\sigma}_{(S\ R_q(d,\xi)\ s_q)}(T) \cup \ddot{\sigma}_{(S\ kNN(d,k)\ s_q)}(T) \Leftrightarrow kOrRange(\theta, s_q, k, \xi); \tag{4.29}$$

em que θ passado como parâmetro nos algoritmos *kAndRange* e *kOrRange* indica se o predicado é por abrangência e por vizinhança mais próxima ou se o predicado é por abrangência inversa e por vizinhos mais distante. Este caso específico apresentado por Traina-Jr. et al. (2006b) é importante na otimização de consultas por similaridade pois eles varrem o conjunto de dados apenas uma vez, enquanto a interseção dos predicados precisa varrer duas vezes o conjunto de dados para responder à mesma consulta por similaridade complexa. Portanto, o custo de E/S e de CPU diminuem.

As Regras 4.13, 4.14, 4.15, 4.16 e 4.17 envolvem os operadores binários da álgebra relacional.

A álgebra relacional permite que σ seja distribuído sobre os operadores binários diferentes ramos do plano de consulta. Assim, a seguinte expressão, usando o operador de união ($\cup$), é válida desde que as relações T_1 e T_2 sejam compatíveis em domínio:

$$\sigma_{c_1}(T_1 \cup T_2) = \sigma_{c_1}(T_1) \cup \sigma_{c_2}(T_2). \tag{4.30}$$

No entanto, se trocarmos σ por $\ddot{\sigma}$, a Expressão 4.30 deixa de ser válida, originando a Regra 4.13.

Regra 4.13. *O operador de seleção por similaridade* **`kNN`** *não é distributivo sobre união.*

$$\ddot{\sigma}_{(S\ kNN(d,k)\ s_q)}(T_1 \cup T_2) \neq \ddot{\sigma}_{(S\ kNN(d,k)\ s_q)}(T_1) \cup \ddot{\sigma}_{(S\ kNN(d,k)\ s_q)}(T_2). \tag{4.31}$$

O seguinte contra-exemplo mostra esta regra.

Exemplo: *"Selecione as 5 cidades mais próximas da cidade de São Carlos-SP, cuja latitude* $=-22.02$ *e longitude* $=47.89$*, que pertençam à relação "CidadeSaoCarlos"* ***ou*** *à relação "CidadeAraraquara", considerando a distância Euclidiana* L_2*".*

$$\begin{aligned}
&\ddot{\sigma}_{(coordenada\ kNN(L_2,5)\ (-22.02,47.89))}(\texttt{CidadeAraraquara} \cup \texttt{CidadeSaoCarlos}) \\
&= \ddot{\sigma}_{(coordenada\ kNN(L_2,5)\ (-22.02,47.89))}(\texttt{CidadeAraraquara}) \cup \\
&\quad \ddot{\sigma}_{(coordenada\ kNN(L_2,5)\ (-22.02,47.89))}(\texttt{CidadeSaoCarlos}).
\end{aligned} \tag{4.32}$$

A Figura 4.5 mostra que o resultado da execução das duas expressões algébricas pertencentes à Equivalência 4.32 não é o mesmo.

Na Figura 4.5a, é executada primeiro a união das relações `CidadeAraraquara` e `CidadeSaoCarlos` e depois a seleção por vizinhança. O resultado desta consulta é {`São Carlos, Ibaté, Itirapina, Analândia, Ribeirão Bonito`}. Na Figura 4.5b, é executada primeiro a seleção por vizinhança em cada uma das relações

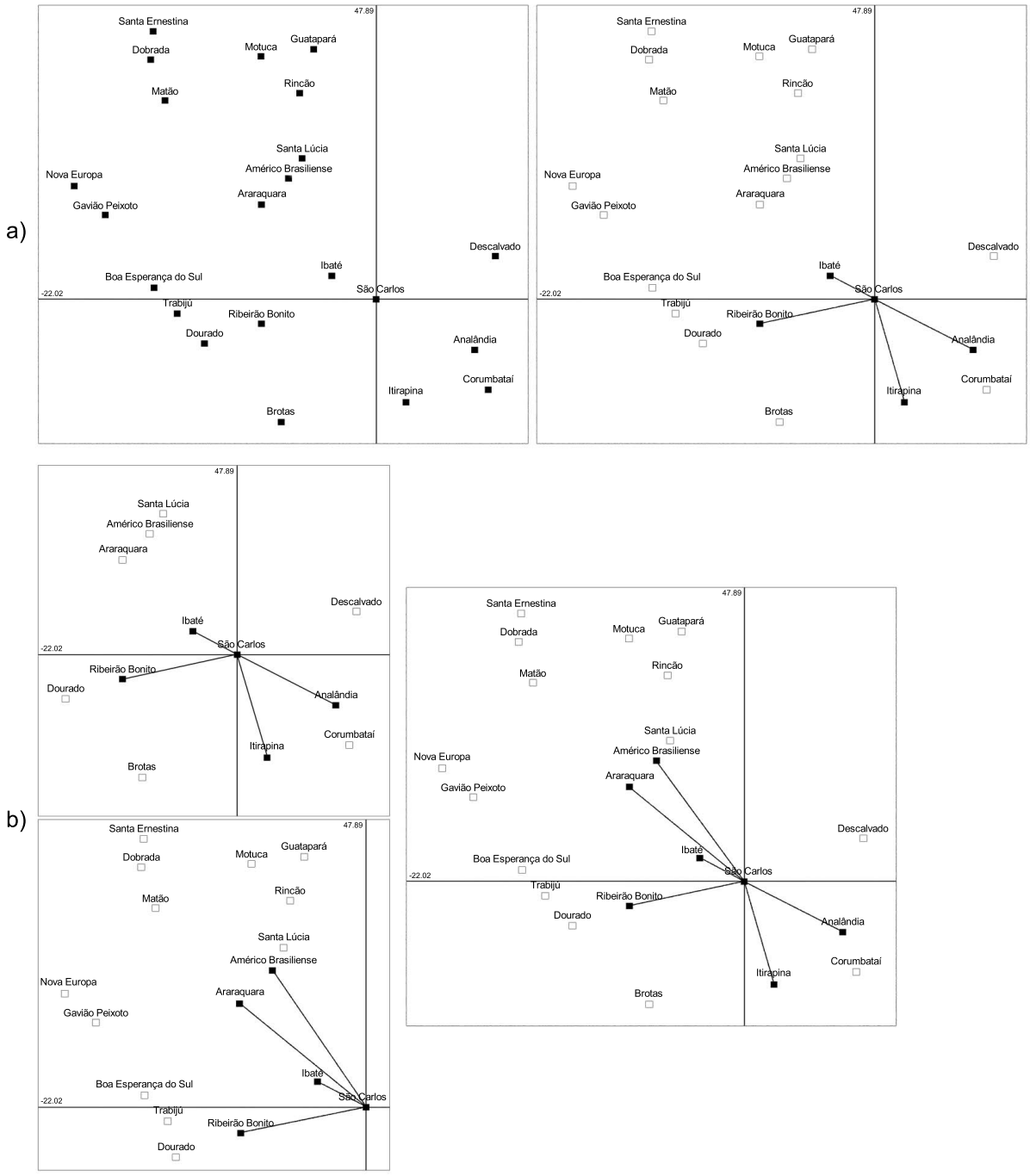

Figura 4.5: Ilustração da execução da consulta apresentada pela Equivalência 4.32. a) Consulta em que se executa primeiro a união das relações `CidadeSaoCarlos` e `CidadeAraraquara` e, em seguida, a seleção por vizinhança sobre o seu resultado. b) Consulta em que se executa primeiro a seleção por vizinhança nas relações `CidadeSaoCarlos` e `CidadeAraraquara` e, depois, a união de seus resultados.

`CidadeAraraquara` e `CidadeSaoCarlos` e, em seguida, a união dos resultados. O resultado desta união é {`São Carlos, Analândia, Ribeirão Bonito, Ibaté, Itirapina, Araraquara, Américo Brasiliense`}.

No caso da diferença $(-)$, σ deve ser distribuído para o primeiro ramo do plano de consulta e, opcionalmente, para o segundo ramo, na álgebra relacional, conforme é mostrado na seguinte expressão. Esta regra também requer que as relações T_1 e T_2 sejam compatíveis em domínio.

$$\sigma_{c_1}(T_1 \ - \ T_2) = \sigma_{c_1}(T_1) \ - \ T_2 = \sigma_{c_1}(T_1) \ - \ \sigma_{c_1}(T_2). \tag{4.33}$$

Entretanto, se a seleção tradicional for trocada pela seleção por similaridade aos vizinhos mais próximos, a Expressão 4.33 torna-se inválida, originando a Regra 4.14.

Regra 4.14. *A operação de seleção por similaridade* ***kNN*** *não pode ser distribuída sobre a diferença.*

$$\begin{aligned} &\ddot{\sigma}_{(S\ kNN(d,k)\ s_q)}(T_1 \ - \ T_2) \\ &\neq \ \ddot{\sigma}_{(S\ kNN(d,k)\ s_q)}(T_1) \ - \ T_2 \\ &\neq \ \ddot{\sigma}_{(S\ kNN(d,k)\ s_q)}(T_1) \ - \ \ddot{\sigma}_{(S\ kNN(d,k)\ s_q)}(T_2). \end{aligned} \tag{4.34}$$

Esta regra é mostrada pelo seguinte contra-exemplo.

Exemplo: *"Selecione as 5 cidades mais próximas da cidade de São Carlos-SP, cuja latitude* $= -22.02$ *e longitude* $= 47.89$*, que pertençam à relação "CidadeAraraquara"* ***mas não*** *à "CidadeSaoCarlos", considerando a distância Euclidiana* L_2*".*

$$\begin{aligned} &\ddot{\sigma}_{(coordenada\ kNN(L_2,5)\ (-22.02,47.89))}(\texttt{CidadeAraraquara} \ - \ \texttt{CidadeSaoCarlos}) \\ &= \ddot{\sigma}_{(coordenada\ kNN(L_2,5)\ (-22.02,47.89))}(\texttt{CidadeAraraquara}) \ - (\texttt{CidadeSaoCarlos}) \\ &= \ddot{\sigma}_{(coordenada\ kNN(L_2,5)\ (-22.02,47.89))}(\texttt{CidadeAraraquara}) \ - \\ &\quad \ddot{\sigma}_{(coordenada\ kNN(L_2,5)\ (-22.02,47.89))}(\texttt{CidadeSaoCarlos}). \end{aligned} \tag{4.35}$$

O resultado da execução das expressões algébricas da Equivalência 4.35 não é o mesmo, conforme observado na Figura 4.6.

Na Figura 4.6a, é executada primeiro a diferença das relações `CidadeAraraquara` e `CidadeSaoCarlos` e, depois, a seleção por vizinhança. O resultado desta consulta é {`Boa Esperança do Sul, Guatapará, Motuca, Rincão, Trabiju`}. Na Figura 4.6b, é executada primeiro a seleção por vizinhança na relação `CidadeAraraquara` e, depois, é realizada a diferença de seu resultado com a relação `CidadeSaoCarlos`. O resultado desta diferenciação é vazio $(\varnothing)$, isto é, todas as tuplas retornadas na seleção por vizinhança pertencem à relação `CidadeSaoCarlos`. Na Figura 4.6c, é primeiro executada a seleção por vizinhança nas relações `CidadeAraraquara` e `CidadeSaoCarlos` e, em seguida, a

Figura 4.6: Ilustração da execução da consulta apresentada pela Equivalência 4.35. a) Consulta que executa primeiro a diferença das relações `CidadeSaoCarlos` e `CidadeAraraquara` e, em seguida, a seleção por vizinhança sobre o resultado. b) Consulta que executa primeiro a seleção por vizinhança na relação `CidadeAraraquara` e, depois, a diferença de seu resultado com a relação `CidadeSaoCarlos`. c) Consulta que executa primeiro a seleção por vizinhança nas relações `CidadeAraraquara` e `CidadeSaoCarlos` e, depois, a diferença de seus resultados.

diferença de seus resultados. O resultado desta diferenciação é {`Américo Brasiliense, Araraquara`}.

A seleção tradicional deve ser distribuída sobre a interseção ($\cap$) para um dos ramos da árvore e, opcionalmente, para ambos, na álgebra relacional, gerando a seguinte expressão. Esta regra requer que as relações T_1 e T_2 sejam compatíveis em domínio.

$$\sigma_{c_1}(T_1 \cap T_2) = \sigma_{c_1}(T_1) \cap T_2 = T_1 \cap \sigma_{c_1}(T_2) = \sigma_{c_1}(T_1) \cap \sigma_{c_1}(T_2). \tag{4.36}$$

Porém, quando utilizamos o $\ddot{\sigma}$ no lugar do σ, esta expressão torna-se inválida e dá origem a Regra 4.15.

Regra 4.15. *A operação de seleção por similaridade* **`kNN`** *não pode ser distribuída sobre o operador de interseção.*

$$\begin{aligned}
&\ddot{\sigma}_{(S\,kNN(d,k)\,s_q)}(T_1 \cap T_2) \\
&\neq \ddot{\sigma}_{(S\,kNN(d,k)\,s_q)}(T_1) \cap T_2 \\
&\neq T_1 \cap \ddot{\sigma}_{(S\,kNN(d,k)\,s_q)}(T_2) \\
&\neq \ddot{\sigma}_{(S\,kNN(d,k)\,s_q)}(T_1) \cap \ddot{\sigma}_{(S\,kNN(d,k)\,s_q)}(T_2).
\end{aligned} \tag{4.37}$$

Mostramos esta regra por meio do seguinte contra-exemplo.

Exemplo: *"Selecione as 5 cidades mais próximas da cidade de São Carlos-SP, com latitude* $= -22.02$ *e longitude* $= 47.89$*, que pertençam às relações "CidadeAraraquara"* ***e*** *a "CidadeSaoCarlos", considerando a distância Euclidiana* L_2*".*

$$\begin{aligned}
&\ddot{\sigma}_{(coordenada\,kNN(L_2,5)\,(-22.02,47.89))}(\texttt{CidadeAraraquara} \cap \texttt{CidadeSaoCarlos}) \\
&= \ddot{\sigma}_{(coordenada\,kNN(L_2,5)\,(-22.02,47.89))}(\texttt{CidadeAraraquara}) \cap \texttt{CidadeSaoCarlos} \\
&= \texttt{CidadeAraraquara} \cap \ddot{\sigma}_{(coordenada\,kNN(L_2,5)\,(-22.02,47.89))}(\texttt{CidadeSaoCarlos}) \\
&= \ddot{\sigma}_{(coordenada\,kNN(L_2,5)\,(-22.02,47.89))}(\mathbf{CidadeAraraquara}) \cap \\
&\quad \ddot{\sigma}_{(coordenada\,kNN(L_2,5)\,(-22.02,47.89))}(\texttt{CidadeSaoCarlos}).
\end{aligned} \tag{4.38}$$

O resultado da execução das expressões algébricas da Equivalência 4.38 é apresentado na Figura 4.7.

Na Figura 4.7a, é executada primeiro a interseção das relações `CidadeAraraquara` e `CidadeSaoCarlos` e, depois, a seleção por vizinhança. O resultado desta consulta é {`São Carlos, Ribeirão Bonito, Ibaté, Araraquara, Américo Brasiliense`}. Na Figura 4.6b, é primeiro executada a seleção por vizinhança nas relações `CidadeAraraquara` e `CidadeSaoCarlos` e, em seguida, a interseção de seus resultados. O resultado desta interseção é {`Ribeirão Bonito, São Carlos, Ibaté`}. A Figura 4.7 do Exemplo 4.2.3 mostra que uma das equivalências deste exemplo não é valida. Portanto, a Equi-

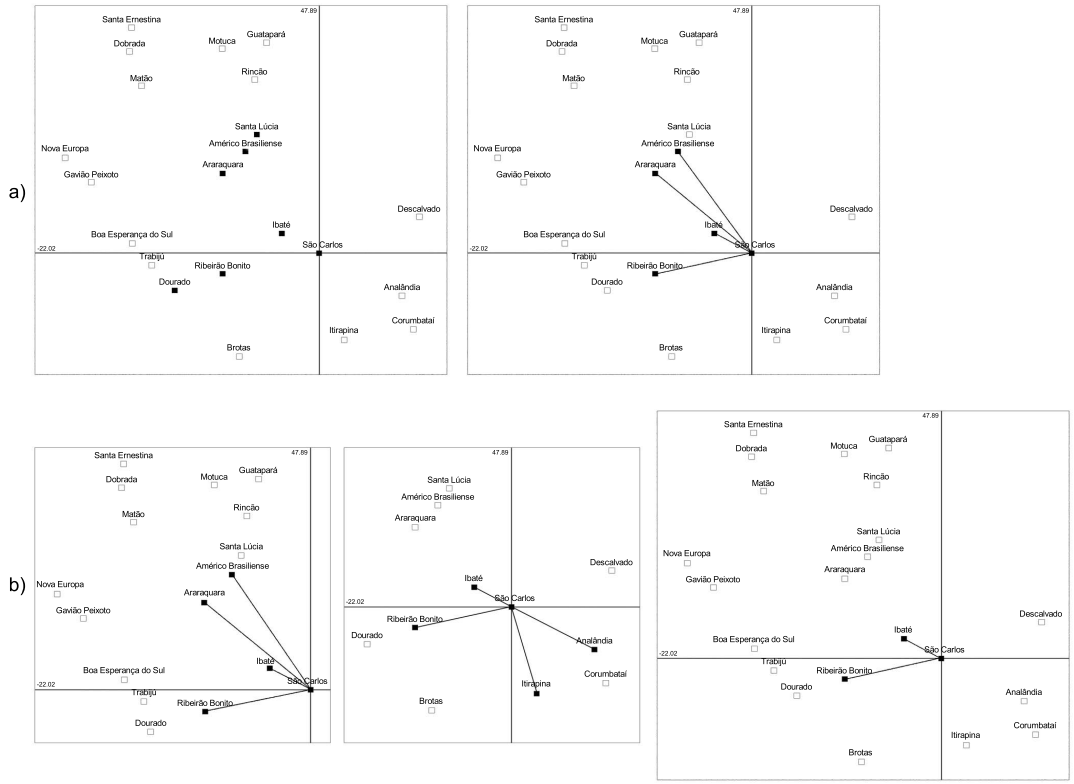

Figura 4.7: Ilustração da execução da consulta apresentada pela Equivalência 4.38. a) Consulta que executa primeiro a interseção das relações `CidadeSaoCarlos` e `CidadeAraraquara` e, em seguida, a seleção por vizinhança do resultado. b) Consulta que executa primeiro a seleção por vizinhança nas relações `CidadeAraraquara` e `CidadeSaoCarlos` e, depois, a interseção dos resultados.

valência 4.38 também não é válida. Assim, a Regra 4.15 mostra que a seleção por vizinhança não é distributiva sobre a interseção.

No caso dos operadores binários junção ($\bowtie$) e produto cartesiano ($\times$), σ deve ser distribuído para o ramo do plano de consulta que tem todos os atributos mencionados na condição c, isto é, a seleção tradicional somente pode ser distribuída para uma relação se todos os atributos mencionados em c pertencerem a esta relação.

Para o produto cartesiano, a seguinte expressão é válida, considerando a álgebra relacional.

$$\sigma_c(T_1 \times T_2) = \sigma_c(T_1) \times T_2 = T_1 \times \sigma_c(T_2) = \sigma_c(T_1) \times \sigma_c(T-2). \tag{4.39}$$

No entanto, se a seleção σ é substituída pela seleção $\ddot{\sigma}$, a expressão deixa de ser válida, originando a Regra 4.16

Regra 4.16. *A seleção por similaridade* `NN` *não é distributiva sobre o produto cartesiano.*

$$\begin{aligned}
&\ddot{\sigma}_{(S\,kNN(d,k)\;s_q)}(T_1 \times T_2)\\
&\neq \ddot{\sigma}_{(S\,kNN(d,k)\;s_q)}(T_1) \times T_2\\
&\neq T_1 \times \ddot{\sigma}_{(S\,kNN(d,k)\;s_q)}(T_2)\\
&\neq \ddot{\sigma}_{(S\,kNN(d,k)\;s_q)}(T_1) \times \ddot{\sigma}_{(S\,kNN(d,k)\;s_q)}(T_2).
\end{aligned} \tag{4.40}$$

Esta regra pode ser mostrada com o seguinte contra-exemplo.

Exemplo: *"Selecione os 5 pares de cidades mais próximas da cidade de São Carlos-SP, cuja latitude* $= -22.02$ *e longitude* $= 47.89$*, pertencentes às relações "CidadeAraraquara" e a "CidadeSaoCarlos", considerando a distância Euclidiana* L_2*".*

$$\begin{aligned}
&\underbrace{\ddot{\sigma}_{(coordenada\;kNN(L_2,5)\;(-22.02,47.89))}(\texttt{CidadeAraraquara} \times \texttt{CidadeSaoCarlos})}_{1}\\
&= \underbrace{\ddot{\sigma}_{(coordenada\;kNN(L_2,5)\;(-22.02,47.89))}(\texttt{CidadeAraraquara}) \times \texttt{CidadeSaoCarlos}}_{2}\\
&= \underbrace{\texttt{CidadeAraraquara} \times \ddot{\sigma}_{(coordenada\;kNN(L_2,5)\;(-22.02,47.89))}(\texttt{CidadeSaoCarlos})}_{3}\\
&= \ddot{\sigma}_{(\texttt{coordenada}\;kNN(L_2,5)\;(-22.02,47.89))}(\texttt{CidadeAraraquara}) \times\\
&\quad \underbrace{\ddot{\sigma}_{(coordenada\;kNN(L_2,5)\;(-22.02,47.89))}(\texttt{CidadeSaoCarlos})}_{4}.
\end{aligned} \tag{4.41}$$

A Expressão Algébrica 1 da Equivalência 4.41 retorna 204 tuplas ($|17| * |12|$); a Expressão 2 retorna 60 tuplas ($|5| * |12|$); a Expressão 3 retorna 85 tuplas ($|17| * |5|$); e a 4 retorna 25 tuplas ($|5| * |5|$).

Na álgebra relacional, a Expressão 4.39 também é válida se o $\times$ for trocado pelo $\bowtie$.

$$\sigma_c(T_1 \bowtie T_2) = \sigma_c(T_1) \bowtie T_2 = T_1 \bowtie \sigma_c(T_2). \tag{4.42}$$

Trocando o σ pelo $\ddot{\sigma}$ na Expressão 4.42, esta expressão torna-se inválida, dando origem a Regra 4.17.

Regra 4.17. *A seleção por similaridade* `kNN` *não é distribuída sobre a junção.*

$$\begin{aligned}
&\ddot{\sigma}_{(S\,kNN(d,k)\;s_q)}(T_1 \bowtie T_2)\\
&\neq \ddot{\sigma}_{(S\,kNN(d,k)\;s_q)}(T_1) \bowtie T_2\\
&\neq T_1 \bowtie \ddot{\sigma}_{(S\,kNN(d,k)\;s_q)}(T_2).
\end{aligned} \tag{4.43}$$

Esta regra é mostrada utilizando o seguinte contra-exemplo.

Exemplo: *"Selecione as 5 cidades mais próximas da cidade de São Carlos-SP, cuja latitude* $= -22.02$ *e longitude* $= 47.89$*, pertencentes às relações "CidadeAraraquara" e "CidadeSaoCarlos", considerando a distância Euclidiana* L_2*".*

$$\underbrace{\ddot{\sigma}_{(coordenada\ kNN(L_2,5)\ (-22.02,47.89))}(\texttt{CidadeAraraquara} \bowtie \texttt{CidadeSaoCarlos})}_{1}$$
$$= \underbrace{\texttt{CidadeAraraquara} \bowtie \ddot{\sigma}_{(coordenada\ kNN(L_2,5)\ (-22.02,47.89))}(\texttt{CidadeSaoCarlos})}_{2}. \quad (4.44)$$

A execução das expressões algébricas pertencentes a Equivalência 4.44 produz como resultado: a Expressão Algébrica 1 retorna 5 tuplas, sendo elas {`São Carlos, Ribeirão Bonito, Ibaté, Araraquara, Américo Brasiliense`}; e a Expressão Algébrica 2 retorna 3 tuplas, sendo elas: {`São Carlos, Ribeirão Bonito, Ibaté`}.

De maneira alternativa, se a condição de seleção c for conjuntiva e puder ser escrita como c_1 and c_2, na qual a condição c_1 envolve apenas atributos de T_1 e a condição c_2 envolve apenas atributos de T_2, a seguinte expressão é válida.

$$\sigma_{c_1 \wedge c_2}(T_1 \bowtie T_2) = \sigma_{c_1}(T_1) \bowtie \sigma_{c_2}(T_2). \quad (4.45)$$

Porém, se trocarmos a seleção tradicional pela seleção por similaridade aos vizinhos mais próximos, a Expressão 4.45 torna-se inválida.

$$\ddot{\sigma}_{((S_1\ kNN(d,k)\ s_q) \wedge (S_2\ kNN(d,k)\ s_q))}(T_1 \bowtie T_2)$$
$$\neq \ddot{\sigma}_{(S_1\ kNN(d,k)\ s_q)}(T_1) \bowtie \ddot{\sigma}_{(S_2\ kNN(d,k)\ s_q)}(T_2). \quad (4.46)$$

Pelas Regras 4.7 e 4.17 mostramos que a propriedade distributiva, válida para a seleção tradicional, não vale para a seleção por vizinhança. Então, a Equivalência 4.46 completa a Equivalência 4.43 da Regra 4.17.

Como a seleção por vizinhança aceita somente a Regra 4.12 e quatro casos especiais (Regras 4.2.3, 4.2.3, 4.28 e 4.29), essas seleções não podem utilizar os algoritmos de otimização de consultas existentes no SGBD relacional para otimizar suas consultas. Assim, algoritmos de otimização específicos para este tipo de consulta devem ser implementados no módulo de Reescrita de Planos do otimizador de consultas, como, por exemplo, os algoritmos *kAndRange* e *kOrRange* (Traina-Jr. et al., 2006b) criados especificamente para tratar a comutatividade das consultas por abrangência e por vizinhança sobre o mesmo elemento de consulta. Outra observação importante é nenhuma das regras aplicadas aos operadores binários da álgebra relacional funcionaram com a seleção por vizinhança. Portanto, quando a seleção por similaridade aos k-vizinhos mais próximos envolverem operadores binários nenhum tipo de otimização com relação a $\ddot{\sigma}$ pode ser realizada.

Somente quando $\ddot{\sigma}$ envolver operadores unários, este tipo de consulta poderá sofrer algum tipo de otimização, utilizando as cinco regras válidas (uma regra e quatro casos especiais).

4.3 Considerações finais

Neste capítulo, foi apresentada a álgebra por similaridade, para os operadores unários, que foi incorporada ao módulo de Reescrita de Planos do otimizador de consultas do SIREN. O objetivo da álgebra proposta é permitir que as consultas por similaridade sejam otimizadas em SGBDs relacionais.

As regras algébricas válidas para as operações de similaridade também foram apresentadas. Pelas regras da seleção por abrangência (ou por abrangência inversa) provou-se que esta operação divide as mesmas regras de equivalência que as seleções tradicionais e, com isto, os algoritmos de otimização de consultas tradicionais também podem ser usados para otimizar consultas por similaridade. As regras algébricas para a seleção por vizinhança também foram apresentadas. Comprovou-se que esta seleção não atende a maioria das regras de equivalência aceita pelos operadores tradicionais. Com isto, algoritmos específicos para a otimização de consultas por vizinhança devem ser criados para que estas consultas possam ser otimizadas.

Capítulo

5

Otimizador de consultas por similaridade

5.1 Introdução

As consultas por similaridade são as mais adequadas para manipular e recuperar dados em domínios complexos, conforme apresentado no Capítulo 3. Para que um SGBD relacional dê suporte a dados complexos, ele precisa aceitar, além das consultas tradicionais, as consultas por similaridade. Porém, os SGBDs não estão aptos a realizar este tipo de consulta.

Com o intuito dos SGBDs relacionais responderem a estas consultas, Barioni et al. (2006) propuseram um mecanismo, denominado SIREN (*SImilarity Retrieval ENgine*), que permite a realização de consultas por similaridade em SQL. O SIREN possui, como seu principal componente, o compilador de consultas. Porém, ele não processa eficientemente este tipo de consulta pois não possui o otimizador de consultas em sua estrutura. O funcionamento deste mecanismo é descrito, resumidamente, no Apêndice A.

Neste trabalho de mestrado foi desenvolvido o otimizador de consultas por similaridade para o SIREN, que é apresentado na Seção 5.2. Na Seção 5.3, é mostrado um exemplo de otimização de consulta por similaridade utilizando o novo otimizador do SIREN. As considerações finais são apresentadas na Seção 5.4.

As definições e propriedades apresentadas neste capítulo sobre o otimizador de consultas por similaridade foram publicadas em Ferreira et al. (2007).

5.2 Otimizador de consultas por similaridade

O *SImilarity Retrieval ENgine* (SIREN) é um serviço implementado entre o SGBD relacional e os programas de aplicação, que intercepta todo comando SQL enviado pela aplicação para verificar se existe ou não alguma construção sintática relacionada às operações por similaridade. Como a primeira versão do SIREN (veja Apêndice A) possui somente o compilador de consultas, ele interpreta e executa as consultas por similaridade mas não é capaz de otimizá-las. Assim, o otimizador de consultas foi adicionado à sua estrutura, a fim de que ele possa otimizar as consultas por similaridade de maneira análoga à otimização de consultas no SGBD relacional. O nova arquitetura do SIREN, com o otimizador de consultas por similaridade, é mostrada na Figura 5.1.

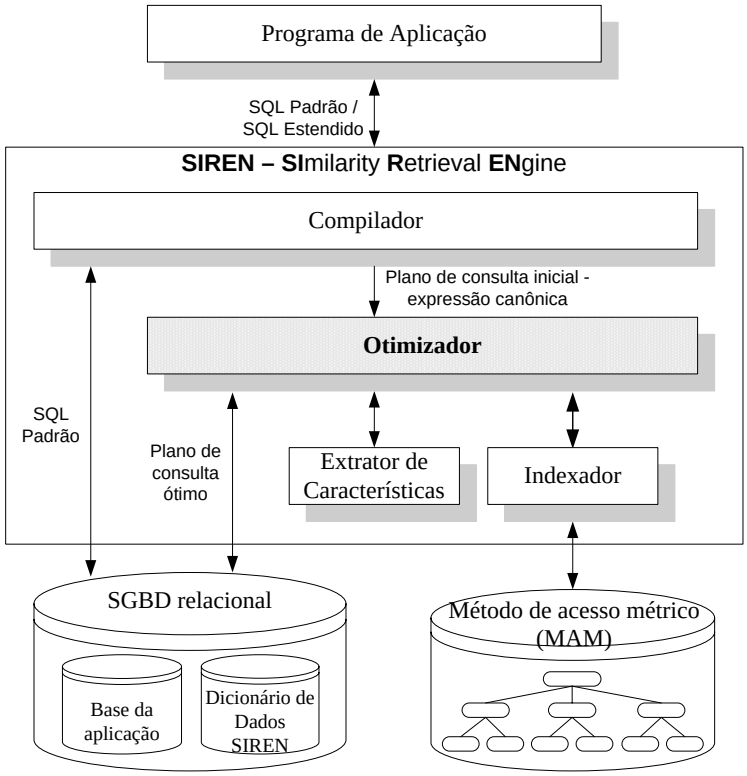

Figura 5.1: Arquitetura do SIREN com o otimizador de consultas.

Após o SIREN interceptar a consulta SQL, ela é analisada pelo compilador para saber se existe alguma construção sintática relacionada às operações por similaridade, isto é, se a consulta é composta somente por predicados tradicionais, somente por predicados por similaridade ou por ambos os predicados.

Se a consulta possui somente predicados tradicionais, ela é enviada ao SGBD relacional que, após processá-la, retorna o resultado para a aplicação. Assim, o SIREN é transparente no processamento das consultas tradicionais.

No entanto, caso a consulta possua somente predicados por similaridade ou ambos os predicados, o compilador faz sua analise léxica, sintática e semântica. Após executar todas

as análises com êxito, o compilador considera a consulta válida e gera o plano de consulta inicial. Este plano, também chamado de expressão canônica, é usado como entrada no otimizador. A Figura 5.2 mostra a arquitetura do otimizador do SIREN.

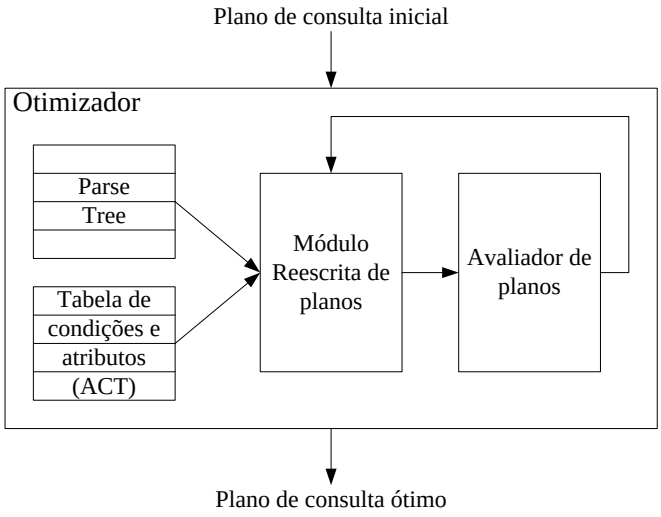

Figura 5.2: Arquitetura do otimizador do SIREN.

No otimizador, o plano de consulta inicial é analisado e as operações, a ordem de execução e os custos de cada condição são armazenados na estrutura ***Parse Tree***. Na Figura 5.3 são apresentados os registros da estrutura *Parse tree* (Figura 5.3a) e da Tabela de Condições e Atributos (Figura 5.3b).

(a) Parse tree:

Operação	Ramo Direito (i)	Ramo Esquerdo (i)	Referência ACT	Custo

(b) Tabela de Atributos e Condições (ACT):

Atributo	Tipo de Dado	Operador	Valor	k/ξ	Função de Distância	Lista de Empate	Operador de Agregação

Figura 5.3: Estrutura da (a) *Parse Tree* e da (b) Tabela de Condições e Atributos (ACT) do otimizador de consultas do SIREN.

Nesta estrutura, o campo **"Operação"** indica o tipo de operação executada; os campos **"Ramo Direito(i)"** e **"Ramo Esquerdo(i)"** são ponteiros para outras operações na *Parse Tree*; o campo **"Referência ACT"** é um ponteiro para a Tabela de Condições e Atributos (*Attributes and Conditions Table* - ACT); e o campo **"Custo"** armazena o custo estimado atribuído a cada operador.

A estrutura **Tabela de Condições e Atributos** (**ACT**), mostrada na Figura 5.3b, armazena o predicado de cada operação apresentada na *Parse Tree*. A ACT, neste trabalho, trata somente os predicados das consultas. A *Parse Tree* completa é tratada pelo SGBD relacional.

O armazenamento dos predicados tradicionais e por similaridade na estrutura ACT é realizado de maneira diferente. Para os **predicados tradicionais**, somente são preenchidos os campos:

- **"Atributo"**: armazena o nome do atributo;
- **"Tipo de Dado"**: indica que o atributo é um atributo tradicional e, então, não é tratado pelo otimizador do SIREN;
- **"Operador"**: recebe um dos operadores tradicionais ($=$, $\neq$, $>$ etc.);
- **"Valor"**: pode receber uma constante, um nome de atributo ou um ponteiro para outra *Parse Tree* no caso deste ser o resultado de uma subconsulta (*subselect*).

Os **predicados por similaridade** requerem mais parâmetros do que os predicados tradicionais. O campo:

- **"Tipo de Dado"**: indica que o atributo é complexo e permite os tipos de dados `PARTICULATE`, `STILLIMAGE` e `AUDIO`;
- **"Operador"**: é usado para distinguir o tipo de operação (seleção ou junção) por similaridade indicada no campo "Operação" da *Parse Tree* e pode ser:
  - similaridade por abrangência - R_q;
  - similaridade aos k-vizinhos mais próximos - kNN;
  - similaridade agregada por abrangência - AR_q;
  - similaridade agregada aos k-vizinhos mais próximos - $AkNN_q$;
  - junção por abrangência - $\overset{R_q}{\bowtie}$;
  - junção por vizinhança - $\overset{kNN}{\bowtie}$;
  - junção por proximidade - $\overset{kCN}{\bowtie}$;
  - e suas variações;
- **"Valor"**: depende da operação e do tipo de dado para ser preenchido. Se a operação é:
  - seleção por similaridade e o tipo de dado é:
    - `PARTICULATE`: valor recebe um conjunto não vazio de elementos de consulta, em que cada elemento pode ser um conjunto de pares "`atributo/valor`" ou um ponteiro para outra *Parse Tree*, no caso de subconsultas;

        * `STILLIMAGE` ou `AUDIO`: valor pode receber um conjunto não vazio de elementos indicado por uma seqüência de nome de arquivos para o elemento de consulta, ou seja, o caminho para o elemento de consulta, ou um ponteiro para a *Parse Tree* da subconsulta que retorna o elemento de consulta;
    - junção por similaridade: valor recebe um atributo complexo;

- **"k/ξ"**: recebe o valor de k ou ξ, dependendo do predicado por similaridade;
- **"Função de Distância"**: especifica a função de distância definida para cada domínio complexo;
- **"Lista de Empate"** (*tie list*): indica se, nas consultas por vizinhança ou proximidade, serão aceitos ou não os elementos empatados na k-ésima posição;
- **"Operador de Agregação"**: usado somente em operações de agregação, recebe os valores `SUM`, `MAX` e `ALL`, indicando a estratégia usada para processar a similaridade agregada.

Após as estruturas *Parse Tree* e ACT serem criadas, elas são usadas como entrada no módulo de **Reescrita de Planos** do otimizador de consulta. O funcionamento do otimizador de consultas do SIREN é apresentada no Algoritmo 5.1.

Algoritmo 5.1 Funcionamento do otimizador de consultas do SIREN.

Entrada: Plano de consulta inicial.
Saída: Plano de consulta ótimo.

1: O plano de consulta inicial cria as estruturas *Parse Tree* and ACT;
2: As estruturas *Parse Tree* (*Parse1*) e a ACT são usadas como entrada no módulo de **Reescrita de Planos**;
3: *Parse1* é duplicada gerando *Parse1* e *Parse2*;
4: *Parse1* é enviada ao módulo **Avaliador de Planos**, no qual seu custo será estimado - custo(*Parse1*);
5: $i \leftarrow 1$;
6: **for** $i \leftarrow 1, x$ **do**
7: *Parse2* é reescrita usando regras algébricas;
8: *Parse2* é enviada ao Avaliador de Planos e seu custo também é estimado - custo(*Parse2*);
9: **if** custo(*Parse1*) > custo(*Parse2*) **then**
10: *Parse2* troca com *Parse1*;
11: **end if**
12: *Parse1* é mantida no Avaliador de Planos;
13: *Parse2* é reenviada ao módulo de Reescrita de Planos;
14: $i \leftarrow i + 1$;
15: **end for**
16: **return** *Parse1*

Devido ao grande número de planos equivalentes que podem ser gerados para uma consulta, o número de iterações deste algoritmo é definido pelo parâmetro x. Os passos 6 a 15 do Algoritmo 5.1 são repetidos x vezes, gerando um plano alternativo em cada iteração

do algoritmo. Os planos alternativos são gerados utilizando a álgebra por similaridade, apresentada no Capítulo 4.

O plano que apresenta o menor custo estimado dentre todos os planos gerados é chamado de plano de consulta ótimo. Este é selecionado para ser executado, usando o SGBD relacional para executar as operações sobre os dados tradicionais. O SGBD relacional responde ao comando reescrito e esta resposta é enviada ao programa de aplicação.

5.3 Exemplo

Nessa seção é apresentado um exemplo de execução de uma consulta por similaridade realizada por meio da utilização do SIREN. A descrição dos conjuntos de dados `CidadeBR` e `Populacao`, utilizados no exemplo a seguir, é apresentado na Capítulo 2.

Um exemplo de consulta por similaridade utilizando o esquema relacional apresentado na Figura 2.2 é:

Q2: "*Selecione o nome, o estado e a população das 10 cidades brasileiras mais próximas da cidade de "São Carlos-SP", cuja latitude é igual a* -22.02 *e a longitude é igual a* 47.89*, que distam desta até* 0.3 *unidades de distância, considerando a distância Euclidiana* L_2*, e que suas populações totais não excedam 1000 habitantes.*"

Na álgebra por similaridade, esta consulta é representada por:

$$\begin{aligned}&(\hat{\sigma}_{(\texttt{Coordenada}\ Range(L_2,0.3)\ (-22.02,47.89))}\\&\hat{\sigma}_{(\texttt{Coordenada}\ kNN(L_2,10)\ (-22.02,47.89))}\,\texttt{CidadeBR}\\&\ \bowtie\ \sigma_{(poptotal\leq 1000)}\,\texttt{Populacao},\end{aligned} \tag{5.1}$$

e, em SQL estendido, ela é expressa como:

```
SELECT cidade, estado, poptotal
  FROM CidadeBR, Populacao
 WHERE CidadeBR.cidade = Populacao.cidade
   AND CidadeBR.estado = Populacao.estado
   AND coordenada NEAR (-22.02 as latitude,
                        47.89 as longitude) RANGE 0.3
   AND coordenada NEAR (-22.02 as latitude,
                        47.89 as longitude) STOP AFTER 10
    AND poptotal <= 1000.
```

O comando `NEAR` é um operador por similaridade que utiliza a métrica definida pelo atributo; os comandos `RANGE` e `STOP AFTER` indicam os tipos de consulta por similaridade que devem ser aplicadas, isto é, similaridade por abrangência (R_q) e similaridade aos k-vizinhos mais próximos (kNN), respectivamente.

O programa de aplicação envia a consulta SQL estendida para o SGBD relacional. SIREN intercepta a consulta e verifica que ela é composta por predicados tradicionais e por similaridade. O compilador do SIREN realiza a análise léxica, sintática e semântica desta consulta, certifica-se de que ela é válida e transforma-a no plano de consulta inicial, mostrado na Figura 5.4, que é enviado como entrada para o otimizador de consultas.

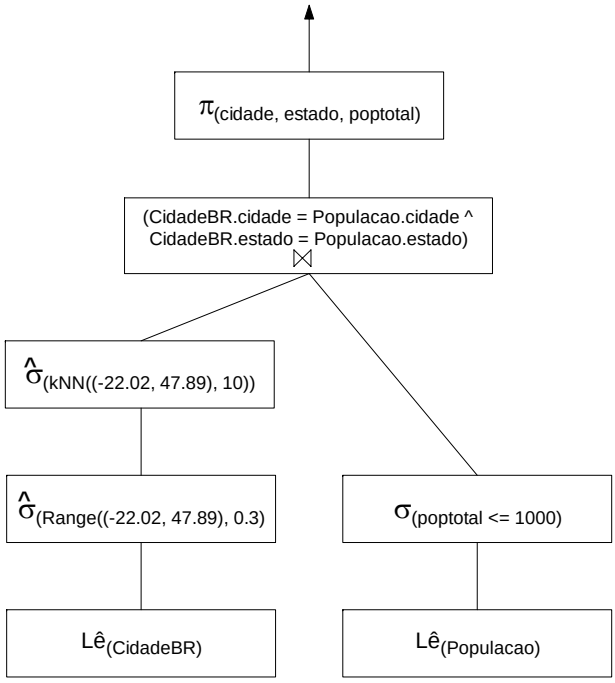

Figura 5.4: Plano de consulta inicial da consulta **Q2**.

No otimizador do SIREN, o Algoritmo 5.1 é executado. O plano de consulta inicial é examinado, as estruturas *Parse Tree* e ACT são organizadas, conforme mostrado na Figura 5.5, e são usadas como entrada no módulo de Reescrita de Plano.

O plano de consulta inicial (Figura 5.5a) submetido para o módulo de Reescrita de Plano é duplicado e um deles é enviado para o módulo Avaliador de Planos para ter seu custo estimado. O outro é reescrito usando as propriedades algébricas da álgebra por similaridade e é enviado para o módulo Avaliador de Planos para que seu custo também seja estimado. A Figura 5.6 mostra o plano de consulta inicial e um dos planos alternativos que foi gerado pelo módulo Reescrita de Planos. A Figura 5.6b mostra o plano de consulta alternativo obtido usando o teorema "Conjunção de predicados *k-nearest* e *range* com o mesmo centro", proposto por Traina-Jr. et al. (2006b), o qual demonstra que a intersecção dos resultados dos predicados *k-nearest* e *range* sobre a mesma relação é equivalente à

(a)

1	Lê	CidadeBR			
2	Lê		Populacao		
3	$\hat{\sigma}$	1		c1	
4	$\hat{\sigma}$	3		c2	
5	σ		2	c3	
6	$\bowtie$	4	5	c4 ^ c5	
7	π	6			

(b)

c1	coordenada	P	RANGE	(-22.02 as latitude, 47.89 as longitude)	0.3	L2	N	
c2	coordenada	P	STOP AFTER	(-22.02 as latitude, 47.89 as longitude)	10	L2	N	
c3	poptotal	T	<=	1000				
c4	CidadeBR.cidade	T	=	Populacao.cidade				
c5	CidadeBR.estado	T	=	Populacao.estado				

Figura 5.5: Estruturas de operação por similaridade para a expressão algébrica 5.1. (a) Plano de consulta inicial. (b) Tabela de Condições e Atributos - ACT.

conjunção dos predicados *k-nearest* e *range*. Portanto, o algoritmo $kAndRange(\theta, t_q, k, \xi)$ pode ser usado para executar as condições $c1$ e $c2$ juntas.

(a)

1	Lê	CidadeBR			
2	Lê		Populacao		
3	$\hat{\sigma}$	1		c1	
4	$\hat{\sigma}$	3		c2	
5	σ		2	c3	
6	$\bowtie$	4	5	c4 ^ c5	
7	π	6			

(b)

1	Lê	CidadeBR			
2	Lê		Populacao		
3	$\hat{\sigma}$	1		c1 ^ c2	
4	σ		2	c3	
5	$\bowtie$	3	4	c4 ^ c5	
6	π	5			

Figura 5.6: (a)Plano de consulta inicial. (b) Plano de consulta alternativo gerado pelo módulo de Reescrita de Plano.

Os custos do plano de consulta inicial e do plano alternativo são comparados e o plano com o menor custo computacional é mantido no módulo Avaliador de Planos enquanto o outro é retornado ao módulo de Reescrita de Planos. Neste trabalho foi realizada uma estimativa do custo de CPU das consultas pois o módulo Avaliador de Planos faz parte das propostas de trabalhos futuros apresentadas no Capítulo 6. Este ciclo continua até que o número de planos estipulado como parâmetro sejam formados e comparados. O plano com o menor custo computacional, dentre todos os produzidos, é selecionado e usado como saída do otimizador, para ser executado pelo SGBD relacional. Por ser um

exemplo ilustrativo do funcionamento do otimizador do SIREN, o plano de consulta inicial foi duplicado e sua cópia foi reescrita somente uma vez.

Neste exemplo, o plano mostrado na Figura 5.6b é considerado como o plano de consulta ótimo. SIREN reescreve a consulta SQL baseada no plano de consulta escolhido e executa esta consulta usando o SGBD relacional para executar as operações sobre os dados tradicionais. O SGBD relacional responde ao comando reescrito e esta resposta é enviada ao programa de aplicação.

5.4 Considerações finais

O SIREN foi, inicialmente, proposto para compilar e executar consultas por similaridade sobre dados complexos armazenados no SGBD relacional. Em sua versão original, o SIREN não otimiza estas consultas, pois não possui em sua estrutura um otimizador de consultas.

Neste capítulo, foi apresentado o otimizador de consultas desenvolvido para o SIREN, capaz de interpretar, traduzir, avaliar e executar as consultas por similaridade sobre os dados complexos no SGBD relacional. Este otimizador é baseado em reescrita de consulta e utiliza as estruturas de operações por similaridade (*Parse Tree* e Tabela de Condições e Atributos), a álgebra por similaridade e suas propriedades algébricas para gerar planos alternativos.

Como o otimizador precisa das regras de reescrita de consultas baseadas em propriedades algébricas e não existe ainda um estudo sistemático que elenque essas regras, isso foi feito neste trabalho. O otimizador de consultas utiliza a álgebra por similaridade e as propriedades algébricas pertencentes a ela, que foram apresentadas no Capítulo 4, para otimizar as consultas por similaridade no otimizador de consultas do SIREN.

Capítulo

6

Conclusão

6.1 Considerações finais

Atualmente, o armazenamento de dados multimídia, espaciais, séries temporais, seqüências genéticas, entre outros, em SGBDs relacionais tem atraído muita atenção de pesquisadores e, com isto, é amplamente estudado. Mas para que um SGBD relacional possa armazenar e recuperar estes tipos de dados complexos, o processador de consultas do SGBD relacional deve prover suporte às consultas por similaridade, visto que as consultas por meio das relações de igualdade e da ROT não são aplicáveis a estes dados. Isto implica em: permitir que as consultas por similaridade sejam expressas em álgebra relacional, possibilitando a sua composição em expressões mais complexas e sua tradução em expressões equivalentes; avaliar as possibilidades de otimização considerando as regras algébricas e estimativas de custo e de seletividade; e executar a expressão com o menor custo computacional.

Para que as consultas por similaridade possam ser reescritas, esta dissertação propõe um modelo de dados e a álgebra por similaridade. A álgebra por similaridade e as regras algébricas capazes de transformar as expressões algébricas em expressões equivalentes, apresentadas nesta dissertação, são para as consultas por similaridade unárias, isto é, para a seleção por similaridade por abrangência e por vizinhança e suas variantes.

As regras algébricas apresentadas para simplificar uma expressão por similaridade complexa permitem que o Gerador de Planos possa gerar múltiplas representações para uma mesma consulta. O otimizador, tendo as várias representações equivalentes de uma consulta, pode estimar o custo de cada uma delas, escolhendo a de menor custo computacional.

Dentro deste contexto, esta dissertação também complementa a implementação do SIREN, que é uma camada intermediária entre a aplicação e o SGBD relacional, incorporando a ele um otimizador de consultas capaz de otimizar as consultas por similaridade. No otimizador do SIREN foram definidas as estruturas *Parse Tree* e Tabela de Condições e Atributos, o módulo Reescrita de Planos e o módulo Avaliador de Planos. A álgebra por similaridade, proposta nesta dissertação, foi incorporada ao módulo de Reescrita de Planos do SIREN.

Esta dissertação contribui para a incorporação de consultas por similaridade complexas em SGBDs relacionais, pois incorpora o otimizador de consultas, o modelo de dados e a álgebra por similaridade no SIREN. Assim, a realização de otimização de consultas por similaridade nos SGBDs relacionais torna-se possível.

6.2 Principais contribuições deste trabalho

As principais contribuições desta dissertação são:

- definição de regras algébricas para transformação de expressões compostas pelos operadores unários das consultas por similaridade, seleção por similaridade por abrangência e seleção por similaridade aos vizinhos mais próximos, formando assim o Espaço Algébrico do otimizador de consultas do SIREN. Para as consultas por similaridade por abrangência são válidas todas as regras algébricas válidas para as consultas tradicionais. Já para as consultas por similaridade são válidas somente cinco regras algébricas, sendo que quatro delas são especializações das regras algébricas para consultas tradicionais envolvendo operadores unários.

- definição do modelo de dados e da **Álgebra por Similaridade**, baseada no princípio de *ranking*, para consultas por similaridade, usando as funções de distância como critério de *rank*.

- definição do operador unário por vizinhança para que seus resultados intermediários possam ser usados como entrada de outros operadores.

- incorporação do otimizador de consultas ao SIREN, que o tornou capaz de processar as consultas por similaridade de maneira análoga ao processador de consultas do SGBD relacional.

- definição das estruturas *Parse Tree* e Tabela de Condições e Atributos para o tratamento das consultas por similaridade.

- verificação de que os algoritmos de otimização de consultas para seleções tradicionais podem ser usados para otimizar consultas por similaridade por abrangência mas não por vizinhança.

- publicação do artigo intitulado: "*An efficient framework for similarity query optimization*" na conferência "*ACM International Symposium on Advances in Geographic Information Systems*", que aconteceu em *Seattle*, Estados Unidos, em novembro de 2007.

6.3 Proposta para trabalhos futuros

O trabalho realizado neste mestrado permite que novas frentes de pesquisa possam ser desenvolvidas; entre as quais estão:

- extensão da **Álgebra por Similaridade** proposta para tratar também dos operadores binários das consultas por similaridade, definindo as regras algébricas e os algoritmos de otimização de consultas válidos para estes operadores.
- extensão da Álgebra por Similaridade proposta para tratar das operações de agregação, definindo as regras algébricas e os algoritmos de otimização válidos para estas operações.
- incorporação da Álgebra por Similaridade, com os operadores binários e as operações de agregação, ao módulo Reescrita de Planos do otimizador de consultas do SIREN.
- implementação do módulo Avaliador de Planos do otimizador de consultas do SIREN, desenvolvendo estimativas de custo e de seletividade para as consultas por similaridade.
- desenvolvimento de algoritmos específicos para tratar a otimização das consultas por vizinhança.

Outras frentes de pesquisa para as quais esta dissertação também contribui são as seguintes:

- desenvolvimento de novos operadores de consultas por similaridade que possam ser integrados nos já estudados e que auxiliem a responder a consultas que incluam similaridade sobre dados complexos. Exemplos desses operadores são: consulta aos k vizinhos mais próximos reversos; consultas levem em conta agrupamentos por similaridade.
- desenvolvimento de operadores de consultas por similaridade específicos para determinados tipos de dados complexos como, por exemplo, que levem em conta a existência de áudio e texto (criptografia) conjugado com imagens em vídeo.
- incorporação da variável tempo em dados multimídia.
- integrar consultas por similaridade com consulta topológicas e cardinais em dados espaciais.

Referências Bibliográficas

Adali, S.; Bonatti, P.; Sapino, M.; Subrahmanian, V. A multi-similarity algebra. In: Haas, L. M.; Tiwary, A., eds. *SIGMOD'98: Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data*, Seattle, WA, USA: ACM Press, 1998, p. 402–413.

Adali, S.; Bufi, C.; Sapino, M.-L. Ranked relations: query languages and query processing methods for multimedia. *Multimedia Tools and applications Journal (MTAJ)*, v. 24, n. 3, p. 197–214, 2004.

Arantes, A. S. *Consultas por similaridade complexas em gerenciadores relacionais*. Tese de Doutoramento, Universidade de São Paulo (USP), 2005.

Arantes, A. S.; Vieira, M. R.; Traina-Jr., C.; Traina, A. J. M. Operadores de seleção por similaridade para sistemas de gerenciamento de bases de dados relacionais. In: Laender, A. H. F., ed. *SBBD'03: Anais do 18º Simpósio Brasileiro de Bancos de Dados*, Manaus, Amazonas, Brasil: UFAM, 2003, p. 341–355.

Arantes, A. S.; Vieira, M. R.; Traina-Jr., C.; Traina, A. J. M. Efficient algorithms to execute complex similarity queries in RDBMS. *Journal of the Brazilian Computer Society (JBCS)*, v. 9, n. 3, p. 5–24, 2004.

Atnafu, S.; Brunie, L.; Kosch, H. Similarity-based operators and query optimization for multimedia database systems. In: Adiba, M. E.; Collet, C.; Desai, B. C., eds. *IDEAS'01: Proceedings of the International Database Engineering and Applications Symposium*, Grenoble, France: IEEE Computer Society, 2001, p. 346–355.

Baeza-Yates, R. A.; Cunto, W.; Manber, U.; Wu, S. Proximity matching using fixed-queries trees. In: Crochemore, M.; Gusfield, D., eds. *CPM '94: Proceedings of the 5th Annual Symposium on Combinatorial Pattern Matching*, Asilomar, CA: Springer-Verlag, 1994, p. 198–212.

Barioni, M. C. N. *Operações de consulta por similaridade em grandes bases de dados complexos*. Tese de Doutoramento, Universidade de São Paulo (USP), 2006.

Barioni, M. C. N.; Razente, H. L.; Traina, A. J. M.; Traina-Jr., C. SIREN: A similarity retrieval engine for complex data. In: Dayal, U.; Whang, K.-Y.; Lomet, D. B.; Alonso, G.; Lohman, G. M.; Kersten, M. L.; Cha, S. K.; Kim, Y.-K., eds. *VLDB'06: Proceedings of the 32nd International Conference on Very Large Data Bases*, Seoul, Korea: ACM, 2006, p. 1155–1158.

Barioni, M. C. N.; Razente, H. L.; Traina-Jr., C.; Traina, A. J. M. Querying complex objects by similarity in SQL. In: Heuser, C. A., ed. *SBBD'05: Anais do 20º Simpósio Brasileiro de Bancos de Dados*, Uberlândia, MG, Brazil: UFU, 2005, p. 130–144.

Belohlávek, R.; Opichal, S.; Vychodil., V. Relational algebra for ranked tables with similarity: properties and implementation. In: Berthold, M. R.; Shawe-Taylor, J.; Lavrac, N., eds. *IDA'07: Proceedings of the 7th International Symposium on Intelligent Data Analysis*, Springer Verlag, 2007, p. 140–151 (*Lecture Notes in Computer Science*, v.4723).

Benetis, R.; Jensen, C. S.; Karciauskas, G.; Saltenis, S. Nearest neighbor and reverse nearest neighbor queries for moving objects. In: Nascimento, M. A.; Özsu, M. T.; Zaïane, O. R., eds. *IDEAS'02: Proceedings of the International Database Engineering and Applications Symposium*, Edmonton, Canada: IEEE Computer Society, 2002, p. 44–53.

Berchtold, S.; Böhm, C.; Braunmüller, B.; Keim, D. A.; Kriegel, H.-P. Fast parallel similarity search in multimedia databases. In: Peckham, J., ed. *SIGMOD'97: Proceedings of the ACM SIGMOD International Conference on Management of Data*, Tucson, Arizona, USA: ACM Press, 1997, p. 1–12.

Böhm, C.; Berchtold, S.; Keim, D. A. Searching in high-dimensional spaces: Index structures for improving the performance of multimedia databases. *ACM Computing Surveys (CSUR)*, v. 33, n. 3, p. 322–373, 2001.

Böhm, C.; Krebs, F. High performance data mining using the nearest neighbor join. In: *ICDM '02: Proceedings of the 2nd IEEE International Conference on Data Mining*, Maebashi City, Japan: IEEE Computer Society, 2002, p. 43–50.

Böhm, C.; Kriegel, H.-P. A cost model and index architecture for the similarity join. In: *ICDE'01: Proceedings of the 17th International Conference on Data Engineering*, Heidelberg, Germany: IEEE Computer Society, 2001, p. 411–420.

Bozkaya, T.; Ozsoyoglu, M. Distance-based indexing for high-dimensional metric spaces. In: Peckman, J. M.; Ram, S.; Franklin, M., eds. *SIGMOD '97:*

Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data, Tucson, Arizona, United States: ACM Press, 1997, p. 357–368.

Bozkaya, T.; Ozsoyoglu, M. Indexing large metric spaces for similarity search queries. *ACM Transactions on Database Systems (TODS)*, v. 24, n. 3, p. 361–404, 1999.

Brin, S. Near neighbor search in large metric spaces. In: Dayal, U.; Gray, P. M. D.; Nishio, S., eds. *VLDB '95: Proceedings of the 21th International Conference on Very Large Data Bases*, Zurich, Switzerland.: Morgan Kaufmann Publishers Inc., 1995, p. 574–584.

Brinkhoff, T.; Kriegel, H.-P.; Seeger, B. Efficient processing of spacial joins using R-tree. In: *SIGMOD'93: Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, ACM Press, 1993, p. 237–246.

Burkhard, W. A.; Keller, R. M. Some approaches to best-match file searching. *Communications of the ACM (CACM)*, v. 16, n. 4, p. 230–236, 1973.

Chaudhuri, S.; Gravano, L. Optimizing queries over multimedia repositories. In: Widom, J., ed. *SIGMOD'96: Proceedings of the 1996 ACM SIGMOD international conference on Management of data*, Montreal, Quebec, Canada: ACM Press, 1996, p. 91–102.

Cheung, K. L.; Fu, A. W.-c. Enhanced nearest neighbour search on the R-tree. *ACM SIGMOD Records*, v. 27, n. 3, p. 16–21, 1998.

Chávez, E.; Navarro, G.; Baeza-Yates, R. A.; Marroquín, J. L. Searching in metric spaces. *ACM Computing Surveys (CSUR)*, v. 33, n. 3, p. 273–321, 2001.

Ciaccia, P.; Montesi, D.; Penzo, W.; Trombetta, A. Imprecision and user preferences in multimedia queries: a generic algebriac approach. In: Schewe, K.-D.; Thalheim, B., eds. *FoIKS'00: Proceedings of 1st International Sumposium in Foundations of Information and Knowledge Systems*, Burg, Germany: Springer Verlag, 2000, p. 50–71 (*Lecture Notes in Computer Science*, v.1762).

Ciaccia, P.; Patella, M.; Zezula, P. M-tree: An efficient access method for similarity search in metric spaces. In: Jarke, M.; Carey, M. J.; Dittrich, K. R.; Lochovsky, F. H.; Loucopoulos, P.; Jeusfeld, M. A., eds. *VLDB'97: Proceedings of the 23rd International Conference on Very Large Data Bases*, Athens, Greece: Morgan Kaufmann Publishers Inc., 1997, p. 426–435.

Ciaccia, P.; Patella, M.; Zezula, P. A cost model for similarity queries in metric spaces. In: *PODS'98: Proceedings of the 17th ACM SIGACT-SIGMOD-SIGART*

Symposium on Principles of Database Systems, Seattle, Washington, United States: ACM Press, 1998, p. 59–68.

Dohnal, V.; Gennaro, C.; Savino, P.; Zezula, P. Similarity join in metric spaces. In: Sebastiani, F., ed. *ECIR'03: Proceedings of the 25th European Conference on Information Retrieval Research*, Pisa, Italy: Springer Verlag, 2003, p. 452–467 (*Lecture Notes in Computer Science*, v.2633).

Elmasri, R.; Navathe, S. B. *Sistemas de banco de dados.* 4° ed. São Paulo: Addison Wesley, 2005.

Fagin, R. Combining fuzzy information from multiple systems. In: *PODS'96: Proceedings of the 15th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, Montreal, Quebec, Canada: ACM Press, 1996, p. 216–226.

Faloutsos, C. *Searching multimedia databases by content.* The Kluwer international series on advances in database systems. Boston, MA: Kluwer Academc Publishers, 1996.

Faloutsos, C. Indexing of multimedia data. In: *Multimedia Databases in Perspective*, Springer, p. 219–245, 1997.

Ferreira, M. R. P.; Traina-Jr., C.; Traina, A. J. M. An efficient framework for similarity query optimization. In: Samet, H.; Schneider, M.; Shahabi, C., eds. *ACM GIS'07: Proceedings of the 15th ACM International Symposium on Advances in Geographic Information Systems, November 7-9, 2007*, Seattle, WA, USA: ACM Press, 2007, p. 396–399.

Gaede, V.; Günther, O. Multidimensional access methods. *ACM Computing Surveys (CSUR)*, v. 30, n. 2, p. 170–231, 1998.

Garcia-Molina, H.; Ullman, J. D.; Widom, J. *Database system implementation.* New Jersey: Prentice Hall, 2000.

Hjaltason, G. R.; Samet, H. Distance browsing in spatial databases. *ACM Transactions on Database Systems (TODS)*, v. 24, n. 2, p. 265–318, 1999.

Hjaltason, G. R.; Samet, H. Index-driven similarity search in metric spaces. *ACM Transactions on Database Systems (TODS)*, v. 28, n. 4, p. 517–580, 2003.

IBGE Coordenadas geográficas das cidades brasileiras. 2006. Disponível em `http://www.ibge.gov.br, acessado em 07/08/2006`

Ioannidis, Y. E. Query optimization. *ACM Computing Surveys (CSUR)*, v. 28, n. 1, p. 121–123, 1996.

KORN, F.; SIDIROPOULOS, N.; FALOUTSOS, C.; SIEGEL, E.; PROTOPAPAS, Z. Fast nearest neighbor search in medical image databases. In: VIJAYARAMAN, T. M.; BUCHMANN, A. P.; MOHAN, C.; SARDA, N. L., eds. *VLDB'96: Proceedings of the 22th International Conference on Very Large Data Bases*, Mumbai (Bombay), India: Morgan Kaufmann, 1996, p. 215–226.

KOUDAS, N.; SEVCIK, K. C. High dimensional similarity join: Algorithms and performance evaluation. In: *ICDE'98: Proceedings of the 14th International Conference on Data Engineering, February 23–27, 1998*, Orlando, Florida, USA: IEEE Computer Society, 1998, p. 466–475.

LI, C.; CHANG, K. C.-C.; ILYAS, I. F.; SONG, S. RankSQL: query algebra and optimization for relational top-k queries. In: VIJAYARAMAN, T. M.; BUCHMANN, A. P.; MOHAN, C.; SARDA, N. L., eds. *SIGMOD'05: Proceedings of the ACM International Conference on Management of Data, June 14–16, 2005*, Baltimore, Maryland, USA: ACM Press, 2005, p. 131–142.

LIU, B.; WANG, Z.; YANG, X.; WANG, W.; SHI, B. A bottom-up distance-based index tree for metric space. In: WANG, G.; PETERS, J. F.; SKOWRON, A.; YAO, Y., eds. *Rough Sets and Knowledge Technology*, Chongquing, China: Springer, first International Conference, RSKT - Proceedings, 2006, p. 442–449 (*Lecture Notes in Computer Science*, v.4062).

MELTON, J.; MICHELS, J.-E.; JOSIFOVSKI, V.; KULKARNI, K.; SCHWARZ, P. SQL/MED – a status report. *ACM SIGMOD Record*, v. 31, n. 3, p. 81–89, 2002.

NAVARRO, G. Searching in metric spaces by spatial approximation. *The International Journal on Very Large Data Bases*, v. 11, n. 1, p. 28–46, the VLDB Journal, 2002.

OCSA, A.; CUADROS-VARGAS, E. DBM*-Tree: an efficient metric access method. In: *ACM Southeast Regional Conference*, Winston-Salem, North Carolina: ACM Press, 2007, p. 401–406.

PARK, D.-J.; KIM, H.-J. An enhanced technique for k-nearest neighbor queries with non-spatial selection predicates. *Multimedia Tools and Applications*, v. 19, n. 1, p. 79–103, 2003.

PENZO, W. Rewriting rules to permeate complex similarity and fuzzy queries within a relational database system. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, v. 17, n. 2, p. 255–270, 2005.

POLA, I. R. V.; JR., C. T.; TRAINA, A. J. M. The MM-Tree: A memory-based metric tree without overlap between nodes. In: IOANNIDIS, Y. E.; NOVIKOV, B.;

RACHEV, B., eds. *Advances in Databases and Information Systems*, Varna, Bulgaria: Springer, 2007, p. 157–171 (*Lecture Notes in Computer Science*, v.4690).

RAMAKRISHNAN, R.; GEHRKE, J. *Database management systems*. 3° ed. Boston: McGraw-Hill, 2003.

RAZENTE, H. L.; BARIONI, M. C. N.; TRAINA, A. J. M.; JR., C. T. Constrained aggregate similarity queries in metric spaces. In: DA SILVA, A. S., ed. *Simpósio Brasileiro de Banco de Dados*, João Pessoa, Paraíba, Brasil: SBC, 2007, p. 145–159.

ROUSSOPOULOS, N.; KELLEY, S.; VINCENT, F. Nearest neighbor queries. In: CAREY, M. J.; SCHNEIDER, D. A., eds. *SIGMOD'95: Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data*, San Jose, California: ACM Press, 1995, p. 71–79.

SAMET, H. Depth-first k-nearest neighbor finding using the maxnearestdist estimator. In: *ICIAP'03: Proceedings of the 12th International Conference on Image Analysis and Processing*, Mantova, Italy: IEEE Computer Society, 2003, p. 486–491.

SANTOS-FILHO, R. F.; TRAINA, A. J. M.; TRAINA-JR., C.; FALOUTSOS, C. Similarity search without tears: the omni-family of all-purpose access methods. In: *ICDE'01: Proceedings of the 17th International Conference on Data Engineering*, Heidelberg, Germany: IEEE Computer Society, 2001, p. 623–630.

SEIDL, T.; KRIEGEL, H.-P. Optimal multi-step k-nearest neighbor search. In: TIWARY, A.; FRANKLIN, M., eds. *SIGMOD '98: Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data*, Seattle, Washington, United States: ACM Press, 1998, p. 154–165.

SERAPHIM, E. *Operadores binários para consultas de similaridade em banco de dados multimídia.* Tese de Doutoramento, Universidade de São Paulo (USP), 2005.

SHIM, K.; SRIKANT, R.; AGRAWAL, R. High-dimensional similarity joins. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, v. 14, n. 1, p. 156–171, 2002.

TRAINA, A. J. M.; TRAINA-JR, C. Similarity search in multimedia databases. In: FURHT, B.; MARQUES, O., eds. *Handbook of Video Databases: design and applications*, 1st ed, CRC Press, p. 711–738, 2003.

TRAINA, A. J. M.; TRAINA-JR., C.; BUENO, J. M.; AZEVEDO-MARQUES, P. The metric histogram: A new and efficient approach for content-based image retrieval. In: ZHOU, X.; PU, P., eds. *IFIP'02: Proceedings of the IFIP TC2/WG2.6 6th*

Working Conference on Visual Database Systems: Visual and Multimedia Information Management, Brisbane, Australia: Kluwer Academic Publishers, 2002, p. 297–311.

Traina-Jr., C.; Santos-Filho, R. F.; Traina, A. J. M.; Vieira, M. R.; Faloutsos, C. The omni-family of all-purpose access methods: a simple and effective way to make similarity search more efficient. *The VLDB Journal*, publicado online (disponível em http://dx.doi.org/10.1007/s00778-005-0178-0), 2006a.

Traina-Jr., C.; Traina, A. J. M.; Faloutsos, C.; Seeger, B. Fast indexing and visualization of metric data sets using slim-trees. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, v. 14, n. 2, p. 244–260, 2002.

Traina-Jr., C.; Traina, A. J. M.; Seeger, B.; Faloutsos, C. Slim-trees: High performance metric trees minimizing overlap between nodes. In: Zaniolo, C.; Lockemann, P. C.; Scholl, M. H.; Grust, T., eds. *EDBT'00: Proceedings of the 7th International Conference on Extending Database Technology*, Konstanz, Germany: Springer Verlag, 2000, p. 51–65 (*Lecture Notes in Computer Science*, v.1777).

Traina-Jr., C.; Traina, A. J. M.; Vieira, M. R.; Arantes, A. S.; Faloutsos, C. Efficient processing of complex similarity queries in RDBMS through query rewriting. In: *CIKM'06: Proceedings of the 15th ACM international conference on Information and knowledge management*, Arlington, Virginia, USA: ACM Press, 2006b, p. 4–13.

Uhlmann, J. K. Satisfying general proximity/similarity queries with metric trees. *Information Processing Letters*, v. 40, n. 4, p. 175–179, 1991.

Uribe, R.; Navarro, G.; Barrientos, R. J.; Marín, M. An index data structure for searching in metric space databases. In: Alexandrov, V. N.; van Albada, G. D.; Sloot, P. M. A.; Dongarra, J., eds. *International Conference on Computational Science (1)*, UK: Springer, 2006, p. 611–617 (*Lecture Notes in Computer Science*, v.3991).

Vieira, M. R.; Traina-Jr., C.; Chino, F. J. T.; Traina, A. J. M. DBM-Tree: A dynamic metric access method sensitive to local density data. In: Lifschitz, S., ed. *SBBD'04: XIX Simpósio Brasileiro de Bancos de Dados*, Brasília, Distrito Federal, Brasil: UnB, 2004, p. 163–177.

Vieira, M. R.; Traina-Jr., C.; Traina, A. J. M.; Arantes, A.; Faloutsos, C. Boosting k-nearest neighbor queries estimating suitable query radii. In: *SSDBM'2007: Proceedings of 19th International Conference on Scientific and Statistical Database Management, July 9–11, 2007*, Banff, Canada: IEEE Computer Society, 2007, p. 1–10.

WANG, J. T.-L.; SHASHA, D. Query processing for distance metrics. In: MCLEOD, D.; SACKS-DAVIS, R.; SCHEK, H.-J., eds. *VLDB'90: Proceedings of the 16th International Conference on Very Large Data Bases*, Brisbane, Queensland, Australia: Morgan Kaufmann, 1990, p. 602–613.

WEBER, R.; SCHEK, H.-J.; BLOTT, S. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In: NASCIMENTO, M. A.; ÖZSU, M. T.; ZAÏANE, O. R., eds. *VLDB '98: Proceedings of the 24th International Conference on Very Large Data Bases*, New York, USA: Morgan Kaufmann, 1998, p. 194–205.

YANG, C.; LIN, K.-I. An index structure for improving closest pairs and related join queries in spatial databases. In: NASCIMENTO, M. A.; ÖZSU, M. T.; ZAÏANE, O. R., eds. *IDEAS02: Proceedings of the International Database Engineering and Applications Symposium*, Edmonton, Canada: IEEE Computer Society, 2002, p. 140–149.

YIANILOS, P. N. Data structures and algorithms for nearest neighbor search in general metric spaces. In: *SODA'93: Proceedings of the 4th Annual ACM-SIAM Symposium on Discrete Algorithms*, Austin, Texas, United States: Society for Industrial and Applied Mathematics, 1993, p. 311–321.

YU, C. T.; MENG, W. *Principles of database query processing for advanced applications.* 1° ed. San Francisco, California, USA: Morgan Kaufman, 2002.

Apêndice

A

SIREN

As definições e propriedades apresentadas nesta seção são baseadas nos trabalhos de Barioni et al. (2006) e Barioni (2006).

O *SImilarity Retrieval ENgine* (SIREN) é um serviço implementado entre o SGBD relacional e os programas de aplicação, que permite a representação e execução de consultas por similaridade usando a extensão da linguagem SQL proposta em Barioni et al. (2005), conforme mostrado na Figura A.1. O SIREN intercepta todo comando SQL enviado pela aplicação para verificar se existe ou não alguma construção sintática relacionada às operações por similaridade.

Caso o comando não possua nenhuma cláusula que envolva construções por similaridade nem nenhuma referência a elementos complexos, o SIREN envia o comando para o SGBD relacional, que processa a consulta e retorna a resposta para a aplicação. Assim, quando a aplicação submete somente comandos em SQL padrão, o SIREN é transparente.

Por outro lado, caso o comando possua alguma cláusula que envolva construções de similaridade ou referência a elementos complexos, o comando é analisado pelo compilador do SIREN, que executa as análises léxica, sintática e semântica, reescreve o comando original e o envia ao SGBD relacional. As operações por similaridade são executadas internamente pelo SIREN, que usa o SGBD relacional para executar as operações sobre os dados tradicionais. O SGBD relacional responde ao comando reescrito e esta resposta é enviada ao programa de aplicação.

A arquitetura do SIREN é constituída por três componentes principais, conforme mostrado na Figura A.1:

- o **compilador** responsável pela interpretação da especificação de uma extensão da sintaxe SQL para a definição e a manipulação de elementos complexos, considerando aspectos relacionados à realização de consultas por similaridade.

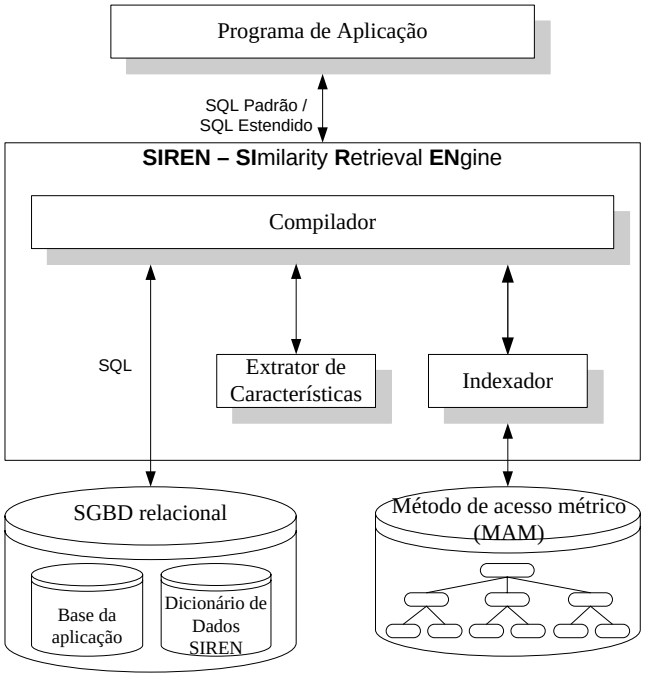

Figura A.1: Arquitetura do SIREN Barioni (2006).

- o **extrator de características** responsável pela extração de características que são utilizadas para a representação e a indexação dos elementos complexos.
- o **indexador** responsável pela utilização de estruturas de indexação apropriadas, os MAM, que respondam às consultas por similaridade.

Barioni et al. (2006) definiram dois domínios de dados complexos que permitem representar as consultas por similaridade na extensão do SQL padrão: o domínio `PARTICULATE`, que considera aqueles elementos que são armazenados como um conjunto de atributos tradicionais; e o domínio `MONOLITHIC`, que considera aqueles elementos armazenados como um único elemento binário BLOB (*Binary Large Objects*).

Novos tipos de dados foram definidos para que a similaridade pudesse ser associada. Para o domínio `PARTICULATE`, o tipo de dado `PARTICULATE` é usado para indicar que o elemento complexo é composto por uma coleção de atributos tradicionais, cujos valores são utilizados para calcular a distância entre cada par de elemento complexo. No domínio `MONOLITHIC`, os tipos de dados `STILLIMAGE` e `AUDIO` são usados para indicar que o elemento complexo não pode ser dividido em um conjunto de atributos da relação. Para comparar elementos destes tipos de dados, é necessário aplicar algoritmos de extração de características sobre eles. Estas características são armazenadas junto com o elemento complexo sem precisar que o usuário se preocupe com o seu armazenamento. Os tipos de dados `STILLIMAGE` e `AUDIO` são especializações do domínio `MONOLITHIC` para o armazenamento de imagens e áudio, respectivamente. Estes tipos de dados são utilizados pelo SIREN para representar elementos complexos nas tabelas do SGBD relacional.

Para responder as consultas que envolvam predicados por similaridade, o SIREN conta com onze operadores por similaridade:

- dois operadores realizam seleção por similaridade: consulta por similaridade por abrangência - R_q e consulta por similaridade aos vizinhos mais próximos - kNN.
- seis operadores realizam seleção agregada por similaridade: consulta por similaridade agregada por abrangência - AR_q e consulta por similaridade agregada aos vizinhos mais próximos - $AkNN_q$, considerando os padrões de agregação por similaridade `SUM`, `MAX` e `ALL`.
- três operadores realizam junção por similaridade: junção por abrangência - $\overset{R_q}{\bowtie}$, junção por vizinhança - $\overset{kNN}{\bowtie}$ e junção por proximidade - $\overset{kCN}{\bowtie}$.

As consultas por similaridade agregada (*Aggregate Similarity Queries*) (Razente et al., 2007) são generalizações das consultas por similaridade que utilizam um único centro de consulta, de modo que a consulta possa ter mais que um centro.

O método de acesso métrico (MAM) utilizado pelo SIREN, para indexar os atributos complexos é o MAM *Slim-tree* (Traina-Jr. et al., 2002). Nessa implementação da *Slim-tree* já existem procedimentos para a execução de R_q, kNN e dos seis operadores de seleção agregada por similaridade (Razente et al., 2007). Entretanto, ainda não há procedimentos publicados para a execução dos demais operadores por similaridade, nem na *Slim-tree* nem em qualquer outro MAM. Dessa maneira, se um atributo complexo é associado a um índice, o SIREN executa a seleção por similaridade e a seleção agregada por similaridade utilizando a Slim-tree. Porém, as junções por similaridade são realizadas por meio de busca seqüencial.