UNIVERSIDADE DE SÃO PAULO

Instituto de Ciências Matemáticas e de Computação

Desenvolvimento de operadores de agrupamento por similaridade em SGBD relacionais

Natan de Almeida Laverde

Dissertação de Mestrado do Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional (PPG-C²MC)

SERVIÇO DE PÓS-GRADUAÇÃO DO ICMC-USP

Data de Depósito:

Assinatura: _____

Natan de Almeida Laverde

Desenvolvimento de operadores de agrupamento por similaridade em SGBD relacionais

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

Área de Concentração: Ciências de Computação e Matemática Computacional

Orientador: Prof. Dr. Caetano Traina Júnior

USP – São Carlos
Julho de 2018

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

L399d Laverde, Natan de Almeida
Desenvolvimento de operadores de agrupamento por
similaridade em SGBD relacionais / Natan de Almeida
Laverde; orientador Caetano Traina Júnior. -- São
Carlos, 2018.
102 p.

Dissertação (Mestrado - Programa de Pós-Graduação
em Ciências de Computação e Matemática
Computacional) -- Instituto de Ciências Matemáticas
e de Computação, Universidade de São Paulo, 2018.

1. Agrupamento por Similaridade.. 2. Agregação
por Similaridade.. 3. Similaridade.. 4. SGBDR.. 5.
Banco de Dados.. I. Traina Júnior, Caetano, orient.
II. Título.

Natan de Almeida Laverde

Development of similarity group operators in Relational DBMS

Master dissertation submitted to the Institute of
Mathematics and Computer Sciences – ICMC-USP,
in partial fulfillment of the requirements for the
degree of the Master Program in Computer Science
and Computational Mathematics. *FINAL VERSION*

Concentration Area: Computer Science and
Computational Mathematics

Advisor: Prof. Dr. Caetano Traina Júnior

**USP – São Carlos
July 2018**

AGRADECIMENTOS

Agradeço aos meus pais, Vilma e Sérgio e aos meus irmãos Cristina e Giovani, por todo o apoio.

À minha namorada Patrícia, pela companhia, apoio e compreensão.

Aos Professores Dr. Caetano Traina Jr. e Dr^a Agma J. M. Traina. e aos demais professores do GBdI.

Aos amigos do Grupo de Bases de Dados e Imagens (GBdI) pelas ideias e companhia.

Aos meus amigos de apartamento e colegas de disciplinas.

Ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP) e projeto RESCUER pelo apoio financeiro.

Ao Instituto de Ciências Matemáticas e de Computação (ICMC) e à Universidade de São Paulo (USP) por me concederem a oportunidade de me desenvolver pessoal e profissionalmente.

*“A ciência não é perfeita.
Ela as vezes é mau usada;
É apenas uma ferramenta, mas é a melhor ferramenta que temos.
Auto-corrigindo, sempre mudando,
aplicável a todas as coisas.
Com esta ferramenta, nós vencemos o impossível.”
(Carl Sagan)*

RESUMO

LAVERDE, N. A. **Desenvolvimento de operadores de agrupamento por similaridade em SGBD relacionais**. 2018. 102 p. Dissertação (Mestrado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2018.

O operador de agrupamento e as funções de agregação são as principais ferramentas utilizadas para sumarizar dados em um Sistema de Gerenciamento de Base de Dados Relacionais (SGBDR). O operador de agrupamento funciona criando partições nos dados utilizando comparações por identidade, e permite que sejam aplicadas funções de agregação que retornam um único valor representando o grupo como um todo. Entretanto, para dados métricos, agrupamento utilizando identidade tem pouca utilidade. Neste caso, adotar o conceito de similaridade é frequentemente uma abordagem mais promissora. A literatura apresenta alguns operadores que podem agrupar os dados utilizando similaridade. Todos eles utilizam um limiar de valor de distância para atribuir os elementos aos grupos. No entanto, estes operadores não obtêm resultados satisfatórios quando a distribuição dos dados apresenta variações significativas na densidade de objetos em diferentes regiões do espaço. Para alcançar melhores resultados nestas situações, propusemos um novo operador que atribui os grupos utilizando uma eleição envolvendo grupos já atribuídos. Também propusemos generalizações, para os operadores existentes e propostos, para trabalhar com uma quantidade de vizinhos mais próximos e aproximação dos vizinhos mais próximos ao invés de um limiar de distância. Para possibilitar a inclusão destes operadores em SGBDR, propusemos uma extensão à *Structured Query Language* (SQL) e novas funções de agregação. Implementamos estes operadores em nosso *framework* em C++ usando a biblioteca *Arboretum*. Para avaliar os métodos propostos, analisamos tanto qualidade dos resultados quanto tempo de execução, utilizando conjuntos de dados reais e sintéticos. Os operadores propostos alcançaram melhores resultados quanto à qualidade de resultados, e mantiveram os tempos de execução similares. Os operadores que utilizam aproximação aos vizinhos mais próximos produziram resultados de qualidade similar quando comparados aos operadores que utilizando os vizinhos mais próximos, podendo ser executados em menor tempo que estes.

Palavras-chave: Agrupamento por Similaridade, Agregação por Similaridade, Similaridade, SGBDR.

ABSTRACT

LAVERDE, N. A. **Development of similarity group operators in Relational DBMS.** 2018. 102 p. Dissertação (Mestrado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2018.

The grouping operator and aggregation functions are the primary tools used to summarize data inside a Relational Database Management Systems (RDBMS). The grouping operator works creating partitions in data using identity comparisons, and allow applying aggregation functions that return a single value that represent the entire group. However, for metric data, grouping by identity is seldom useful. In this case, adopting the concept of the similarity is often a better approach. The literature presents few operators that can group data using similarity. All of them use a distance threshold value to assign the elements in groups. However, these operators do not achieve satisfactory results when the data distribution present a significant variation in the density of objects in different regions of the space. To achieve better results in these situations, we have proposed a novel operator that assign groups using an election involving already assigned groups. We also proposed generalizations to existing and proposed operators to work with an amount of nearest neighbors and approximate neighbors instead of a distance threshold. To support these operators in RDBMS, we propose an extension to *Structured Query Language* (SQL) and new aggregation functions. Our proposed algorithms can run the proposed and existing operators. We implemented these operators in our framework in C++ using *Arboretum* library. To evaluate the proposed methods, we assess both results quality and the execution time, using both real and synthetic datasets. The proposed operators achieved better results comparing the quality and maintained similar executing time. The operators that use the approximate nearest neighbors produced similar quality results comparing with the operators that use the exact neighbors and can execute faster than that.

Keywords: Similarity Group, Similarity Aggregation, Similarity, RDBMS.

LISTA DE ILUSTRAÇÕES

Figura 1 – Consultas por similaridade	41
Figura 2 – Junções por similaridade	42
Figura 3 – Possíveis particionamentos para um conjunto de dados em duas dimen- sões.	43
Figura 4 – Possível particionamento de um conjunto de dados utilizando Gh-Tree.	44
Figura 5 – Possível particionamento de um conjunto de dados utilizando VP-Tree.	45
Figura 6 – Possível particionamento de um conjunto de dados utilizando M-Tree.	45
Figura 7 – Possível particionamento de um conjunto de dados utilizando Slim-Tree.	46
Figura 8 – Comportamento de uma função (d_1, d_2, p_1, p_2) -sensitiva.	47
Figura 9 – Exemplo de consulta usando o operador Cluster-by.	57
Figura 10 – Exemplo de diferentes consultas com <i>similarity group by</i>	59
Figura 11 – Semântica dos predicados SGB com $\epsilon = 3$	60
Figura 12 – Exemplos de <i>simsets</i> produzidos.	63
Figura 13 – Representação gráfica de <i>skyline</i>	64
Figura 14 – Pontos em um espaço Euclidiano bi-dimensional.	70
Figura 15 – Conjuntos de dados sintéticos.	81
Figura 16 – Exemplo de imagens dos conjuntos de dados Corel e GHIM.	81
Figura 17 – Valores de AMI para conjuntos de dados sintéticos usando <i>Range-SGB</i>	82
Figura 18 – Valores de AMI para conjuntos de dados sintéticos usando <i>KNN-SGB</i>	83
Figura 19 – Valores de AMI para conjuntos de dados sintéticos usando <i>ANN-SGB</i>	83
Figura 20 – Valores de AMI para conjuntos de dados reais usando <i>Range-SGB</i>	84
Figura 21 – Valores de AMI para conjuntos de dados reais usando <i>KNN-SGB</i>	84
Figura 22 – Valores de AMI para conjuntos de imagens usando <i>ANN-SGB</i>	85
Figura 23 – Tempo de execução (em segundos) para os operadores <i>Range-SGB</i> nos conjuntos de dados sintéticos.	85
Figura 24 – Tempo de execução (em segundos) para os operadores <i>KNN-SGB</i> nos conjuntos de dados sintéticos.	85
Figura 25 – Tempo de execução (em segundos) para os operadores <i>ANN-SGB</i> nos conjuntos de dados sintéticos.	86
Figura 26 – Tempo de execução para os operadores <i>Range-SGB</i> nos conjuntos de dados reais.	86
Figura 27 – Tempo de execução para os operadores <i>KNN-SGB</i> nos conjuntos de dados reais.	86

Figura 28 – Valores de AMI para conjuntos de imagens usando <i>ANN-SGB</i>	87
Figura 29 – Particionamentos obtidos sobre o conjunto Aggregation	89
Figura 30 – Particionamentos obtidos sobre o conjunto Compound	90
Figura 31 – Particionamentos obtidos sobre o conjunto D31	91
Figura 32 – Particionamentos obtidos sobre o conjunto R15	92

LISTA DE QUADROS

Quadro 1 – Comparação do tempo de execução das implementações da estrutura <i>union-find</i>	51
Quadro 2 – Comparação entre os trabalhos relacionados	67
Quadro 3 – Funções de agregação	77

LISTA DE ALGORITMOS

Algoritmo 1 – SGB-Any	74
Algoritmo 2 – SGB-All	75
Algoritmo 3 – SGB-Vote	76

LISTA DE CÓDIGOS-FONTE

Código-fonte 1 – Funções de agregações em SQL	35
Código-fonte 2 – Agregações com agrupamento em SQL	36
Código-fonte 3 – Sintaxe do GROUP BY estendido com agrupamento definido pelo usuário.	54
Código-fonte 4 – Sintaxe do GROUP BY estendido com funções de distância de- finidas pelo usuário.	54
Código-fonte 5 – Sintaxe do operador <i>ClusterRank</i>	55
Código-fonte 6 – Sintaxe do operador Cluster-by	55
Código-fonte 7 – Exemplo de consulta usando o operador Cluster-by.	56
Código-fonte 8 – Sintaxe do SGB-U	57
Código-fonte 9 – Sintaxe do SGB-A	58
Código-fonte 10 – Sintaxe do SGB-D	58
Código-fonte 11 – Sintaxe do SGB-Any	58
Código-fonte 12 – Sintaxe do SGB-All	59
Código-fonte 13 – Exemplo de consulta usando o operador CUBE.	61
Código-fonte 14 – Exemplo de funções de janelamento em SQL	62
Código-fonte 15 – Sintaxe do operador <i>skyline</i>	65
Código-fonte 16 – Sintaxe proposta para os operadores SGB.	72
Código-fonte 17 – Sintaxe proposta para os operadores SGB com funções de jane- lamento.	73

LISTA DE TABELAS

Tabela 1 – Relação funcionários.	34
Tabela 2 – Resultado de agregações na relação “funcionários”.	35
Tabela 3 – Resultado de agrupamento por “departamento” e agregações na relação “funcionários”.	37
Tabela 4 – Resultado de agrupamento por “cargo” e agregações na relação “funcionários”.	37
Tabela 5 – Resultado de operador CUBE sobre atributos da relação “funcionários”.	61
Tabela 6 – Exemplo de resultado funções de janelamento.	62
Tabela 7 – Descrição dos conjuntos de dados.	80

LISTA DE ABREVIATURAS E SIGLAS

k -NNq	<i>k-Nearest Neighbors: query</i> – consulta aos k -vizinhos mais próximos
SGB-Vote	<i>Similarity Group-By Vote</i>
AMI	<i>Adjusted Mutual Information</i> – Informação Mútua Ajustada
ANN-SGB	<i>Approximate k-Nearest Neighbors Similarity Group-By</i>
ANNq	<i>Approximate k-Nearest Neighbors query</i>
CAPES	Coordenação de Aperfeiçoamento de Pessoal de Nível Superior
CNPq	Conselho Nacional de Desenvolvimento Científico e Tecnológico
FAPESP	Fundação de Amparo à Pesquisa do Estado de São Paulo
FMI-SiR	Features, Metrics and Indexes for Similarity Retrieval
GBdI	Grupo de Bases de Dados e Imagens
Gh-Tree	<i>Generalized Hyperplane Tree</i> – Árvore de Hiperplano Generalizado
ICEIS	<i>International Conference on Enterprise Information Systems</i>
ICMC	Instituto de Ciências Matemáticas e de Computação
ISO	<i>International Organization for Standardization</i>
KNN-SGB	<i>k-Nearest Neighbors Similarity Group-By</i>
LSH	<i>Locality-Sensitive Hashing</i> – Particionamento sensível a localidade
M-tree	Metric Tree – Árvore Métrica
MAM	Métodos de Acesso Métrico
MI	<i>Mutual Information</i> – Informação Mútua
MST	<i>Minimum Spanning Tree</i> – Árvore de Extensão Mínima
MVP-Tree	<i>Multivantage Point Tree</i> – Árvore por múltiplos pontos de vantagem
PostgreSQL-IE	PostgreSQL with Image-handling Extension
Range-SGB	<i>Range Similarity Group-By</i>
RDBMS	Relational Database Management Systems
RI	Relações de Identidade
RO	Relação de Ordem
Rq	<i>Range Query</i> – consulta por abrangência
SGB	<i>Similarity Group-By</i>
SGB-A	<i>Supervised Similarity Group Around</i>
SGB-All	<i>Similarity Group-By All</i>
SGB-Any	<i>Similarity Group-By Any</i>

SGB-D	<i>Supervised SGB using Delimiters</i>
SGB-U	<i>Unsupervised Similarity Group-by</i>
SGBD	Sistemas de Gerenciamento de Bases de Dados
SGBDR	Sistemas de Gerenciamento de Bases de Dados Relacionais
SimbA	Similarity by Authorities
SimDB	similarity-aware database management system
SIREN	SImilarity Retrieval ENgine
SISAP	<i>Similarity Search and Applications</i>
SQL	<i>Structured Query Language</i>
STL	Standard Template Library
USP	Universidade de São Paulo
VP-Tree	<i>Vantage Point Tree</i> – Árvore por ponto de vantagem
Simsets	<i>similarity sets</i>

LISTA DE SÍMBOLOS

σ — Operador de Seleção

π — Operador de Projeção

$\times$ — Operador de Produto Cartesiano

ρ — Operador de Renomeação

$\cup$ — União de Conjuntos

$-$ — Diferença de Conjuntos

γ — Operador de Agrupamento

τ — Operador de Eliminação de Duplicatas

d — Função de distância

$\mathbb{S}$ — Domínio de dados métricos

$\mathbb{R}^+$ — Conjunto dos números reais não negativos

s_i — Elemento de um espaço métrico

G_i — Atributo de agregação

S_i — Segmentação do domínio de G_i , em segmentos sem sobreposição

L — Lista de atributos e funções de agregação

SUMÁRIO

1	INTRODUÇÃO	29
1.1	Objetivos	30
1.2	Organização da Monografia	30
2	CONCEITOS FUNDAMENTAIS	33
2.1	Modelo Relacional	33
2.1.1	<i>Funções de Agregação</i>	35
2.1.2	<i>Operador de Agrupamento</i>	36
2.2	Similaridade	38
2.2.1	<i>Espaços Métricos</i>	39
2.2.1.1	<i>Funções de Distância</i>	40
2.2.2	<i>Consultas por Similaridade</i>	41
2.2.3	<i>Consultas Aproximadas por Similaridade</i>	42
2.2.4	<i>Métodos de Acesso Métrico</i>	43
2.2.5	<i>Hashing por Similaridade</i>	46
2.2.6	<i>Suporte à Similaridade em SGBDR</i>	48
2.3	Clustering	48
2.3.1	<i>Métodos de Avaliação de Clusters</i>	49
2.4	Estruturas de Dados	50
2.5	Considerações	51
3	TRABALHOS RELACIONADOS	53
3.1	Agrupamento por Similaridade em SGBDR	53
3.1.1	<i>Agrupamentos Definidos Pelo Usuário</i>	53
3.1.2	<i>Operador ClusterRank</i>	54
3.1.3	<i>Operador Cluster By</i>	55
3.1.4	<i>Similarity Group-By</i>	56
3.1.5	<i>Similarity Group-By Multidimensional</i>	58
3.2	Outros Operadores	60
3.2.1	<i>Operador CUBE</i>	60
3.2.2	<i>Funções de Janelamento</i>	61
3.2.3	<i>Simsets</i>	62
3.2.4	<i>Operador de Skyline</i>	64

3.3	Considerações	65
4	MÉTODOS PROPOSTOS	69
4.1	Operadores	69
4.1.1	<i>Operador SGB-Vote</i>	69
4.1.2	<i>SGB baseado em vizinhos mais próximos: KNN-SGB e ANN-SGB</i> .	71
4.1.3	<i>Identificação dos Operadores</i>	71
4.2	Extensão SQL	72
4.3	Algoritmos	73
4.3.1	<i>SGB-Any</i>	73
4.3.2	<i>SGB-All</i>	74
4.3.3	<i>SGB-Vote</i>	74
4.4	Funções de Agregação	75
4.5	Considerações	77
5	EXPERIMENTOS	79
5.1	Implementação	79
5.2	Conjuntos de Dados	80
5.3	Resultados	82
5.3.1	<i>Qualidade dos Agrupamentos</i>	82
5.3.2	<i>Tempo de Execução</i>	84
5.3.3	<i>Particionamentos Obtidos</i>	87
5.4	Considerações	88
6	CONCLUSÕES	93
6.1	Contribuições	93
6.2	Publicações	94
6.3	Sugestões de Futuras Pesquisas	94
	REFERÊNCIAS	97

INTRODUÇÃO

Sistemas de Gerenciamento de Bases de Dados (SGBD) são utilizados para armazenar, recuperar e analisar grandes volumes de dados, provendo o acesso para que outras aplicações possam fazer uso dos dados. Quando utilizam o modelo de dados relacional, usualmente eles são chamados de Sistemas de Gerenciamento de Bases de Dados Relacionais (SGBDR). A linguagem de consulta usada na maioria dos SGBDR é a SQL, uma linguagem declarativa baseada no modelo relacional.

Os SGBDRs foram originalmente desenvolvidos visando tratar alguns tipos de dados convencionalmente chamados de *dados escalares*. Atualmente os SGBDR permitem manipular também outros tipos de dados, neste documento chamamos de *dados complexos*, como imagens, vídeo, áudio, séries temporais, etc. Usamos este termo para designar dados que não são comparados diretamente, mas indiretamente usando uma função de mapeamento. Os operadores usados para comparação dos dados escalares normalmente não podem ser utilizados para dados complexos, pois dados complexos em geral não possuem algumas propriedades que os dados escalares possuem. Uma alternativa para comparar estes tipos de dados são as chamadas *consultas por similaridade*.

Os SGBDR possuem diversos operadores que possibilitam consultas aos dados como seleção, junção, funções de agregação e agrupamento. O operador de seleção seleciona as tuplas que satisfazem uma condição. Os operadores de junção combinam tuplas de duas relações com uma condição de junção. As *funções de agregação* sumarizam coleções de valores em um único valor. Algumas funções de agregação muito comumente utilizadas em SGBDR são funções estatísticas como média, contagem, soma, máximo e mínimo. O operador de *agrupamento* cria subconjuntos das tuplas de uma relação, reunindo as tuplas de acordo com o valor de seus atributos. Funções de agregação geralmente são utilizadas em conjunto com o operador de agrupamento, retornando um valor por grupo ao invés de um valor por relação.

Os SGBDR implementam o operador de agrupamento utilizando comparações por identidade, assim cada grupo gerado tem o mesmo valor nos atributos usados para o agrupamento. Porém, no contexto de dados complexos, esse tipo de comparador não acrescenta muita semântica aos resultados. Desta forma, são necessários novos operadores que utilizem o conceito de similaridade para que possam ser aplicados em domínios de dados complexos.

Além disto, a inclusão deste tipo de operação no ambiente de execução de um SGBDR permite aproveitar os índices construídos sobre tais dados, podendo assim acelerar a execução. Outro ponto interessante é que os operadores podem ser combinados com outras operações presentes nos SGBDR. Por exemplo, podendo ser combinado com operadores de junção e filtragem.

1.1 Objetivos

O objetivo principal deste trabalho é o desenvolvimento de operadores de agrupamento por similaridade, que sejam genéricos, e que possam ser utilizados sobre qualquer domínio de dados complexo. Além disso, para apoiar a utilização prática dos novos operadores, este projeto também inclui:

- Desenvolvimento de funções de agregação que possam ser utilizadas em conjunto com os operadores de agrupamento por similaridade. As funções de agregação clássicas continuam úteis também para os novos operadores, porém é importante disponibilizar novas funções de agregação específicas para serem usadas quando o critério de agrupamento é baseado em similaridade;
- Desenvolvimento de algoritmos para a execução dos operadores de agrupamento por similaridade. Existem algumas abordagens para a execução do agrupamento clássico, tais como: a busca sequencial, a ordenação e o uso de índices *hash*. Desenvolvemos algoritmos genéricos para os operadores propostos, que podem utilizar diversas abordagens para a execução. Como por exemplo, abordagens baseadas em métodos de acesso métrico e em estruturas de indexação baseadas em *hash*;
- Especificação de uma extensão para a linguagem SQL para que os SGBDR possam incorporar os novos operadores de agrupamento. A extensão foi planejada como uma alteração da cláusula `GROUP BY`, de forma que o usuário possa indicar os parâmetros necessários para a execução de consultas com este tipo de agrupamento.

1.2 Organização da Monografia

Os demais capítulos desta monografia estão organizados da seguinte forma:

- O [Capítulo 2](#) explica os principais conceitos básicos relacionados ao modelo relacional e que são úteis para o desenvolvimento deste trabalho, em particular explorando conceitos do operador agrupamento e das funções de agregação. Também são introduzidos conceitos relacionados à similaridade, tais como espaços métricos, funções de distância e os diferentes tipos de consultas por similaridade. São apresentadas algumas estruturas que podem ser utilizadas para agilizar a execução dessas consultas e como pode ser feita sua inclusão em SGBDR. Além disto, são detalhadas algumas técnicas de *clustering*¹ e algumas técnicas para avaliação da qualidade dos agrupamentos. Também são apresentadas algumas estruturas de dados que foram empregadas nos algoritmos propostos;
- O [Capítulo 3](#) aborda trabalhos correlatos, detalhando trabalhos que tratam de agrupamento por similaridade em SGBDR e de alguns outros operadores interessantes no contexto deste trabalho;
- O [Capítulo 4](#) apresenta o operador *SGB-Vote*, proposto neste trabalho. Além disto, também são propostas variações dos operadores existentes para utilizar predicados baseados no número de vizinhos mais próximos ao invés de um limiar de distância. O capítulo traz novos algoritmos para a execução dos operadores além de uma extensão à linguagem SQL. Também foram desenvolvidas algumas funções de agregação baseadas em similaridade e funções de agregação que podem ser úteis para tratar alguns domínios complexos;
- O [Capítulo 5](#) detalha a implementação dos operadores e o ambiente utilizado para execução dos algoritmos que foram implementados em um arcabouço (*framework*) próprio utilizando a linguagem C++. O capítulo apresenta a forma como foram conduzidos os experimentos e discute os resultados obtidos pelos operadores. Os experimentos executados sobre diversos conjuntos de dados reais e sintéticos mostraram que os novos operadores propostos podem trazer resultados superiores em termos de qualidade aos dos operadores existentes na literatura;
- O [Capítulo 6](#) traz uma sumarização das contribuições do trabalho e apresenta possíveis trabalhos futuros.

¹ Neste texto, o termo *clustering* é utilizado para designar as técnicas de agrupamento existentes em mineração de dados.

CONCEITOS FUNDAMENTAIS

Neste capítulo são abordados os principais conceitos utilizados neste trabalho. A [Seção 2.1](#) aborda conceitos do *modelo relacional*, com foco em operadores de agregação na [Subseção 2.1.1](#) e o operador de agrupamento na [Subseção 2.1.2](#).

Na [Seção 2.2](#) são abordados conceitos relacionados à similaridade. Tratam-se os espaços métricos, na [Subseção 2.2.1](#), métodos de acesso métrico na [Subseção 2.2.4](#), consultas por similaridade na [Subseção 2.2.2](#) e a inclusão de consultas por similaridade em SGBDR na [Subseção 2.2.6](#).

A [Seção 2.3](#) traz uma breve introdução sobre *clustering*, que envolve diversas técnicas utilizadas para encontrar grupos em conjuntos dados. Por fim, a [Seção 2.5](#) traz uma sumarização do capítulo.

2.1 Modelo Relacional

O modelo relacional foi originalmente definido por [Codd \(1970\)](#) e representa uma base de dados como uma coleção de relações, tendo sua definição teórica baseada na teoria dos conjuntos e na lógica de predicados.

Numa implementação do Modelo Relacional (em SGBDR), cada relação pode ser imaginada como uma *tabela* de valores. Cada linha da tabela representa uma coleção de dados relacionados, usualmente correspondendo a um objeto ou relacionamento no mundo real. Na terminologia do modelo relacional, uma linha da tabela é chamada *tupla*, cada coluna é chamado *atributo*, a tabela é chamada *relação* e o tipo de dados que pode aparecer em cada coluna é chamado *domínio* ([ELMASRI; NAVATHE, 2011](#)).

A [Tabela 1](#) ilustra a terminologia utilizada. Aqui, “funcionários” é o nome da relação, cada linha da tabela é uma tupla (exceto o cabeçalho) e são atributos “nome”, “departamento”, “cargo”, “contrato” e “salário”. Cada atributo possui seu domínio, por

Tabela 1 – Relação funcionários.

funcionários				
nome	departamento	cargo	contrato	salário
Antônio	Vendas	Vendedor(a)	17/03/2007	2400,00
Edgar	Vendas	Vendedor(a)	21/11/2008	2300,00
Flávia	Vendas	Vendedor(a)	15/12/2010	2000,00
João	Gerência	Gerente Geral	07/01/1999	7000,00
José	Gerência	Gerente Vendas	03/06/2000	5000,00
Júlia	Marketing	Diretor(a)	01/06/2000	3500,00
Luiza	Vendas	Vendedor(a)	13/05/2001	3000,00
Marcia	RH	Diretor(a)	15/01/2005	3500,00
Mário	Vendas	Vendedor(a)	12/07/2005	2500,00
Milton	Marketing	Auxiliar	13/11/2007	2100,00

exemplo, “nome”, “departamento” e “cargo” eles podem ser cadeias de caracteres de tamanho fixo ou variável, “contrato” é uma data no formato dd/mm/aaaa e “salário” é um valor numérico.

Um conceito importante é a indicação do estado *nulo* (**NULL**) para um atributo, usado para indicar valores que não são conhecidos ou não são aplicáveis à tupla. A *Álgebra Relacional* é uma linguagem formal para especificar consultas no modelo de dados relacional (ELMASRI; NAVATHE, 2011). Os operadores fundamentais da Álgebra Relacional são:

- *Seleção* (σ) : seleciona tuplas que satisfaçam a uma determinada condição;
- *Projeção* (π) : projeta uma lista de atributos;
- *Produto cartesiano* ($\times$) : combina tuplas de duas relações, sem a necessidade de argumentos em comum. A relação resultante contém todas as combinações de todas as tuplas possíveis das duas relações;
- *Renomeação* (ρ) : renomeia o nome da relação e/ou dos atributos de uma relação;
- Também são operadores fundamentais os operadores de conjuntos, *união* ($\cup$) e *diferença de conjuntos* ($-$).

Linguagens como SQL possuem diversas operações de grande importância para as aplicações. Todas elas podem ser definidas a partir dos operadores fundamentais. No entanto eles tanto facilitam a representação de expressões de consulta quanto permitem desenvolver algoritmos muito mais eficientes para realizar a consulta do que seria possível executando cada operador fundamental necessário para expressar a mesma consulta. Desta forma, são incluídos diversos operadores da linguagem na Álgebra Relacional. Es-

tes operadores são usualmente chamados de *operadores estendidos da Álgebra Relacional* (GARCIA-MOLINA; ULLMAN; WIDOM, 2000). Eles aumentam o poder de expressividade da Álgebra Relacional (ELMASRI; NAVATHE, 2011).

2.1.1 Funções de Agregação

Funções de agregação são utilizadas para sumarizar ou agregar valores de um atributo da relação. Exemplo de funções frequentemente utilizadas são: *soma* (SUM), *média* (AVG), *máximo* (MAX), *mínimo* (MIN) e *contagem* (COUNT) (GARCIA-MOLINA; ULLMAN; WIDOM, 2000; ELMASRI; NAVATHE, 2011).

As funções SUM e AVG produzem, respectivamente, a soma e a média de um atributo com valores numéricos. MIN e MAX, quando aplicados a um atributo com valores numéricos, produzem o menor e o maior valor, respectivamente, e quando aplicados em colunas com cadeias de caracteres produzem, respectivamente, o primeiro ou o último em ordem alfabética. COUNT produz o número de valores (não necessariamente distintos) em uma coluna, e quando aplicado a qualquer atributo da relação produz o número de tuplas na relação, incluindo duplicatas (GARCIA-MOLINA; ULLMAN; WIDOM, 2000). Em SQL, as funções de agregação aparecem diretamente na cláusula SELECT de uma consulta como exemplificado no [Código-fonte 1](#).

A [Tabela 2](#) ilustra o resultado da execução das funções de agregação de contagem, mínimo, máximo, média e soma aplicadas sobre o atributo “salário” da relação “funcionários” que foi definida na [Tabela 1](#).

Código-fonte 1 – Funções de agregações em SQL

```

1:  SELECT
2:      departamento ,
3:      COUNT(*) ,
4:      MIN(salário) ,
5:      MAX(salário) ,
6:      AVG(salário) ,
7:      SUM(salário)
8:  FROM funcionários;

```

Tabela 2 – Resultado de agregações na relação “funcionários”.

count(*)	min(salário)	max(salário)	avg(salário)	sum(salário)
10	2000,00	7000,00	3330,00	33300,00

Existe uma particularidade na agregação de atributos que possuem valor nulo.

Em geral, esses valores são descartados quando uma função de agregação é aplicada a um atributo (ELMASRI; NAVATHE, 2011).

2.1.2 Operador de Agrupamento

Em muitos casos deseja-se saber o valor de uma função de agregação aplicada a partes da relação ao invés da relação toda. Um *agrupamento* cria uma partição das tuplas de uma relação, reunindo em um mesmo grupo tuplas que compartilham o(s) mesmo(s) valor(es) de um subconjunto dos atributo(s). Desta forma, as funções de agregação são aplicadas sobre cada grupo e retornam um valor por grupo ao invés de um valor por relação.

Existem diversas notações de agrupamento em Álgebra Relacional. Neste texto será usada a notação de Garcia-Molina, Ullman e Widom (2000), usando para representar um agrupamento a seguinte expressão:

$$\gamma_L(T)$$

O símbolo γ é usado para representar o operador de agrupamento, o qual opera sobre uma relação T e uma lista de atributos L . A lista L pode conter um ou mais atributos da relação T , chamados de *atributos de agrupamento*, e funções de agregação aplicadas sobre atributos da relação T , chamados de *atributos de agregação* (GARCIA-MOLINA; ULLMAN; WIDOM, 2000).

Usando essa notação, é possível descrever algebricamente um agrupamento pelo atributo “departamento” na relação “funcionários” usando a agregação de contagem sobre qualquer atributo e as agregações de mínimo, máximo, média e soma sobre o atributo salário. A expressão ficaria da seguinte forma:

$$\gamma_{\text{departamento}, \text{count}(*), \text{min}(\text{salrio}), \text{max}(\text{salrio}), \text{avg}(\text{salrio}), \text{sum}(\text{salrio})}(\text{funcionrios})$$

Em linguagem SQL o operador de agrupamento é descrito pela cláusula GROUP BY. Os atributos de agrupamento devem estar listados na cláusula GROUP BY, enquanto as funções de agregação são indicadas na cláusula SELECT. O Código-fonte 2 expressa a mesma consulta na linguagem SQL.

Código-fonte 2 – Agregações com agrupamento em SQL

```

1:      SELECT
2:          departamento ,
3:          COUNT(*) ,
4:          MIN(salário) ,
5:          MAX(salário) ,
6:          AVG(salário) ,

```

```

7:      SUM(salário)
8:      FROM funcionários
9:      GROUP BY departamento;

```

A consulta expressa no [Código-fonte 2](#) traria como resposta o resultado das agregações para cada valor diferente de departamento. O resultado desta consulta está exemplificado na [Tabela 3](#)

Tabela 3 – Resultado de agrupamento por “departamento” e agregações na relação “funcionários”.

departamento	count(*)	min(salário)	max(salário)	avg(salário)	sum(salário)
Gerência	2	5000,00	7000,00	6000,00	12000,00
RH	1	3500,00	3500,00	3500,00	3500,00
Marketing	2	2100,00	3500,00	2800,00	5600,00
Vendas	5	2000,00	3000,00	2440,00	12200,00

Alterando os atributos de agrupamento, pode-se mudar o número de grupos e o resultado das agregações. Para o caso de mais de um atributo de agrupamento, a ordem dos atributos também pode influenciar no resultado.

Por exemplo, executando um agrupamento por “cargo” na relação “funcionários” e as mesmas funções de agregação utilizadas anteriormente, haveriam 5 grupos formados ao invés dos 4 grupos formados no agrupamento pelo atributo “departamento”. O resultado deste agrupamento pode ser visualizado na [Tabela 4](#).

Tabela 4 – Resultado de agrupamento por “cargo” e agregações na relação “funcionários”.

cargo	count(*)	min(salário)	max(salário)	avg(salário)	sum(salário)
Auxiliar	1	2100,00	2100,00	2100,00	2100,00
Diretor(a)	2	3500,00	3500,00	3500,00	7000,00
Gerente Geral	1	7000,00	7000,00	7000,00	7000,00
Gerente Vendas	1	5000,00	5000,00	5000,00	5000,00
Vendedor(a)	5	2000,00	3000,00	2440,00	12200,00

Em SQL existe uma cláusula que executa uma filtragem após o agrupamento, a cláusula **HAVING**. A cláusula aparece conjuntamente com a cláusula **GROUP BY** e provê uma condição sobre a informação agregada de cada grupo. Desta forma, apenas os grupos que satisfazem à condição dada são retornados no resultado da consulta ([ELMASRI; NAVATHE, 2011](#)). Em Álgebra Relacional, esse cláusula corresponde a um operador de seleção aplicado sobre a relação resultante do agrupamento.

A Álgebra Relacional também possui um operador de eliminação de duplicatas. Para representar este operador usualmente é usado o símbolo τ . Este operador é redundante pois o operador de eliminação de duplicatas pode ser considerado um caso especial

do operador de agrupamento aplicado sobre todos os atributos da relação e sem a aplicação de funções de agregações. Como o resultado do agrupamento contém exatamente uma tupla para cada grupo o efeito deste tipo de agrupamento é a eliminação de duplicatas (GARCIA-MOLINA; ULLMAN; WIDOM, 2000).

2.2 Similaridade

Em aplicações usando dados escalares, a maior parte dos dados são informações textuais ou numéricas. Recentemente, os SGBDR passaram a tratar outros tipos de dados como imagens, áudio, vídeo (ELMASRI; NAVATHE, 2011). Dados textuais ou numéricos são comumente denominados *dados escalares* porque correspondem a apenas um valor que não pode ser decomposto e são comparados diretamente. Por outro lado, os dados como imagens, áudio, vídeo, séries temporais, dados geográficos são usualmente chamados de *dados complexos*, porque sua comparação requer um mapeamento preliminar.

Os dados escalares atendem às propriedades de Relações de Identidade (RI) e Relação de Ordem (RO). A RI garante que os dados podem ser comparados por igualdade utilizando comparadores $=$ e $\neq$, enquanto as propriedades das RO garantem que os dados possam ser consultados usando comparadores $<$, $\leq$, $\geq$ e $>$.

Os operadores de comparação para os dados complexos, por sua vez, em geral não atendem às propriedades das RO, ou seja, não podem ser consultados usando os operadores $<$, $\leq$, $\geq$ e $>$. A propriedade RI pode ser utilizada para a comparação de dados complexos. Porém nesses tipos de dados a comparação por comparadores $=$ e $\neq$ normalmente em geral não tem significado (ZEZULA *et al.*, 2006).

Para comparar dados complexos é necessário fazer um mapeamento para um outro domínio de dados, e definir as funções de comparação nesse domínio. Assim, para comparar dados dados complexos usualmente utiliza-se o conceito de *similaridade* para quantificar a proximidade entre objetos. Buscas por proximidade possuem aplicação em diversas áreas, incluindo recuperação de informação multimídia, mineração de dados, reconhecimento de padrões, aprendizado de máquina, etc. (ZEZULA *et al.*, 2006).

A noção de similaridade é frequentemente subjetiva, uma vez que diferentes usuários podem ter diferentes percepções da similaridade de um mesmo par de objetos. Além disto, a similaridade para o mesmo par de objetos pode seguir diferentes medidas de acordo com o contexto (BARIONI *et al.*, 2010). Por exemplo, imagens podem ser comparadas considerando que são similares pela cor, ou pela forma, etc. Isso é contemplado definindo diferentes funções de comparação e/ou diferentes mapeamentos dos dados complexos.

O processo de avaliação de similaridade usualmente é complexo e especializado para um determinado domínio. Para a avaliação da similaridade, deve ser definida por

uma função que compara um par de objetos complexos e retorna um valor numérico. Este valor numérico quantifica a (dis)similaridade entre os objetos (BARIONI *et al.*, 2010). Essa função, pode ser uma *função de similaridade* ou uma *função de distância*. As funções de similaridade retornam valores maiores para objetos mais similares, enquanto as funções de distância retornam valores menores para objetos mais similares, retornando zero para objetos idênticos (ZEZULA *et al.*, 2006). Usualmente funções de distâncias são mais utilizadas, pois a existência de uma referência fixa para objetos idênticos (dois objetos iguais têm distância zero) facilita usar suas propriedades em algoritmos projetados para agilizar buscas.

Para a comparação dos objetos complexos, normalmente não se utilizam os dados complexos em seu formato original. Ao invés disso, são extraídas características que descrevem o dado complexo em nível mais “semântico”. As características são extraídas por um *extrator de características*, que recebe um objeto complexo como entrada e tem um *vetor de características* como saída. As características podem ser interpretadas como uma assinatura matemática que representa o objeto respeitando algum critério (BARIONI *et al.*, 2010).

Existem diversos extratores de características para cada domínio de dados. Usando imagens como exemplo, os principais extratores utilizados são baseados em cor, textura ou forma. Para vídeos, usualmente são utilizados conjuntos de quadros que mostram uma ação. Para cada conjunto, usualmente um quadro representativo é escolhido, do qual são extraídas as características. Além disso, outras características podem ser extraídas de outros elementos contidos no vídeo, como o áudio, a temporização entre os conjuntos de quadro e as legendas (BARIONI *et al.*, 2010).

2.2.1 Espaços Métricos

Uma formalização útil é a noção matemática de *espaço métrico*. A base matemática de espaço métrico permite que regras simples de particionamento e poda possam ser utilizadas. Estas regras são muito importantes para o desenvolvimento de estruturas de índices (ZEZULA *et al.*, 2006).

Um espaço métrico $\mathbb{M}$ é definido como $\mathbb{M} = \langle \mathbb{S}, d \rangle$, onde $\mathbb{S}$ é o domínio dos dados e d é uma função de distância. As propriedades da função $d : \mathbb{S} \times \mathbb{S} \rightarrow \mathbb{R}^+$ devem atender aos seguintes *postulados métricos*, sendo que s_1, s_2 e s_3 são elementos de $\mathbb{S}$ (ZEZULA *et al.*, 2006):

- **identidade dos indiscerníveis:** $d(s_1, s_2) = 0 \Rightarrow s_1 = s_2$;
- **não-negatividade:** $0 < d(s_1, s_2) < \infty, \forall s_1 \neq s_2$;
- **simetria:** $d(s_1, s_2) = d(s_2, s_1)$;

- **desigualdade triangular:** $d(s_1, s_2) \leq d(s_1, s_3) + d(s_3, s_2)$.

Uma função que satisfaça a estas propriedades é chamada de *métrica*. Uma importante vantagem de usar os espaços métricos é que eles permitem que se desenvolvam estruturas de indexação (BARIONI *et al.*, 2010).

2.2.1.1 Funções de Distância

As funções de distância computam a dissimilaridade entre dois objetos complexos. A indicação das funções que são adequadas para uma aplicação usualmente é feita por especialistas do domínio de dados. Existem diversas funções de distância definidas na literatura.

Dentre estas funções, pode-se destacar as funções da família *Minkowski*, que são comumente utilizadas (BARIONI *et al.*, 2010; ZEZULA *et al.*, 2006). As funções *Minkowski* são definidas para vetores de números reais n -dimensionais. Elas são chamadas métricas L_p pois correspondem a normas que dependem de um parâmetro numérico p (ZEZULA *et al.*, 2006). As funções da família *Minkowski* entre dois objetos de dimensão n , $s_1 = \{x_1 \dots x_n\}$ e $s_2 = \{y_1 \dots y_n\}$ são definidas como:

$$L_p(s_1, s_2) = \sqrt[p]{\sum_{i=1}^n |x_i - y_i|^p}$$

Desta família, algumas são funções mais conhecidas. A métrica L_1 é conhecida como *distância Manhattan* ou *City-Block*. A função de distância L_2 é chamada de *distância Euclidiana*. A métrica L_∞ é conhecida como *Chebychev* (BARIONI *et al.*, 2010; ZEZULA *et al.*, 2006).

A função de distância *Jaccard* é utilizada para medir a similaridade entre conjuntos. A função é definida como 1 menos a razão entre o tamanho da intersecção dos conjuntos pelo tamanho da união entre os conjuntos (LESKOVEC; RAJARAMAN; ULLMAN, 2014; ZEZULA *et al.*, 2006). Formalmente, dados dois conjuntos S_1 e S_2 , a Função de *Jaccard* é definida como:

$$d(S_1, S_2) = 1 - \frac{|S_1 \cap S_2|}{|S_1 \cup S_2|}$$

A distância do *Cosseno* pode ser utilizada em espaços que tenham um número fixo de dimensões n . A distância do cosseno é definida como o ângulo formado entre dois vetores, retornando um valor entre 0° e 180° , não importando o número de dimensões existentes no espaço (LESKOVEC; RAJARAMAN; ULLMAN, 2014). O cosseno do ângulo formado por dois objetos n -dimensionais $s_1 = \{x_1, \dots, x_n\}$ e $s_2 = \{y_1, \dots, y_n\}$ é o produto

vetorial entre s_1 e s_2 dividido pelos módulos dos vetores s_1 e s_2 :

$$\cos(\theta) = \frac{s_1 \cdot s_2}{\|s_1\|_2 \|s_2\|_2} = \frac{\sum_{i=1}^n x_i y_i}{\sqrt{\sum_{i=1}^n x_i^2} \sqrt{\sum_{i=1}^n y_i^2}}$$

Sobre este valor é aplicada a função arco cosseno que transforma o ângulo em um valor no intervalo 0-180 graus (LESKOVEC; RAJARAMAN; ULLMAN, 2014).

2.2.2 Consultas por Similaridade

Existem diversos tipos de consultas por similaridade, as mais conhecidas são a *Range Query* – consulta por abrangência (Rq) e a *k-Nearest Neighbors: query* – consulta aos k -vizinhos mais próximos (k -NNq) (ZEZULA *et al.*, 2006).

A consulta por abrangência sobre um conjunto $S \subseteq \mathbb{S}$ retorna todos os elementos s_i que são dissimilares a um elemento de consulta $s_q \in \mathbb{S}$ por no máximo um limiar ξ . Formalmente a consulta é definida de seguinte forma:

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

Uma consulta k -NNq sobre um conjunto $S \subseteq \mathbb{S}$ retorna os k elementos mais próximos ao elemento de consulta $s_q \in \mathbb{S}$. A consulta é definida formalmente como:

$$k\text{-NNq}(s_q, k) = \{S' \subseteq S, |S'| = k \wedge \forall s_i \in S', s_j \in S - S' : d(s_q, s_i) \leq d(s_q, s_j)\}$$

A Figura 1 exemplifica consultas realizadas em um espaço bi-dimensional euclidiano. Na Figura 1a temos um exemplo de consulta por abrangência com centro de consulta s_q e raio ξ . A Figura 1b representa uma consulta aos k -vizinhos mais próximos com $k = 4$ e s_q como centro de consulta.

Figura 1 – Consultas por similaridade

Existem também as *junções por similaridade*, que atuam sobre dois conjuntos de dados complexos. Uma junção por similaridade sobre dois conjuntos $S \subseteq \mathbb{S}$ e $R \subseteq \mathbb{S}$

recupera todos os pares $(s_i \in S, r_j \in R)$ que satisfaçam a condição de junção (ZEZULA *et al.*, 2006). Algoritmos para responder a junções por similaridade podem ser encontrados, por exemplo, em Jacox e Samet (2008) e em Silva, Aref e Ali (2010).

Os tipos de Junção mais comuns são a *junção pelos vizinhos mais próximos* (*k-NN Join*) e a *junção por abrangência* (*Range-Join*). Estas junções estão exemplificadas na Figura 2 considerando um espaço bi-dimensional euclidiano. Para representar os elementos dos dois conjuntos de dados foram utilizados pontos pretos para o conjunto R e pontos cinzas para o conjunto S . Uma junção por abrangência com raio ξ está representada na Figura 2a. Uma junção aos vizinhos mais próximos com $k = 2$ está representada na Figura 2b.

Figura 2 – Junções por similaridade

(a) Junção por Abrangência com raio ξ .

(b) Junção aos vizinhos mais próximos ($k = 2$).

2.2.3 Consultas Aproximadas por Similaridade

As técnicas aproximadas de consultas por similaridade relaxam algumas restrições das consultas por similaridade “exatas”. Estas técnicas permitem redução de custos, tais como número de acessos a discos ou número de cálculo de distâncias. No entanto, podem adicionar imprecisão aos resultados (ZEZULA *et al.*, 2006).

As técnicas aproximadas de consultas por similaridade podem ser classificadas em duas abordagens: a) Transformações sobre o espaço métrico, que modificam a representação dos objetos ou funções de distância com objetivo de reduzir custos de comparações; b) Redução do conjunto de dados a serem examinados, que omitem partes do conjunto de dados onde haja menor probabilidade de encontrar os objetos similares (FERHATOS-MANOGLU *et al.*, 2001; ZEZULA *et al.*, 2006).

Existe uma variação da *k-NNq* proposta por Indyk e Motwani (1998), que encontra uma solução aproximada para os vizinhos mais próximos ao invés de encontrar uma solução exata. Esta variação é usualmente denominada *consulta aproximada aos k-vizinhos mais próximos* – *Approximate k-Nearest Neighbors query* (ANNq). Para o vizinho mais próximo a consulta é definida como: $\epsilon\text{-NNq}(s_q) = \{s_p, \forall s'_i \in S, d(s_p, s_q) < (1 + \epsilon)d(s'_i, s_q)\}$, onde ϵ é o erro máximo admissível para a aproximação obtida. Esta definição pode ser

generalizada para $k > 1$, devendo retornar k elementos $s_1, \dots, s_k$ de tal forma que a distância de s_i para o elemento de consulta s_q seja no máximo $(1 + \varepsilon)$ vezes a distância do i -ésimo elemento mais próximo de s_q (GIONIS; INDYK; MOTWANI, 1999).

2.2.4 Métodos de Acesso Métrico

Para responder a consultas sobre dados escalares com eficiência, os SGBDR utilizam estruturas de dados e técnicas de buscas baseadas em estruturas de indexação. Tipicamente são usadas estruturas como *B-tree* e *Hashing* (ELMASRI; NAVATHE, 2011). Estas técnicas são apropriadas para responder consultas sobre dados escalares que atendem às relações de ordem e de identidade, porém elas não são apropriadas para buscas por similaridade.

Os Métodos de Acesso Métrico (MAM) são construídos sobre estruturas de indexação que consideram os dados complexos imersos em um espaço métrico. Os MAM são baseados nas distâncias entre objetos e podem ser utilizados para reduzir o número de operações de comparação, podendo assim agilizar a execução de consultas por similaridade. A maioria dos MAM usam a propriedade da desigualdade triangular para podar subconjuntos do espaço de busca (BARIONI *et al.*, 2010).

Esses índices são tipicamente estruturados em formas de árvores e funcionam particionando os conjuntos de dados em subconjuntos. O particionamento em subconjuntos pode ser feito de duas formas: particionamento por bolas e particionamento por hiperplano generalizado (SAMET, 2005). Estas formas de particionamento estão ilustradas na Figura 3 para um espaço bi-dimensional euclidiano.

Figura 3 – Possíveis particionamentos para um conjunto de dados em duas dimensões.

Fonte: Samet (2005).

Outras formas de categorização de MAM podem ser feitas de acordo com a forma de persistência da estrutura (memória ou disco) e a possibilidade de realizar atualizações na estrutura (estática ou dinâmica). Estruturas estáticas não permitem realizar operações

de atualização após a construção do índice, tais como remoção e inserção de novos dados, enquanto estruturas dinâmicas permitem operações de atualização após a construção.

Dentre as formas baseadas em memória podem ser citadas a *Generalized Hyperplane Tree* – Árvore de Hiperplano Generalizado (Gh-Tree) proposta por Uhlmann (1991) e a *Vantage Point Tree* – Árvore por ponto de vantagem (VP-Tree) proposta por Yianilos (1993). Dentre as formas baseadas em disco podem ser citadas a *Metric Tree* – Árvore Métrica (M-tree) proposta por Ciaccia, Patella e Zezula (1997), a *Slim-tree* proposta por TRAINA JR. *et al.* (2002) e a família OMNI proposta por TRAINA JR. *et al.* (2007).

O MAM Gh-Tree utiliza particionamento por hiperplano e trata-se de uma estrutura estática. A Figura 4 traz um possível particionamento de um conjunto de dados em um espaço bi-dimensional euclidiano utilizando uma Gh-Tree. A Gh-Tree particiona recursivamente os objetos utilizando dois pivôs $s_1, s_2 \in \mathbb{S}$ e divide o restante do conjunto em subconjuntos S_1 e S_2 com base no pivô mais próximo (SAMET, 2005):

$$S_1 = \{s \in \mathbb{S} | d(s_1, s) \leq d(s_2, s)\} \quad \bigwedge \quad S_2 = \{s \in \mathbb{S} | d(s_2, s) < d(s_1, s)\}$$

Figura 4 – Possível particionamento de um conjunto de dados utilizando Gh-Tree.

Fonte: Samet (2005).

A VP-Tree utiliza particionamento por bolas e também é uma estrutura estática. A Figura 5 ilustra um possível particionamento para um conjunto de dados em um espaço bi-dimensional euclidiano utilizando a VP-Tree. O método particiona os objetos recursivamente selecionando um elemento $s \in \mathbb{S}$ como pivô e calculando a mediana ξ das distâncias dos outros elementos para s . Em seguida os objetos restantes são divididos em dois subconjuntos S_1, S_2 de tamanhos iguais da seguinte forma (SAMET, 2005):

$$S_1 = \{s_1 \in \mathbb{S} | d(s, s_1) < \xi\} \quad \bigwedge \quad S_2 = \{s_2 \in \mathbb{S} | d(s, s_2) \geq \xi\}$$

Existem algumas variações da VP-Tree que podem ser utilizadas em disco como a *Multivantage Point Tree* – Árvore por múltiplos pontos de vantagem (MVP-Tree) proposta por Bozkaya e Özsoyoglu (1999).

Figura 5 – Possível particionamento de um conjunto de dados utilizando VP-Tree.

Fonte: Samet (2005).

A M-Tree foi a primeira estrutura métrica com organização dinâmica. Ela pode ser utilizada para arquivos de dados que variam dinamicamente de cardinalidade, por exemplo em casos de inserções ou remoções frequentes de objetos. A estrutura da M-Tree é construída sob uma abordagem de “baixo pra cima” (*bottom-up*) e mantém a estrutura balanceada (ZEZULA *et al.*, 2006). Existem dois tipos de nós em uma estrutura M-Tree, os nós internos e os nós folhas. Todos os objetos são armazenados em nós folhas, enquanto os nós internos mantêm ponteiros para os nós dos próximos níveis, junto com informações adicionais das subárvores (SAMET, 2005).

A Figura 6 mostra um particionamento possível para um conjunto de dados em um espaço bi-dimensional euclidiano utilizando a M-Tree. Note-se que o elemento x poderia ser inserido em qualquer uma das subárvores P_1 ou P_3 , pois encontra-se em uma região de sobreposição entre essas subárvores. A sobreposição entre as subárvores impacta diretamente no custo de execução de consultas.

Figura 6 – Possível particionamento de um conjunto de dados utilizando M-Tree.

Fonte: Adaptada de Samet (2005).

A Slim-Tree é um aperfeiçoamento da estrutura M-Tree, com o objetivo principal de reduzir a sobreposição entre as subárvores. Dentre as diversas vantagens da utilização

da Slim-Tree, pode-se citar uma redução nos tempos de inserção e divisão de nós ao mesmo tempo que reduz o espaço de armazenamento necessário (ZEZULA *et al.*, 2006). A Figura 7 ilustra um possível particionamento de um conjunto de dados em um espaço bi-dimensional euclidiano utilizando a estrutura Slim-Tree.

O algoritmo de inserção da Slim-Tree procura inserir os novos elementos em subárvores que já cubram a região que o elemento está localizado no espaço. Caso nenhuma subárvore cubra essa região, é escolhido o nó representante mais próximo. Caso mais de uma subárvore cubra essa região do espaço, é utilizada uma política de escolha de subárvore.

Quando um elemento deve ser inserido em um nó que encontra-se com a sua capacidade máxima esgotada, esse nó deve ser dividido em dois. Também é utilizada uma política para determinar como a divisão é realizada. Existem diversas políticas para a realização da divisão de nós e para a escolha dos representantes. Pode-se destacar a política MST, que constrói uma *Minimum Spanning Tree* – Árvore de Extensão Mínima (MST) sobre os elementos da subárvore. Em seguida, a maior aresta da MST é eliminada, sendo escolhido como representante de cada nó o elemento mais central de cada componente conexo (ZEZULA *et al.*, 2006).

Outra contribuição da Slim-Tree é a definição do algoritmo de pós processamento *Slim-down*. O objetivo do algoritmo é minimizar o fator de gordura (*fat-factor*) da árvore. Este fator mede a sobreposição entre as subárvores contabilizando a quantidade de elementos que poderiam estar localizados em outras subárvores. O algoritmo funciona movendo elementos entre nós irmãos que cubram a mesma região do espaço, caso isso reduza o raio de cobertura de uma das subárvores (SAMET, 2005).

Figura 7 – Possível particionamento de um conjunto de dados utilizando Slim-Tree.

2.2.5 Hashing por Similaridade

O termo *Locality-Sensitive Hashing* – Particionamento sensível a localidade (LSH), definido por Indyk e Motwani (1998) compreende um arcabouço de funções de *hashing*. O conceito de LSH pode ser aplicado para responder a consultas aproximadas por similaridade em tempo sublinear. Funções LSH diferem das funções de *hashing* tradicionais pois

estas têm por objetivo maximizar a probabilidade de colisão para objetos similares (LESKOVEC; RAJARAMAN; ULLMAN, 2014).

Uma família de funções LSH $\mathcal{F}$ é definida sobre um espaço Métrico $\mathbb{M} = \langle \mathbb{S}, d \rangle$. A família $\mathcal{F}$ é uma família de funções $f : \mathbb{M} \rightarrow \mathbb{S}$ que mapeia os elementos de um espaço métrico para algum *bucket* $s \in \mathbb{S}$, onde $\mathbb{S}$ é um conjunto de elementos do domínio $\mathbb{M}$. Sejam dois valores de distância $d_1 < d_2$ de acordo com alguma função de distância d , uma família $\mathcal{F}$ é dita (d_1, d_2, p_1, p_2) -sensitiva se satisfazer as seguintes condições para toda função $f \in \mathcal{F}$ (LESKOVEC; RAJARAMAN; ULLMAN, 2014):

- Se $d(s_1, s_2) \leq d_1$, então a probabilidade de $f(s_1) = f(s_2)$ é maior que p_1 .
- Se $d(s_1, s_2) \geq d_2$, então a probabilidade de $f(s_1) = f(s_2)$ é menor que p_2 .

Onde $f(s_1) = f(s_2)$ representa que houve colisão entre o par s_1 e s_2 , ou seja que os objetos s_1 e s_2 foram mapeados para o mesmo *bucket*. A Figura 8 ilustra os valores esperados de uma função (d_1, d_2, p_1, p_2) -sensitiva. Não é possível afirmar o que acontece com valores de distância entre d_1 e d_2 , porém é possível fazer os valores tão próximos quanto desejável com a penalidade de tipicamente fazer os valores p_1 e p_2 muito próximos também. É possível distanciar os valores de p_1 e p_2 enquanto se mantem os valores de d_1 e d_2 fixos (LESKOVEC; RAJARAMAN; ULLMAN, 2014).

Figura 8 – Comportamento de uma função (d_1, d_2, p_1, p_2) -sensitiva.

Fonte: Adaptada de Leskovec, Rajaraman e Ullman (2014).

Existem diferentes tipos de famílias LSH para diferentes tipos de funções de distância, incluindo as funções de distância *Minkowski*, distância de cosseno, distância de *Jaccard* e outras (WANG *et al.*, 2014). Uma família LSH baseada na distribuição p-estável foi projetada por Datar *et al.* (2004) e pode ser utilizada para aproximar as distâncias de Manhattan e Euclidiana. Uma família LSH que utiliza projeção aleatória foi desenvolvida por Charikar (2002) para aproximar a distância de cosseno entre dados dimensionais.

2.2.6 Suporte à Similaridade em SGBDR

Para tratar dados complexos de maneira eficiente, uma abordagem promissora é a inclusão do suporte à similaridade dentro dos SGBDR (BARIONI *et al.*, 2010).

Existem diversos trabalhos na literatura que implementam operadores por similaridade em SGBDR. “PostgreSQL with Image-handling Extension (PostgreSQL-IE)” é uma extensão ao SGBDR PostgreSQL que inclui operações por similaridade para imagens (GULIATO *et al.*, 2009). O “similarity-aware database management system (SimDB)” adiciona novas palavras chaves e operações por similaridade modificando o SGBDR PostgreSQL (SILVA *et al.*, 2010).

O “SImlarity Retrieval ENgine (SIREN)” usa uma versão estendida da linguagem SQL. Para isto é adicionada uma camada entre a aplicação e o SGBDR. Primeiro, os predicados envolvendo similaridade são identificados e executados usando um MAM para indexar o conjunto de dados. Em seguida os predicados escalares são enviados ao SGBDR (BARIONI *et al.*, 2006). O arcabouço “Similarity by Authorities (SimbA)” estende o SIREN e permite a inclusão de extratores de características, funções de distância e MAM pelo usuário (BEDO; TRAINA; TRAINA JR., 2014).

O “Features, Metrics and Indexes for Similarity Retrieval (FMI-SiR)” é um módulo de similaridade para o SGBDR Oracle que usa funções definidas pelo usuário para extração de características e indexação (KASTER *et al.*, 2010).

Também existem alguns trabalhos que focam no desenvolvimento de linguagens de consultas para suporte à similaridade (BARIONI *et al.*, 2005; BUDÍKOVÁ; BATKO; ZEZULA, 2012).

2.3 Clustering

Cluster analysis ou *clustering* são técnicas que consistem em encontrar agrupamentos nos dados. Essas técnicas são bastante utilizadas em diversos domínios, como inteligência artificial, reconhecimento de padrões, química, economia, marketing, medicina, política, psicologia, biologia, etc. (KAUFMAN; ROUSSEEUW, 2009).

Em *clustering*, os grupos são formados de maneira que os objetos do mesmo grupo possuam alta similaridade entre eles e ao mesmo tempo sejam dissimilares com objetos de outros grupos. Cada grupo pode ser visto com uma classe de objetos. Existem diversas técnicas de *clustering* e usualmente elas são organizadas em categorias: *métodos particionais*, *métodos hierárquicos*, *métodos baseados em densidade* e *métodos baseados em grade* (HAN; KAMBER; PEI, 2011).

Métodos particionais realizam o particionamento em um único nível, dividindo um conjunto de n objetos em k partições, onde cada partição representa um grupo, sendo

$k \leq n$. Estes métodos criam um particionamento inicial e iterativamente vão realocando os objetos, movendo objetos de um grupo a outro utilizando alguns critérios (CORDEIRO; FALOUTSOS; TRAINA JR., 2013). Exemplos de algoritmos particionais são o *k-means* (LLOYD, 1982) e o *CLARANS* (NG; HAN, 1994).

Métodos hierárquicos criam hierarquias de objetos e podem ser divisivos ou aglomerativos. Métodos aglomerativos começam com cada objeto sendo um grupo separado e iterativamente juntam grupos semelhantes até formar um único grupo, que será o primeiro nível da hierarquia. Os métodos divisivos começam com um único grupo, e então os grupos são iterativamente divididos em grupos menores até que cada grupo contenha somente um objeto (CORDEIRO; FALOUTSOS; TRAINA JR., 2013). Os algoritmos *AGNES* e *DIANA* definidos por Kaufman e Rousseeuw (2009), são exemplos de métodos hierárquicos aglomerativo e divisivo, respectivamente.

Os métodos baseados em densidade formam os grupos baseados na densidade (número de objetos em uma região). Um grupo é expandido até a densidade atingir um determinado limiar. Este tipo de método pode ser usado para eliminar ruídos no conjunto de dados e encontrar grupos de formato arbitrário, uma vez que os métodos particionais são comumente limitados a buscar apenas grupos em formato esférico (HAN; KAMBER; PEI, 2011). Os algoritmos *DBSCAN* (ESTER *et al.*, 1996) e *OPTICS* (ANKERST *et al.*, 1999) são exemplos de algoritmos baseados em densidade.

Métodos baseados em grade dividem o espaço em um número finito de células, de forma a criar uma estrutura em grade. A principal vantagem desta abordagem é que o processo tipicamente depende somente do número de células do espaço e independe do número de objetos. Estes métodos podem ser combinados com métodos baseados em densidade e métodos hierárquicos (HAN; KAMBER; PEI, 2011). Os algoritmos *STING* (WANG; YANG; MUNTZ, 1997) e *CLIQUE* (AGRAWAL *et al.*, 1998) são exemplos de algoritmos baseados em grade.

Na maioria dos casos, o processo de *clustering* não é aplicado dentro dos SGBDR, embora muitas aplicações pudessem se beneficiar da execução destas técnicas aplicadas dentro de um SGBDR, por exemplo, aproveitando índices presentes nos dados. Além disto, outras operações presentes em SGBDR poderiam ser aplicadas sobre os grupos.

2.3.1 Métodos de Avaliação de Clusters

A avaliação de um método de *clustering* é realizada avaliando o seu resultado. Os métodos para avaliação de qualidade de agrupamentos podem ser divididos em duas abordagens, de acordo com a disponibilidade dos particionamentos ideais. Quando os particionamentos ideais estão disponíveis, pode ser utilizado um método *extrínseco* que compara os particionamentos obtidos com os particionamentos ideais. Caso contrário deve

ser utilizado um método *intrínseco* que avalia a qualidade das partições obtidas considerando a separação dos grupos (HAN; KAMBER; PEI, 2011).

O método extrínseco de *Mutual Information* – Informação Mútua (MI) mede a quantidade de informação entre duas variáveis. O valor MI para dois particionamentos X e Y é dado pela equação:

$$MI(X, Y) = \sum_{y \in Y} \sum_{x \in X} p(x, y) \log \left(\frac{p(x, y)}{p(x) p(y)} \right)$$

Onde $p(x, y)$ é a probabilidade de distribuição conjunta e $p(x)$ e $p(y)$ são probabilidades de distribuição marginais (VINH; EPPS; BAILEY, 2010).

O método de *Adjusted Mutual Information* – Informação Mútua Ajustada (AMI) é a versão ajustada ao acaso do método MI. Utilizando o método AMI, uma pontuação igual a 1 significa que os particionamentos são idênticos, enquanto um resultado próximo de 0 significa uma intersecção nula entre os particionamentos. Para dois particionamentos X e Y o AMI é dado pela seguinte equação:

$$AMI(X, Y) = \frac{MI(X, Y) - E(MI(X, Y))}{\max(H(X), H(Y)) - E(MI(X, Y))}$$

Onde $E(MI(X, Y))$ denota o valor esperado para o MI adotando um modelo hipergeométrico de aleatoriedade, enquanto $H(X)$ e $H(Y)$ são os valores de entropia associada para os particionamentos X e Y , respectivamente (VINH; EPPS; BAILEY, 2010).

2.4 Estruturas de Dados

Neste trabalho foram propostos alguns algoritmos que podem ser implementados de forma mais simplificada e eficiente usando algumas estruturas de dados existentes na literatura como *union-find*, *multiset* e *map*.

A estrutura de dados *union-find*, também chamada de *conjuntos disjuntos* ou *conjunto merge-find* pode manter subconjuntos não sobrepostos de elementos. Esta estrutura auxilia a execução das seguintes operações: *makeSet* que cria um conjunto de cardinalidade n e contém apenas elementos isolados, *union* que mescla dois subconjuntos em um mesmo subconjunto e a operação *find* que retorna qual subconjunto contém um determinado elemento. Existem diversos algoritmos para implementação da estrutura de dados *union-find*: *quick-find*, *quick-union*, *quick-union* ponderado e *quick-union* ponderado com compressão de caminho. O Quadro 1 traz um comparativo entre o tempo de execução (no pior caso) das diversas implementações (SEdgeWICK; WAYNE, 2011).

Multiset é uma generalização do conceito matemático de conjuntos para permitir valores duplicados (SEdgeWICK; WAYNE, 2011). A estrutura *multiset* é incluída na

Quadro 1 – Comparação do tempo de execução das implementações da estrutura *union-find*

Algoritmo	<i>MakeSet</i>	<i>union</i>	<i>find</i>
<i>quick-find</i>	N	N	1
<i>quick-union</i>	N	N (inclui busca)	N
<i>quick-find</i> ponderado	N	$\log N$	$\log N$
<i>quick-find</i> ponderado com compressão de caminho	N	1 (amortizado)	1 (amortizado)

Standard Template Library (STL), que permite a execução de diversas operações, como *insert* para inserção de elementos e *count* para retornar o número de ocorrências de um determinado elemento no multiset (ISO/IEC, 2014).

Uma estrutura de dados do tipo *map* associa uma chave a um valor. Essa estrutura também é chamada de *arranjo associativo*, *tabela de símbolos* ou *dicionário* (SEDIGEWICK; WAYNE, 2011). A estrutura *map* é incluída na STL da linguagem C++ e suporta diversas operações como *insert* para inserção de elementos e *find* que encontra elemento com chave específica (ISO/IEC, 2014).

2.5 Considerações

Neste capítulo foram descritos alguns conceitos relacionados ao modelo relacional e a terminologia utilizada, como relações, atributos, tuplas e atributos nulo. Foram apresentados também os operadores fundamentais da Álgebra Relacional. Também foram apresentadas as principais funções de agregação, o operador agrupamento e a estrutura sintática de uma consulta com a cláusula `GROUP BY` na linguagem SQL.

Além disto, também foram apresentadas as principais consultas por similaridade, a definição de espaços métricos, métodos de acesso e como a inclusão de consultas por similaridade tem sido implementada de maneira prática em SGBDR.

Por fim, foram detalhadas algumas diferentes técnicas de *clustering* que são organizadas em métodos particionais, métodos hierárquicos, métodos baseados em densidade e métodos baseados em grade.

TRABALHOS RELACIONADOS

Neste capítulo são detalhados alguns trabalhos que tratam de agrupamento por similaridade em SGBDR e alguns outros operadores que são úteis para este trabalho.

Na [Seção 3.1](#) são apresentados os operadores que incorporem de alguma forma o conceito de agrupamentos por similaridade no contexto de SGBDR. Já a [Seção 3.2](#) aborda outros operadores úteis para este trabalho. Por fim, na [Seção 3.3](#) é apresentada uma sumarização do capítulo contendo um comparativo dos métodos apresentados.

3.1 Agrupamento por Similaridade em SGBDR

A [Subseção 3.1.1](#) apresenta um conceito de operadores de agrupamentos definidos pelo usuário. Em seguida, a [Subseção 3.1.2](#) apresenta uma generalização do operador de agrupamento, aplicado em conjunto com filtros e ranqueamento. Na [Subseção 3.1.3](#) é apresentado o operador *Cluster By* para atributos espaciais. A [Subseção 3.1.4](#) apresenta o operador *Similarity Group-By* e as suas variações semânticas. Uma extensão multidimensional ao operador *Similarity Group-By* é apresentada na [Subseção 3.1.5](#).

3.1.1 Agrupamentos Definidos Pelo Usuário

[Schallehn, Sattler e Saake \(2001\)](#) desenvolveu uma extensão para a linguagem SQL que permite a incorporação dos conceitos de detecção e eliminação de duplicatas utilizando similaridade. O trabalho provê interfaces genéricas para que a definição de agrupamentos e agregações possa ser feita pelo usuário, permitindo o uso de funções mais complexas. Além disto, o trabalho também define duas estratégias de agrupamento: a *transitive similarity* e a *strict similarity*.

As definições de agregação e agrupamento podem ser estendidas usando uma interface definida para um determinado contexto. Usando a interface, o usuário pode imple-

mentar uma função de agrupamento. A função deve ser definida para receber um atributo e pode receber também outros parâmetros. O [Código-fonte 3](#) traz um exemplo da sintaxe definida, a qual adiciona a cláusula `CONTEXT` seguida de uma função definida pelo usuário para realização do agrupamento, junto com os parâmetros.

Código-fonte 3 – Sintaxe do `GROUP BY` estendido com agrupamento definido pelo usuário.

```

1:      SELECT ...
2:      FROM ...
3:      GROUP BY CONTEXT function(atributo [, ...] );

```

A sintaxe para a extensão utilizando as estratégias *transitive* e *strict* está detalhada no [Código-fonte 4](#). A estratégia *transitive* que considera a similaridade transitiva para todo o grupo, enquanto a *strict* necessita de comparações entre todas as tuplas de um grupo, dividindo-o em caso de conflito. O critério de similaridade é especificado por uma cláusula `ON` e as expressões podem utilizar funções de distância definidas pelo usuário serem combinadas utilizando cláusulas `AND` e `OR`. Para a combinação de expressões é utilizada uma regra de combinação `MIN/MAX` onde a similaridade de expressões combinadas utilizando `AND` é definida como o mínimo entre os valores enquanto o valor de similaridade para expressões que utilizam `OR` é definido como o máximo entre os valores. Quando o atributo aparece sem a utilização de uma função definida pelo usuário, é considerada a igualdade entre atributos ao invés da similaridade.

Código-fonte 4 – Sintaxe do `GROUP BY` estendido com funções de distância definidas pelo usuário.

```

1:      SELECT ...
2:      FROM ...
3:      GROUP BY < TRANSITIVE | STRICT > SIMILARITY ON
4:             < function(atributo) | atributo >
5:             [ AND | OR < function(atributo) | atributo > ]
6:             THRESHOLD  $\varepsilon$ ;

```

3.1.2 Operador *ClusterRank*

ClusterRank é uma generalização do operador `GROUP BY` como um operador *fuzzy* integrando o operador com *ranking* e filtros booleanos definida por [Li et al. \(2007\)](#). Uma consulta *ClusterRank* é executada em três etapas: *filtering*, *clustering* e *ranking*. Na etapa de *filtering* são aplicados filtros booleanos sobre a relação base ou sobre o produto cartesiano das relações-bases. Em seguida, as tuplas resultantes da etapa de *filtering* são partici-

onadas em uma quantidade t de *clusters* baseando-se nos atributos de agrupamento. Por fim, uma função é aplicada sobre um conjunto de atributos de ranqueamento, retornando um valor para cada tupla, onde para cada grupo, um número k de tuplas com as maiores pontuações são retornadas.

O [Código-fonte 5](#) mostra um exemplo da sintaxe da SQL estendida proposta para este operador, onde a cláusula `CLUSTER BY` define os atributos c_i a serem utilizados para o particionamento em t grupos que são definidos após a palavra reservada `INTO`. Os atributos r_i juntamente com respectivos pesos w_i são utilizados para ranqueamento na cláusula `ORDER BY`. Em seguida, para cada grupo são obtidas as k tuplas com os maiores (ou menores) valores, pela cláusula `LIMIT`. Note-se que esta cláusula tem sua semântica alterada, mantendo assim k tuplas (ou menos) por grupo, ao invés de k tuplas (ou menos) no resultado global, diferindo assim do padrão empregado pelos SGBD.

Código-fonte 5 – Sintaxe do operador *ClusterRank*.

```

1: SELECT ...
2: FROM ...
3: WHERE ...
4: CLUSTER BY  $c_1, \dots, c_m$  INTO  $t$ 
5: ORDER BY [ASC | DESC]  $w_1 \times r_1 + \dots + w_n \times r_n$  LIMIT  $k$ ;

```

3.1.3 Operador Cluster By

Junto ao operador *cluster by* definido por [Zhang e Huang \(2007\)](#), foi proposta uma extensão para a linguagem SQL para ser utilizada em bases de dados espaciais. O operador *cluster by* permite criar agrupamentos espaciais a partir de pontos de dados isolados e obter informações através de agregações.

A sintaxe do operador está descrita no [Código-fonte 6](#). Na cláusula `SELECT` deve ser especificada uma função de agregação de grupo sobre um atributo espacial e podem ser especificadas outras funções de agregação sobre atributos não espaciais. Na cláusula `CLUSTER BY` é fornecido um atributo espacial para ser usado no agrupamento e que pode ser acompanhado da cláusula `ALGORITHM`.

Código-fonte 6 – Sintaxe do operador Cluster-by

```

1: SELECT
2:     cluster_aggregate_function(spatial_attribute)
3:     [, aggregate_function(non_spatial_attribute(s))]
4: FROM ...
5: [WHERE ...]

```

```

6:      CLUSTER BY spatial_attribute
7:      [ALGORITHM algorithm_name(parâmetros)];

```

A função de agregação de grupo aceita atributos espaciais como parâmetro. Ela agrega cada grupo em um objeto espacial como um polígono ou uma linha. A cláusula `ALGORITHM` permite que se indique um algoritmo realizar o particionamento, junto com os respectivos parâmetros. Esta cláusula é opcional, pois os SGBD podem assumir um algoritmo padrão.

O [Código-fonte 7](#) exemplifica uma consulta utilizando o operador *cluster by* e a agregação `ST_Polygon`, que transforma um conjunto de pontos em um polígono, a função `ST_Area` é aplicada sobre este polígono para retornar sua área. Além dessas duas funções de agregação, também foram propostas as funções de agregação `ST_Length`, `ST_Boundary`, `ST_Union`, `ST_Difference`, `ST_Centroid`.

Código-fonte 7 – Exemplo de consulta usando o operador Cluster-by.

```

1:      SELECT ST_Area(ST_Polygon(House.location))
2:      FROM House
3:      WHERE House.household_income < 30K
4:      CLUSTER BY House.location

```

O resultado para a consulta expressa no [Código-fonte 7](#) pode ser visualizado na [Figura 9](#). A [Figura 9a](#) apresenta o conjunto de dados armazenado. Estes dados são filtrados pela condição `WHERE` na [Figura 9b](#). Em seguida é aplicado o processo de *clustering*, ilustrado na [Figura 9c](#). Por fim, os dados agregados são mostrados na [Figura 9d](#).

3.1.4 Similarity Group-By

Um operador de agrupamento denominado *Similarity Group-By* (SGB) foi proposto por [Silva, Aref e Ali \(2009\)](#). Trata-se de uma extensão ao operador `GROUP BY` da linguagem SQL, que considera a similaridade para realizar particionamento em domínios de dados unidimensionais. O operador SGB tem o propósito de agrupar objetos similares ou com valores aproximados ao invés de gerar grupos baseados em igualdade.

A definição formal do operador proposto por [Silva, Aref e Ali \(2009\)](#) é a seguinte:

$$(G_1, S_1), \dots, (G_n, S_n) \gamma_L (T)$$

onde T é a relação a ser agrupada, G_i é o atributo de agregação por similaridade, S_i é uma segmentação do domínio de G_i , em segmentos sem sobreposição e L é a lista de funções de agregação a serem aplicadas sobre os atributos de T .

Figura 9 – Exemplo de consulta usando o operador Cluster-by.

Fonte: [Zhang e Huang \(2007\)](#).

Este operador apresenta três abordagens distintas para a formação dos grupos. A primeira é baseada nas propriedades de separação máxima entre os elementos e o diâmetro máximo do grupo. Essa abordagem é denominada *Unsupervised Similarity Group-by* (SGB-U). Outra abordagem é baseada em pontos de interesse, denominada *Supervised Similarity Group Around* (SGB-A). A terceira abordagem é denominada *Supervised SGB using Delimiters* (SGB-D) e é baseada em um conjunto de pontos delimitadores.

A abordagem SGB-U agrupa o conjunto de tuplas de forma não supervisionada, usando dois parâmetros de controle. A sua sintaxe é definida no [Código-fonte 8](#). O parâmetro s da cláusula `MAXIMUM_ELEMENT_SEPARATION` determina que dois elementos estarão em grupos distintos caso a distância entre eles seja maior que s . O parâmetro d presente na cláusula `MAXIMUM_GROUP_DIAMETER` determina que, para cada grupo formado, a distância entre os extremos deve ser menor ou igual a d . Caso nenhum dos parâmetros seja definido, o operador deve se comportar como o `GROUP BY` padrão.

Código-fonte 8 – Sintaxe do SGB-U

```

1:     SELECT atributo, função-de-agregação(atributo)
2:     FROM tabela
3:     GROUP BY atributo

```

```

4:      [MAXIMUM_ELEMENT_SEPARATION s]
5:      [MAXIMUM_GROUP_DIAMETER d];

```

A abordagem SGB-A agrupa o conjunto de tuplas de forma supervisionada, baseado em um conjunto de pontos de interesse, chamados de pontos centrais. A sintaxe desta forma de agrupamento pode ser vista em [Código-fonte 9](#). A cláusula `AROUND` define um conjunto de pontos de interesse que serão os centros dos grupos. As cláusulas do SGB-U também podem ser utilizadas.

Código-fonte 9 – Sintaxe do SGB-A

```

1:      SELECT ...
2:      FROM ...
3:      GROUP BY atributo AROUND pontos

```

A abordagem SGB-D agrupa o conjunto de tuplas de forma supervisionada, usando um conjunto de pontos limites. A sintaxe esta definida no [Código-fonte 10](#). A cláusula `DELIMITED BY` é utilizada para determinar o conjunto dos pontos limitantes, recebendo um conjunto de pontos ou o resultado de uma subconsulta.

Código-fonte 10 – Sintaxe do SGB-D

```

1:      SELECT ...
2:      FROM ...
3:      GROUP BY atributo DELIMITED BY [pontos | subconsulta];

```

Alguns exemplos de consulta com estes operadores podem ser vistos na [Figura 10](#). As três formas estão aplicadas sobre o mesmo conjunto de dados e resultam em agrupamentos distintos.

3.1.5 *Similarity Group-By Multidimensional*

O operador SGB é focado em dados unidimensionais. Para atributos multidimensionais, cada dimensão é tratada como um atributo independente. Para tratar dados multidimensionais, o operador SGB foi estendido por [Tang et al. \(2016\)](#). Para isto, os predicados *Similarity Group-By Any* (SGB-Any) e *Similarity Group-By All* (SGB-All) foram definidos para tratar duas formas de agrupamento distintas.

O predicado SGB-Any determina que cada elemento inserido em um grupo deve ter uma distância menor ou igual a um limiar ε para *pelo menos um* outro elemento contido no mesmo grupo. A sintaxe para este predicado esta definida no [Código-fonte 11](#).

Figura 10 – Exemplo de diferentes consultas com *similarity group by*.

(a) Agrupamento com SGB-U.

(b) Agrupamento com SGB-A.

(c) Agrupamento com SGB-D.

Fonte: Adaptada de [Silva, Aref e Ali \(2009\)](#).

Código-fonte 11 – Sintaxe do SGB-Any

```

1:    SELECT ...
2:    FROM ...
3:    GROUP BY atributo
4:    DISTANCE-TO-ANY função-de-distância WITHIN  $\varepsilon$ ;

```

O método SGB-All determina que, para todos os elementos inseridos em um grupo, a distância para qualquer outro elemento nesse mesmo grupo deve ser menor ou igual a um limiar ε definido. O [Código-fonte 12](#) exemplifica a sintaxe para este predicado.

Como pode haver elementos com sobreposição de grupos com a forma de agrupamento SGB-All, a cláusula `ON-OVERLAP` foi definida para decidir em qual grupo cada elemento deve ser inserido. Esta cláusula informa qual ação deve ser tomada em caso de uma sobreposição de grupos.

Os valores possíveis para a cláusula `ON-OVERLAP` são `JOIN-ANY`, `ELIMINATE` e `FORM-NEW-GROUP`. O valor `JOIN-ANY` determina que um dado objeto em sobreposição seja inserido aleatoriamente em algum dos grupos que estão sobrepostos. O valor `ELIMINATE` diz para descartar todos os pontos com sobreposição de grupos. O valor `FORM-NEW-GROUP` determina que se crie um novo grupo contendo os pontos com sobreposição.

Código-fonte 12 – Sintaxe do SGB-All

```

1:  SELECT ...
2:  FROM ...
3:  GROUP BY atributo
4:      DISTANCE-TO-ALL função-de-distancia WITHIN  $\epsilon$ 
5:      ON-OVERLAP [JOIN-ANY | ELIMINATE | FORM-NEW-GROUP];

```

A semântica destas formas de agrupamento pode ser vista na Figura 11 para um conjunto de dados bi-dimensionais em espaço euclidiano utilizando o limiar $\epsilon = 3$. Na Figura 11a, todos os pontos ($a-h$) formam um único grupo. Na Figura 11b, o ponto c forma um *clique* com dois grupos distintos: a, b, d, e e f, g .

Figura 11 – Semântica dos predicados SGB com $\epsilon = 3$.

Fonte: Tang *et al.* (2016).

3.2 Outros Operadores

Existem diversas propostas de extensão ao operador de agrupamento relacional. A Subseção 3.2.1 apresenta uma extensão ao operador GROUP BY denominada CUBE, que atualmente já é incorporado em diversos SGBDR. Na Subseção 3.2.2 são apresentadas as funções de janelamento. Na Subseção 3.2.3 são apresentados os conjuntos por similaridade denominados (simsets) e as suas propriedades. Na Subseção 3.2.4 é apresentado o operador *skyline*, usado para filtrar pontos de interesse em conjuntos de dados.

3.2.1 Operador CUBE

O operador CUBE generaliza diversas construções encontradas em relatórios, podendo gerar de maneira eficiente e em uma única consulta o resultado de diversos construtores como histogramas, tabulação cruzada, *roll-up*, *drill-down* e sub-total (GRAY *et al.*, 1996). Este operador foi padronizado pela *International Organization for Standardization*

(ISO) a partir da versão SQL:1999, sendo implementado em diversos SGBDR comerciais e de código aberto.

Código-fonte 13 – Exemplo de consulta usando o operador CUBE.

```
1:      SELECT departamento, cargo, SUM(salário)
2:      FROM funcionários
3:      GROUP BY CUBE(departamento, cargo);
```

Uma consulta com o operador CUBE retorna um agrupamento para cada uma das combinações possíveis de atributos listados e uma agregação geral dos agrupamentos. O [Código-fonte 13](#) traz um exemplo de consulta utilizando o operador que aparece na cláusula GROUP BY, definido pela sub-cláusula CUBE junto com os atributos. O resultado para essa consulta é mostrado na [Tabela 5](#). Algumas variações do comando CUBE, tais como ROLLUP e GROUPING SETS permitem obter apenas algumas das combinações geradas por um comando CUBE.

Tabela 5 – Resultado de operador CUBE sobre atributos da relação “funcionários”.

departamento	cargo	sum(salário)
RH	Diretor(a)	3500,00
RH		3500,00
Gerência	Gerente Vendas	5000,00
Gerência	Gerente Geral	7000,00
Gerência		12000,00
Vendas	Vendedor(a)	12200,00
Vendas		12200,00
Marketing	Diretor(a)	3500,00
Marketing	Auxiliar	2100,00
Marketing		5600,00
	Gerente Geral	7000,00
	Vendedor(a)	12200,00
	Gerente Vendas	5000,00
	Diretor(a)	7000,00
	Auxiliar	2100,00
		33300,00

3.2.2 Funções de Janelamento

Assim como o operador de agrupamento, as funções de janelamento (*window functions*) também são executadas sobre um conjunto de tuplas relacionadas. Porém, ao invés de agrupar os resultados em um único valor agregado, os valores originais das tuplas também são mantidos.

Em SQL, a definição de uma janela de execução para uma função de janelamento é definida pela cláusula `OVER()`. Esta cláusula pode receber os predicados `PARTITION BY`: que define o particionamento das tuplas, e `ORDER BY`: que define a ordem das tuplas dentro do particionamento. As funções de janelamento padronizado pela ISO a partir da versão SQL:2003.

Uma consulta pode incluir múltiplas funções de janelamento, que podem ser definidas sobre a mesma janela ou sobre janelas diferentes. No [Código-fonte 14](#) temos um exemplo de consulta que utiliza funções de janelamento. O resultado desta consulta é apresentado na [Tabela 6](#).

Código-fonte 14 – Exemplo de funções de janelamento em SQL

```

1: SELECT
2:   nome ,
3:   departamento ,
4:   salario ,
5:   AVG(salário) OVER(PARTITION BY departamento),
6:   RANK() OVER(PARTITION BY departamento ORDER BY salário DESC),
7:   RANK() OVER(ORDER BY salário DESC)
8: FROM funcionários;
  
```

Tabela 6 – Exemplo de resultado funções de janelamento.

nome	departamento	salário	avg	rank	rank
João	Gerência	7000,00	6000,00	1	1
José	Gerência	5000,00	6000,00	2	2
Marcia	RH	3500,00	3500,00	1	3
Júlia	Marketing	3500,00	2800,00	1	3
Luiza	Vendas	3000,00	2440,00	1	5
Mário	Vendas	2500,00	2440,00	2	6
Antônio	Vendas	2400,00	2440,00	3	7
Edgar	Vendas	2300,00	2440,00	4	8
Milton	Marketing	2100,00	2800,00	2	9
Flávia	Vendas	2000,00	2440,00	5	10

3.2.3 Simsets

[Pola et al. \(2013\)](#) estendeu o conceito de conjuntos para conjuntos por similaridade ou *similarity sets* (Simsets). Para isto, algumas propriedades foram definidas, onde o operador $\triangleq^{\xi}$ (suficientemente similar) substitui o operador de comparação por igualdade. Dados dois objetos $s_1, s_2 \in \mathbb{S}$ é dito que eles são suficientemente similares e repre-

sentado por $s_1 \hat{=}^\xi s_2$, se $d(s_1, s_2) \leq \xi$. Um ξ -simset é um conjunto onde não existe nenhum par de elementos $s_1, s_2 \in \mathbb{S}$ tal que $s_1 \hat{=}^\xi s_2$.

Para extrair um ξ -simset de um conjunto é utilizado um grafo de similaridade $\mathcal{G}^\xi$. Cada nó v_i deste grafo corresponde a um elemento $s_i \in \mathbb{S}$, e uma aresta é criada para cada par de elementos s_i, s_j , tal que $s_i \hat{=}^\xi s_j$.

Um *simset* pode ser extraído usando políticas diferentes. Foram propostas duas políticas consideradas mais importantes: MIN e MAX. A política MIN extrai um ξ -simset com a menor cardinalidade possível, enquanto a política MAX extrai o ξ -simset com a maior cardinalidade possível.

A Figura 12 ilustra exemplos de 2-simsets produzidos utilizando a distância euclidiana para um conjunto de dados de exemplo. Os dados de um exemplo estão representados na Figura 12a e a representação do grafo de similaridade na Figura 12b. A Figura 12c e a Figura 12d representam os 2-simset extraídos com as políticas MIN e MAX, respectivamente.

Figura 12 – Exemplos de *simsets* produzidos.

Fonte: Pola *et al.* (2015).

3.2.4 Operador de Skyline

Outro operador interessante é o *skyline*, definido por Börzsönyi, Kossmann e Stocker (2001). Este operador é usado quando se deseja maximizar ou minimizar objetivos distintos. Por exemplo, considere que uma pessoa que deseja passar férias na praia e deseja se hospedar em um hotel próximo ao mar e que ao mesmo tempo seja barato. A tendência é que quanto mais próximo à praia, mais caro o valor do hotel. O operador de *skyline* pode ser utilizado para apresentar todos os hotéis interessantes, ou seja, os hotéis que não são piores que qualquer outro hotel em ambos os atributos.

O operador *skyline* é útil para filtrar um conjunto de pontos de interesse de grandes conjuntos de dados. Para um ponto ser considerado como de interesse ele não deve ser dominado por outro ponto. Considerando os objetivos que se deseja maximizar ou minimizar em uma determinada dimensão, pode se dizer que um ponto domina um outro ponto se ele: a) for tão bom quanto ou melhor que o outro ponto em todas as dimensões; b) melhor que o outro ponto em ao menos uma dimensão.

Este operador pode ser usado para visualização de conjuntos de dados e leva este nome por causa de sua representação gráfica, que lembra uma linha do horizonte. Essa comparação pode ser vista na Figura 13. A Figura 13a ilustra a representação gráfica do *skyline*, para o exemplo anterior, os pontos que se encontram na linha representam hotéis interessantes, enquanto os pontos acima desta linha representam hotéis que não são interessantes. Para efeito de ilustração, a Figura 13b mostra o *skyline* para a cidade do Rio de Janeiro.

Figura 13 – Representação gráfica de *skyline*.

(a) Exemplo de skyline

(b) Skyline do Rio de Janeiro

Fonte <<http://goo.gl/HPckfd>>

Para a inclusão do operador de *skyline* em SQL foi proposta uma extensão por Börzsönyi, Kossmann e Stocker (2001). A sintaxe pode ser vista no Código-fonte 15. A cláusula SKYLINE OF deve estar entre as cláusulas HAVING e ORDER BY caso elas sejam utilizadas. Como o *skyline* pode ser executado após a cláusula GROUP BY, o *skyline* pode ser calculado sobre os dados agrupados e agregados. A cláusula SKYLINE OF irá receber os atributos $d_1 \dots d_m$, a serem minimizados, maximizados ou diferentes. Caso haja mais de uma tupla com o mesmo valor em todos atributos na lista do SKYLINE OF, e estas tuplas não sejam dominadas por nenhuma outra, todas elas farão parte do *skyline*. Caso seja desejável retornar apenas uma tupla destas de mesmo valor presentes no *skyline*, a cláusula DISTINCT pode ser utilizada.

Código-fonte 15 – Sintaxe do operador *skyline*

```

1:    SELECT ...
2:    FROM ...
3:    [GROUP BY ...
4:        [HAVING ...] ]
5:    SKYLINE OF
6:        [DISTINCT]  $d_1$  [MIN | MAX | DIFF],
7:        ...,  $d_m$  [MIN | MAX | DIFF]
8:    [ORDER BY ...];

```

O operador de *skyline* auxilia nos processos de decisão baseados em objetos individuais ou em agrupamentos. Algumas extensões ao operador de *skyline* foram propostas para tratar de agrupamentos baseados em *skyline*, tal como a extensão proposta por Zhang e Chomicki (2011) para tratar de decisões baseadas em grupo utilizando preferências, que são representadas como um conjunto de restrições adicionais. Outra extensão proposta por Zhang *et al.* (2014) procura por grupos *skyline* em um conjunto de dados, onde o *skyline* é calculado sobre a agregação dos atributos de cada grupo. Cada grupo *skyline* possui k tuplas e não pode ser dominado por nenhum outro grupo de k tuplas.

3.3 Considerações

Neste capítulo foram detalhadas algumas abordagens de agrupamento por similaridade em SGBDR, além de alguns operadores e conceitos que estão relacionados com agrupamento ou relacionados com o conceito de similaridade.

Foi apresentada uma extensão que permite a inclusão de funções de agrupamentos e funções de agregações pelo usuário onde, utilizando uma interface definida, o usuário pode implementar funções de distância para utilizar duas estratégias previamente definidas utilizando as cláusulas TRANSITIVE SIMILARITY e STRICT SIMILARITY juntamente

com um limiar de distância, ou definir uma nova forma de agrupamento e utiliza-la pela cláusula `CONTEXT`.

Em seguida, foi apresentado o operador *ClusterRank* que trata-se de uma extensão que generaliza o operador de agrupamento de uma maneira integrada com filtragem e ranqueamento por grupo. O operador *Cluster By*, permite a criação agrupamentos sobre dados espaciais, permitindo ao usuário especificar um algoritmo.

Os operadores SGB-U, SGB-A e SGB-D, permitem a criação de grupos baseados em similaridade, porém tratam cada dimensão individualmente, não sendo adequados para dados multidimensionais ou métricos. Os operadores SGB-Any e SGB-All permitem a criação de grupos em domínios multidimensionais, tratando cada dimensão de forma correlacionada com as demais.

O operador CUBE é uma extensão ao operador `GROUP BY` que permite executar de maneira eficiente e em uma única consulta agregações que sem o uso do operador necessitariam de diversas consultas. Os *simsets* possuem algumas propriedades interessantes como o conceito de suficientemente similar, o grafo de similaridade e as políticas de extração. O operador *skyline* que auxilia nos processos de decisão filtrando pontos de interesse em conjuntos de dados, também são citadas algumas extensões à este operador.

O [Quadro 2](#) traz um comparativo de características das abordagens utilizadas como o tipo de dados utilizado e os parâmetros necessários para cada uma.

Quadro 2 – Comparação entre os trabalhos relacionados

Abordagem	Referência	Tipo de dados	Parâmetros
Context	Schallehn, Sattler e Saake (2001)	Definido pelo usuário	Função de agrupamento, Atributo(s) de agrupamento
Strict	Schallehn, Sattler e Saake (2001)	Definido pelo usuário	Função(ões) de distância, Atributo(s) de agrupamento, limiar ϵ
Transitive	Schallehn, Sattler e Saake (2001)	Definido pelo usuário	Função(ões) de distância, Atributo(s) de agrupamento, limiar ϵ
ClusterRank	Li <i>et al.</i> (2007)	Numéricos	Atributo(s) de agrupamento, Número de grupos t , Atributo(s) de ranqueamento, Número de tuplas k
Cluster by	Zhang e Huang (2007)	Espaciais	Atributo(s) de agrupamento, [Algoritmo]
SGB-U	Silva, Aref e Ali (2009)	Unidimensional	Atributo de agrupamento, [Separação máxima], [diâmetro máximo]
SGB-A	Silva, Aref e Ali (2009)	Unidimensional	Atributo de agrupamento, Pontos centrais, [separação máxima], [diâmetro máximo]
SGB-D	Silva, Aref e Ali (2009)	Unidimensional	Atributo de agrupamento, Pontos delimitantes
SGB-Any	Tang <i>et al.</i> (2016)	Multidimensionais	Atributos de agrupamento, Função de distância, limiar ϵ
SGB-All	Tang <i>et al.</i> (2016)	Multidimensionais	Atributos de agrupamento, Função de distância, limiar ϵ , sobreposição
Cube	Gray <i>et al.</i> (1996)	Tradicionais	Atributos de agrupamento
Simsets	Pola <i>et al.</i> (2013)	Métricos	Atributos de agrupamento, Política, ξ -similaridade
Skyline	Börzsönyi, Kossmann e Stocker (2001)	Multidimensional	Atributos, Função objetivo

MÉTODOS PROPOSTOS

Neste capítulo apresentamos os operadores *Similarity Group-By* propostos neste trabalho de mestrado tendo por objetivo executar operações de agrupamento por similaridade sobre dados em um domínio complexo. Os operadores foram definidos para operar em qualquer domínio de dados onde uma função de distância que atende às propriedades de uma métrica esteja definida. A definição dos operadores é feita na [Seção 4.1](#). Uma extensão para a linguagem SQL para permitir a incorporação dos operadores em um SGBDR é apresentada na [Seção 4.2](#). Para implementar esses operadores alguns algoritmos são essenciais, e são apresentados na [Seção 4.3](#). Algumas funções de agregação que podem ser utilizadas junto com os operadores propostos são apresentadas na [Seção 4.4](#). Por fim a [Seção 4.5](#) traz uma sumarização do capítulo.

4.1 Operadores

A [Subseção 4.1.1](#) apresenta um novo operador SGB que agrupa os objetos através de uma eleição entre os elementos que estejam dentro de um limiar de distância. Na [Subseção 4.1.2](#) é proposta uma generalização dos operadores SGB existentes para realizar o agrupamento utilizando um predicado que tenha como parâmetro um número de vizinhos mais próximos ao invés de um predicado que utilize um valor de distância.

4.1.1 Operador *SGB-Vote*

Os operadores *SGB-Any* e *SGB-All* não são adequados para encontrar agrupamentos relevantes quando o espaço apresenta certa variação na distribuição dos dados.

Considere o exemplo da [Figura 14a](#), que mostra nove pontos em um espaço Euclidiano bi-dimensional. Os elementos que estão à uma distância menor que o limiar ξ na figura estão conectados por linhas pontilhadas. Para representar o limiar ξ , os elementos

p_5 e p_6 possuem um círculo de raio ξ ao seu redor, mas por simplicidade os círculos foram omitidos dos demais elementos. Um dos grupos é composto pelos pontos de p_1 a p_5 e outro é composto pelos pontos de p_6 até p_9 .

Nesta configuração, o operador *SGB-Any* encontra os grupos de maneira apropriada enquanto o operador *SGB-All* divide os elementos em mais do que dois grupos, pois existem alguns elementos que estão a uma distância maior que ξ de outros elementos do mesmo grupo, tal como p_6 e p_9 no segundo grupo. Porém, considere que em seguida fosse processado um ponto p_{10} , como mostrado na Figura 14b. O operador *SGB-Any* iria juntar ambos os grupos enquanto o *SGB-All* iria retornar mais do que dois grupos.

Figura 14 – Pontos em um espaço Euclidiano bi-dimensional.

Outros valores para o limiar ξ poderiam ser empregados no exemplo para a obtenção de melhores resultados. No entanto, o operador *SGB-Any* somente encontraria os grupos adequadamente quando o limiar ξ fosse menor que a distância entre grupos. Por outro lado, o operador *SGB-All* somente particiona os elementos de maneira adequada enquanto o limiar ξ for suficientemente grande para os grupos como um todo estar conectado. Entretanto, observa-se que uma abordagem baseada em uma eleição entre os elementos que estejam a uma distância máxima ξ seria capaz de particionar os elementos entre os grupos definidos em uma variedade maior de cenários.

Usando essa observação, desenvolvemos um novo operador SGB, denominado *Similarity Group-By Vote (SGB-Vote)* que obtém uma qualidade maior nos resultados a um custo de execução comparável ao dos operadores *SGB-All* e *SGB-Any*. A escolha de um grupo para um determinado objeto é baseada nos objetos que estão a uma distância máxima ξ deste elemento, como no *classificador knn*. Os votos são ponderados pela distância do objeto a cada um de seus vizinhos votantes, assim os elementos que estão mais próximos contribuem mais que os elementos mais distantes. O peso atribuído a cada voto é $1/d$, onde d é a distância entre o par de elementos s_1 e s_2 . Caso $d(s_1, s_2) = 0$, ambos os

elementos são atribuídos ao mesmo grupo. Caso nenhum objeto satisfaça à condição de distância ou caso os objetos que satisfaçam à condição não possuam um grupo assinalado ainda, um novo grupo é criado.

4.1.2 ***SGB baseado em vizinhos mais próximos: KNN-SGB e ANN-SGB***

A literatura sobre consultas exatas por similaridade usualmente considera dois predicados como sendo os principais: o predicado baseado em distância (consulta por abrangência) e o predicado baseado em quantidade de vizinhos (consulta aos k -vizinhos mais próximos). Dentre as consultas aproximadas por similaridade, o predicado baseado em quantidade de vizinhos é o mais utilizado.

Os predicados utilizados nos operadores *SGB-Any* e *SGB-All* são ambos baseados em distância. Entretanto, quando a distribuição dos dados apresenta variação grande na densidade de objetos em diferentes regiões do espaço, esses operadores não conseguem encontrar agrupamentos consistentes em todo o espaço. Desta forma, um outro operador, baseado na quantidade de vizinhos mais próximos, seria uma escolha mais apropriada.

Baseados nesse fato, generalizamos os operadores SGB existentes (*SGB-Any*, *SGB-All* e *SGB-Vote*) para utilizar também predicados baseados em quantidade de vizinhos. As variações baseadas em quantidade de vizinhos mais próximos irão operar da forma a seguir.

- O operador *SGB-Any* coloca no mesmo grupo todos os objetos que estejam entre os k -vizinhos mais próximos de um objeto s_i ao invés dos objetos que estão a uma distância máxima ξ .
- O operador *SGB-All* forma grupos entre os k -vizinhos mais próximos e utiliza as mesmas cláusulas de sobreposição que a variação baseada em distância.
- O operador *SGB-Vote* agrupa os objetos utilizando os k -vizinhos mais próximos, onde cada um dos k -vizinhos com grupo já atribuído contribuirá com a escolha do grupo a ser atribuído. Quando nenhum dos k vizinhos possuírem um grupo atribuído, o objeto será inserido em um novo grupo.

4.1.3 ***Identificação dos Operadores***

Para identificar adequadamente os operadores, utilizamos *Range Similarity Group-By* (Range-SGB) nos métodos baseados em distância. Utilizamos *k-Nearest Neighbors Similarity Group-By* (KNN-SGB) nos métodos exatos baseados em quantidade de vizinhos. Para identificar os métodos aproximados baseados número de vizinhos mais próximos é utilizado *Approximate k-Nearest Neighbors Similarity Group-By* (ANN-SGB).

Quando o prefixo *Range*, *KNN* ou *ANN* não for especificado, estamos nos referindo aos métodos de forma genérica independentemente do predicado utilizado. Quando um sufixo *All*, *Any* ou *Vote* não for especificado, nos referimos aos predicados genéricos utilizados, independente das técnicas de agrupamento.

4.2 Extensão SQL

É necessário fazer algumas extensões para a linguagem SQL, para que os SGBDR possam incorporar os novos operadores de maneira apropriada. Para incorporar os operadores *KNN-SGB* e *ANN-SGB*, modificamos as extensões para SQL já existentes na literatura para os operadores *Range-SGB*. Assim, a sintaxe proposta para os novos operadores que desenvolvemos fica compatível com a extensão SQL já existente na literatura para os operadores *Range-SGB-All* e *Range-SGB-Any*.

Para selecionar o operador *KNN-SGB* é empregada a palavra reservada *NEAR* junto com o parâmetro k , que representa o número de vizinhos mais próximos a ser analisado para a escolha do grupo. Para realizar o agrupamento com o operador *ANN-SGB* é utilizada a cláusula *APPROXIMATE* junto com o parâmetro k . Assim o usuário pode escolher entre usar *Range-SGB*, *KNN-SGB* e *ANN-SGB* utilizando as palavras reservadas *WITHIN*, *NEAR* ou *APPROXIMATE*, respectivamente. Para adicionar o suporte ao operador *SGB-Vote*, adicionamos a palavra reservada *DISTANCE-VOTE*.

A sintaxe proposta é mostrada no [Código-fonte 16](#). As cláusulas *DISTANCE-TO-ANY* e *DISTANCE-TO-ALL* representam, respectivamente os operadores *SGB-Any* e *SGB-All*, assim como na extensão anterior existente na literatura. O parâmetro d representa uma função de distância e é opcional, pois o SGBDR pode assumir uma função de distância padrão dependendo do domínio de dados utilizado. Para os operadores *SGB-All*, os valores que podem ser utilizados na cláusula de sobreposição são os valores empregados na literatura: *JOIN-ANY*, *ELIMINATE* ou *FORM-NEW-GROUP*.

Código-fonte 16 – Sintaxe proposta para os operadores SGB.

```

1: SELECT ...
2: FROM ...
3: GROUP BY atributo
4:   < DISTANCE-TO-ANY [ $d$ ] <WITHIN  $\xi$  | NEAR  $k$  | APPROXIMATE  $k$ >
5:   | DISTANCE-TO-ALL [ $d$ ] <WITHIN  $\xi$  | NEAR  $k$  | APPROXIMATE  $k$ >
6:   ON-OVERLAP [JOIN-ANY | ELIMINATE | FORM-NEW-GROUP]
7:   | DISTANCE-VOTE [ $d$ ] <WITHIN  $\xi$  | NEAR  $k$  | APPROXIMATE  $k$ > >

```

Os mesmos conceitos podem ser empregados para trabalhar com funções de janelamento de relações, ao invés de funções de agregação (*Window functions*). De maneira

semelhante, a sintaxe proposta utilizando funções de janela é mostrada no [Código-fonte 17](#). Dessa maneira, torna-se possível usar os operadores para obter valores de funções de agregação, mas sem que as tuplas de uma partição sejam agrupadas em uma única tupla.

Código-fonte 17 – Sintaxe proposta para os operadores SGB com funções de janelamento.

```

1: SELECT
2:   ...,
3:   f OVER (PARTITION BY atributo
4:     < DISTANCE-TO-ANY [d] <WITHIN  $\xi$  | NEAR k | APPROXIMATE k>
5:     | DISTANCE-TO-ALL [d] <WITHIN  $\xi$  | NEAR k | APPROXIMATE k>
6:     ON-OVERLAP [JOIN-ANY | ELIMINATE | FORM-NEW-GROUP]
7:     | DISTANCE-VOTE [d] <WITHIN  $\xi$  | NEAR k | APPROXIMATE k> >
8: )
9: FROM ...

```

4.3 Algoritmos

Esta seção apresenta os algoritmos que desenvolvemos para executar o novo operador proposto *SGB-Vote*. Vale destacar que os algoritmos para a execução dos operadores *Range-SGB* existentes na literatura não são aplicáveis para a execução dos operadores *KNN-SGB*. Desta forma, apresentamos nesta seção algoritmos simples, mas que podem executar ambos os operadores *KNN-SGB* quanto *Range-SGB*. Os algoritmos seguem uma abordagem baseada em laços aninhados, de maneira análoga à junções. O laço interno utiliza uma função $Query(d, c, p)$ para representar tanto consultas *KNN* ou *Range* e que podem ser executadas seguindo uma abordagem sequencial ou usando um método indexado de acesso, como R-Tree ou Slim-Tree. Esta abordagem possui o benefício de permitir empregar as técnicas de otimização existentes para as consultas de seleção por similaridade. Nos algoritmos desta seção, a função *getGroup* utiliza uma estrutura do tipo *map*.

4.3.1 SGB-Any

O [Algoritmo 1](#) mostra o pseudo-código para o operador *SGB-Any*. Ele executa uma consulta de seleção por abrangência ou pelos vizinhos mais próximos para cada objeto de dado. Cada objeto que satisfaz as restrições é incluído no mesmo grupo do objeto que é o centro da consulta. O resultado do algoritmo é uma estrutura *union-find*, onde cada componente conectado é incluído no mesmo grupo.

Algoritmo 1 – SGB-Any

```

1: procedimento SGB-ANY( $S, p, d$ )           ▷  $S$ : Conjunto de objetos,  $p$ : predicado de
   similaridade,  $d$ : Função de distância
2:    $G \leftarrow \text{makeSet}(S.size())$ 
3:   para todo  $s_i \in S$  faça
4:      $result \leftarrow \text{Query}(d, s, p)$ 
5:     para todo  $r \in result$  faça
6:        $G.union(s_i.id, r.id)$ 
7:     fim para
8:   fim para
9:   retorna  $G$                                ▷ Conjunto de Grupos
10: fim procedimento

```

4.3.2 SGB-All

O Algoritmo 2 apresenta o pseudo-código para o operador *SGB-All*. Ele executa uma consulta por similaridade para cada objeto s_i presente nos dados de entrada. Para cada objeto no resultado da consulta por similaridade com um grupo previamente atribuído, o identificador deste grupo é inserido em um *multiset* de sobreposição. Depois o número de ocorrências de cada partição no *multiset* de sobreposição é comparado com o número de elementos contidos nos grupos. Se os valores forem iguais, isso significa que as restrições de similaridade são satisfeitas para todos os objetos incluídos nesse grupo, e então o identificador deste grupo é inserido em um conjunto de candidatos.

Em seguida, se o conjunto de candidatos estiver vazio, um novo grupo é criado para o objeto. Se o conjunto de candidatos possuir um único elemento, é porque não existe sobreposição e então o objeto é inserido nesse grupo. Quando houver mais de um elemento no conjunto de candidatos, uma ação é tomada de acordo com o valor da cláusula de sobreposição *CLS*. Os valores possíveis para esta cláusula são *JOIN-ANY*: que insere o objeto em qualquer um dos grupos em sobreposição, *ELIMINATE*: que descarta o objeto que encontra em sobreposição, e *FORM-NEW-GROUP*: que cria um novo grupo e insere o objeto em sobreposição.

4.3.3 SGB-Vote

O Algoritmo 3 apresenta o pseudo-código para o operador *SGB-Vote*. Ele executa uma consulta por similaridade para cada objeto s_i nos dados de entrada. Para cada objeto no resultado da consulta que possuir um grupo previamente atribuído, uma entrada em uma estrutura *map* é incrementada com o inverso da distância entre s_i e r (ou é atribuído ∞ se distância = 0) assim os objetos mais próximos contribuem mais para a escolha do grupo que objetos mais distantes. Quando a estrutura *map* de pesos for nulo significa que nenhum dos objetos que satisfaça ao predicado de similaridade possui um grupo previamente atribuído, então um novo grupo é criado para o objeto c . Caso contrário, o

Algoritmo 2 – SGB-All

```

1: procedimento SGB-ALL( $S, p, d, CLS$ )    ▷  $S$ : Conjunto de objetos,  $p$ : predicado de
   similaridade,  $d$ : Função de distância,  $CLS$ : Cláusula de sobreposição
2:    $G \leftarrow NULL$ 
3:   para todo  $s_i \in S$  faça
4:      $candidatos \leftarrow NULL$ 
5:      $overlap \leftarrow NULL$ 
6:      $result \leftarrow Query(d, s_i, p)$ 
7:     para todo  $r \in result$  faça
8:        $g \leftarrow getGroup(r, G)$ 
9:       se  $g$  não está vazio então
10:         $overlap.insert(g)$ 
11:       fim se
12:     fim para
13:     para todo  $distinct(o) \in overlap$  faça
14:       se  $overlap.count(o) = G[o].size()$  então
15:         $candidatos.insert(o)$ 
16:       fim se
17:     fim para
18:     se  $candidatos$  está vazio então
19:       Cria um novo grupo e insere  $s_i$ 
20:     senão se  $candidatos.size() = 1$  então
21:       Insere  $s_i$  em no grupo contido no conjunto  $candidatos$ 
22:     senão
23:       se  $CLS = \text{"JOIN-ANY"}$  então
24:         Insere  $s_i$  em algum dos grupos contidos no conjunto  $candidatos$ 
25:       senão se  $CLS = \text{"FORM-NEW-GROUP"}$  então
26:         Cria um novo grupo e insere  $s_i$ 
27:       senão se  $CLS = \text{"ELIMINATE"}$  então
28:         Descarta o ponto  $s_i$ 
29:       fim se
30:     fim se
31:   fim para
32:   retorna  $G$                                 ▷ Conjunto de Grupos
33: fim procedimento

```

objeto c é assinalado ao grupo que possuir o maior valor de peso.

4.4 Funções de Agregação

As funções de agregação existentes nos SGBDR continuam úteis para serem utilizadas junto com os operadores de agrupamento por similaridade. No entanto, algumas novas funções podem ser interessantes, podendo incorporar informações sobre as distâncias dos elementos ou outras informações sobre o domínio e distribuição dos dados.

Podemos definir algumas funções de agregação que podem ser uteis para o con-

Algoritmo 3 – SGB-Vote

```

1: procedimento SGB-VOTE( $S, p, d$ )           ▷  $S$ : Conjunto de objetos,  $p$ : predicado de
   similaridade,  $d$ : Função de distância
2:    $G \leftarrow NULL$ 
3:   para todo  $s_i \in S$  faça
4:      $weights \leftarrow NULL$ 
5:      $result \leftarrow Query(d, s_i, p)$ 
6:     para todo  $r \in result$  faça
7:        $g \leftarrow getGroup(r, G)$ 
8:       se  $g$  não está vazio então
9:         se  $r.distance = 0$  então
10:           $weights[g] \leftarrow \infty$ 
11:        senão
12:           $weights[g] \leftarrow weights[g] + (1/r.distance)$ 
13:        fim se
14:      fim se
15:    fim para
16:    se  $weights$  está vazio então
17:      Cria um novo grupo e insere  $s_i$ 
18:    senão
19:      Insere  $s_i$  no grupo com maior valor de peso
20:    fim se
21:  fim para
22:  retorna  $G$                                ▷ Conjunto de Grupos
23: fim procedimento

```

texto de similaridade no geral. Uma função útil para diversos domínios é a que retorna o valor de um atributo para o elemento mais central de um grupo, ou uma função que retorna todos os valores para um determinado atributo. Funções estatísticas aplicadas sobre distâncias entre os elementos de um grupo também podem ser uteis, como distância mínima, distância média, distância máxima, desvio padrão e variância das distâncias.

Para domínios espaciais existem algumas funções de agregação na literatura que foram apresentadas na [Subseção 3.1.3](#). Essas funções podem agregar pontos para outros objetos como linhas ou polígonos, calcular o centroide de um grupo, ou extrair informações como comprimento ou área de objetos espaciais.

Para domínios textuais pode-se definir funções de agregações que retornem as palavras que mais ocorrem em um grupo junto com as respectivas frequências. Outra função útil é calcular o número de palavras distintas para cada grupo, ou seja o tamanho do vocabulário do grupo. Para domínios de dados de imagens, é interessante retornar informações que podem ser extraídas sobre as imagens do grupo como um todo, como a frequência de cores das imagens pertencentes a um grupo.

O [Quadro 3](#) traz algumas funções de agregação que podem ser uteis, juntamente com os parâmetros, o domínio de dados complexos sobre o qual ela é processada, o domínio

do valor de retorno, e uma descrição da função de agregação. Além destas funções de agregação outras poderiam ser definidas para um domínio específico por especialistas.

Quadro 3 – Funções de agregação

Função	Domínio	Retorno	Descrição
CENTER(A)	Qualquer	Tipo do atributo A	Retorna o valor para o atributo A do elemento central do grupo.
DISTINCT_VALUES(A)	Qualquer	Lista	Retorna uma lista contendo todos os valores distintos do atributo A.
MIN_DIST(S)	Complexo	Numérico	Retorna a menor distância entre dois elementos de um grupo.
MAX_DIST(S)	Complexo	Numérico	Retorna a maior distância entre dois elementos de um grupo.
AVG_DIST(S)	Complexo	Numérico	Retorna a distância média entre dois elementos de um grupo.
STD_DIST(S)	Complexo	Numérico	Retorna o desvio padrão para a média das distâncias entre dois elementos de um grupo.
VAR_DIST(S)	Complexo	Numérico	Retorna a variância para a média das distâncias entre dois elementos de um grupo.
DISTINCT_WORDS(S)	Texto	Numérico	Retorna o número de palavras distintas que ocorrem no grupo.
COUNT_WORDS(S)	Texto	Lista	Retorna uma lista contendo pares de palavras e frequência.
TOP_WORDS(S [,w])	Texto	Lista	Retorna uma lista contendo pares de palavras e frequência para as w palavras com maior ocorrência.
AVG_HISTOGRAM(S)	Imagem	Histograma	Retorna o histograma médio para as imagens contidas no grupo.
MAKE_POLYGON(S)	Pontos	Polígono	Retorna o polígono formado pelos pontos contidos no grupo.

4.5 Considerações

Neste capítulo foi apresentado um novo operador denominado *SGB-Vote*, para agrupar os objetos de dados utilizando uma eleição entre os elementos que satisfaçam um predicado por similaridade. Também foram propostas generalizações dos operadores SGB existentes para utilizarem predicados baseados em número de vizinhos mais próximos ao invés de um limiar de distância.

Para possibilitar a incorporação dos novos operadores *Similarity Group-By* em SGBDR, foi proposta uma extensão para a linguagem SQL. Esta extensão foi pensada para ser compatível com uma extensão existente na literatura para outros operadores SGB.

Foram propostos algoritmos simples para a execução dos novos operadores SGB, que também são compatíveis com os operadores existentes na literatura. Também foram apresentadas algumas funções de agregação que poderiam ser utilizadas em conjunto com os operadores SGB.

EXPERIMENTOS

Neste capítulo são descritos alguns dos experimentos realizados para avaliação dos operadores propostos. A [Seção 5.1](#) descreve a implementação dos operadores e como foram realizados os experimentos. A [Seção 5.2](#) descreve os conjuntos de dados utilizados para avaliação dos operadores descrevendo o tamanho, domínio de dados e outros atributos. Na [Seção 5.3](#) são apresentados os resultados obtidos pelos operadores SGB e também os tempos de execução para os operadores sobre os conjuntos de dados apresentados na [Seção 5.2](#). Por fim, a [Seção 5.4](#) traz uma sumarização do capítulo.

5.1 Implementação

Os operadores de agrupamento descritos no capítulo anterior foram implementados na linguagem C++ em um arcabouço (*framework*) próprio. Os operadores foram implementados utilizando uma abordagem de laços aninhados, conforme descrito nos algoritmos. Foram utilizadas as bibliotecas *Arboretum* (GBDI, 2017) e *Annoy* (SPOTIFY, 2017). A biblioteca *Arboretum* foi utilizada para as consultas exatas, enquanto a biblioteca *Annoy* foi utilizada para as consultas aproximadas.

Os operadores *Range-SGB* e *KNN-SGB* foram implementados utilizando o MAM Slim-Tree da biblioteca *Arboretum*, sendo armazenado em disco usando um tamanho de página de 64kb. Os operadores *ANN-SGB* foram implementados usando a biblioteca *Annoy* com LSH utilizando projeções aleatórias. As estruturas de dados utilizam arquivos mapeados em memória com chamadas de sistema (*mmap*), ficando sob a responsabilidade do sistema operacional o carregamento dos arquivos de índice para a memória.

Para a avaliação de qualidade foram comparados os resultados obtidos dos operadores de agrupamentos propostos e dos existentes na literatura com os particionamentos ideais para cada conjunto de dados. Para isto foi utilizado o método AMI apresentado

na Subseção 2.3.1. Os operadores *SGB-All*, *SGB-Vote* e *ANN-SGB* são dependentes da ordem de processamento dos dados, ou seja, podem produzir resultados diferentes quando os dados processados em ordem diferentes. Para uma comparação justa, os operadores foram executados 10 vezes com diferentes ordens aleatórias para os dados de entradas e calculado o valor médio e o desvio padrão para os valores obtidos pelo AML.

Para a comparação do tempo de execução, todos operadores foram executados na mesma plataforma Intel Core™ i5-5200U com *clock* de 2.20GHz, 8Gb de memória RAM DDR3 1600MHz e disco rígido com 5400 RPM utilizando o sistema operacional Fedora 23. Foram variados os parâmetros (ξ e k) utilizados pelos algoritmos. Os operadores foram executados 10 vezes e o valor reportado corresponde ao o tempo de processamento médio e o desvio padrão.

5.2 Conjuntos de Dados

Para a avaliação dos operadores propostos, foram utilizados conjuntos de dados reais e sintéticos. A Tabela 7 apresenta as características principais dos conjuntos de dados utilizados, indicando a cardinalidade, número de dimensões, função de distância utilizada e uma descrição do conjunto.

Tabela 7 – Descrição dos conjuntos de dados.

Nome	Tamanho	Dim	Grupos	Função de Distância	Descrição
Aggregation	788	2	7	Euclidiana	Grupos de diferentes tamanhos e formas.
Compound	399	2	6	Euclidiana	Grupos de diferentes tamanhos e formas.
D31	3100	2	31	Euclidiana	Grupos Gaussianos distribuídos aleatoriamente.
R15	600	2	15	Euclidiana	Grupos Gaussianos dispostos em círculos.
Corel	10000	128	100	Manhattan	Imagens representadas usando o descritor <i>MPEG-7 Color Structure</i> .
GHIM	10000	128	500	Manhattan	Imagens representadas usando o descritor <i>MPEG-7 Color Structure</i> .
Sport	737	–	5	Jaccard	Documentos extraídos do site <i>BBC Sport</i> representados usando o modelo <i>bag-of-words</i> .
News	2225	–	5	Jaccard	Documentos extraídos do site <i>BBC News</i> representados usando o modelo <i>bag-of-words</i> .

Foram utilizados quatro conjuntos de dados sintéticos bi-dimensionais distribuídos em um espaço Euclidiano contendo grupos de diferentes formas e tamanhos¹. O conjunto de dados **Aggregation** contém sete grupos de pontos com diferentes tamanhos e formas. O conjunto **Compound** é composto por seis grupos de diferentes tamanhos e formas. O conjunto **D31** contém 31 grupos Gaussianos distribuídos aleatoriamente. O conjunto de dados **R15** é composto por 15 grupos Gaussianos, 14 deles distribuídos em dois círculos sobre um grupo central. A Figura 15 mostra uma representação gráfica destes conjuntos de dados.

¹ Disponíveis em <http://cs.joensuu.fi/sipu/datasets/>.

Figura 15 – Conjuntos de dados sintéticos.

Para avaliar os operadores de agrupamento sobre conjuntos de dados de imagens foram utilizados os conjuntos de dados **Corel** e **GHIM**². O conjunto **Corel** possui 100 grupos distintos com 100 imagens em cada grupo, totalizando 10.000 imagens. O conjunto **GHIM** possui 10.000 imagens distribuídas em 20 grupos que contém 500 imagens cada. A Figura 16 mostra algumas imagens contidas em cada um dos conjuntos. Para representar as imagens foram utilizados vetores de características obtidos pelo descritor *MPEG-7 Color Structure* totalizando 128 características para cada imagem. As características foram comparados utilizando a função de distância *Manhattan*.

Figura 16 – Exemplo de imagens dos conjuntos de dados **Corel** e **GHIM**.

Para avaliar os operadores sobre conjuntos adimensionais, foram utilizados os conjuntos **News** e **Sport**³. O conjunto **News** é composto por 2.225 documentos extraídos do site *BBC News*, distribuídos em 5 grupos (negócios, entretenimento, política, esporte e tecnologia) com um vocabulário contendo 9.635 palavras distintas. O conjunto **Sport** é composto por 737 documentos extraídos do site *BBC Sport*, que estão distribuídos em 5 grupos (atletismo, críquete, futebol, rugby e tênis) com um vocabulário consistindo de 4.613 palavras distintas. Os documentos foram representados utilizando o modelo *bag-of-words* binário com as palavras contidas no documento e foram comparados pela função de distância Jaccard.

² Disponíveis em <http://www.ci.gxnu.edu.cn/cbir/dataset.aspx>.

³ Disponíveis em <http://mlg.ucd.ie/datasets/bbc.html>.

5.3 Resultados

Os resultados obtidos são apresentados nesta seção. Na [Subseção 5.3.1](#) avalia-se a qualidade dos agrupamentos obtidos pelos diversos operadores. Na [Subseção 5.3.2](#), os tempos de execução dos diversos operadores utilizando os algoritmos propostos são comparados. Na [Subseção 5.3.3](#) são mostrados alguns dos particionamentos obtidos nos conjuntos de dados sintéticos.

5.3.1 Qualidade dos Agrupamentos

Para avaliar a qualidade dos operadores, comparamos os particionamentos obtidos com os particionamentos ideais utilizando a Informação Mútua Ajustada (AMI). Para os gráficos aqui apresentados, o eixo vertical traz a média e o desvio padrão para os valores de AMI, enquanto o eixo horizontal representa os diferentes valores para os parâmetros ξ e k .

A [Figura 17](#) traz um comparativo da qualidade dos agrupamentos obtidos pelos operadores *Range-SGB* nos conjuntos de dados sintéticos. Para todos os conjuntos sintéticos testados, os melhores valores de AMI foram obtidos utilizando o *Range-SGB-Vote*. Variando o limiar de abrangência, nota-se que para o *SGB-Any* é melhor para pequenos limiares, o *SGB-All* é melhor pra grandes limiares, enquanto o *SGB-Vote* é melhor para limiares intermediários.

Figura 17 – Valores de AMI para conjuntos de dados sintéticos usando *Range-SGB*.

A [Figura 18](#) traz um comparativo da qualidade dos agrupamentos obtidos pelos operadores *KNN-SGB* nos conjuntos de dados sintéticos. Para todos os conjuntos sintéticos testados, os melhores valores de AMI foram obtidos utilizando o *KNN-SGB-Vote*. Para todos os conjuntos, o *SGB-Vote* é o melhor para limiares intermediários de k . No entanto para estes conjuntos, os valores de desvio padrão para o *SGB-Vote* foram bastante elevados se comparados aos demais operadores.

Figura 18 – Valores de AMI para conjuntos de dados sintéticos usando *KNN-SGB*.

A Figura 19 traz um comparativo de qualidade dos agrupamentos obtidos pelos operadores *ANN-SGB* nos conjuntos de dados sintéticos. Pode-se notar que nestes conjuntos de dados, a qualidade dos agrupamentos obtidos pelo *ANN-SGB* é igual a qualidade dos agrupamentos obtidos pelo *KNN-SGB* para a maioria dos valores de k . A principal diferença nos resultados foi entre os operadores *KNN-SGB-Any* e *ANN-SGB-Any*. O *KNN-SGB-Any* apresenta sempre os mesmos agrupamentos independente da ordenação dos dados, enquanto o *ANN-SGB-Any* pode retornar resultados diferentes dependendo da ordenação dos dados.

Figura 19 – Valores de AMI para conjuntos de dados sintéticos usando *ANN-SGB*.

A Figura 20 mostra os resultados para os conjuntos de dados reais utilizando *Range-SGB*. Para os conjuntos de dados textuais, o limiar ξ foi variado entre 0.6 e 0.975 e os melhores resultados foram obtidos pelo *Range-SGB-Vote*, com ξ em torno de 0.9. Para os conjuntos de imagens, o limiar ξ foi variado entre 0.25 e 5.0 e os melhores resultados foram com ξ em torno de 4.0. Para estes conjuntos, podemos notar que o operador *SGB-Any* não tem muita utilidade, enquanto o operador *SGB-Vote* é melhor para valores pequenos e médios de ξ e o *SGB-All* é melhor para valores grandes de ξ .

A Figura 21 mostra os resultados para os conjuntos de dados reais usando *KNN-SGB*. O valor de k foi variado entre 1 e 120 para os conjuntos de dados de texto e entre

Figura 20 – Valores de AMI para conjuntos de dados reais usando *Range-SGB*.

1 e 50 para os conjuntos de dados de imagens. Para ambos os conjuntos de imagem e de textos os melhores resultados foram obtidos utilizando *KNN-SGB-Vote*. Em geral, para os conjuntos de dados textuais e de imagens testados, os resultados obtidos pelo operador *KNN-SGB-Vote* são melhores do que os resultados obtidos por qualquer um dos operadores *Range-SGB*.

Figura 21 – Valores de AMI para conjuntos de dados reais usando *KNN-SGB*.

Os resultados obtidos de AMI para os conjuntos de dados de imagens são usando os operadores *ANN-SGB* mostrados na 22. O valor de k foi variado entre 1 e 50 para os conjuntos de dados de imagens. Em geral, os resultados obtidos pelo operador *ANN-SGB* foram semelhantes aos resultados obtidos pelos operadores *KNN-SGB*. Para esses conjuntos, a variação dos valores de AMI em números absolutos foi apenas a partir do 3º dígito decimal para a maioria das execuções.

5.3.2 Tempo de Execução

A Figura 23 mostra uma comparação dos tempos de execução para os operadores *Range-SGB* variando o valor de ξ para os conjuntos de dados sintéticos. No eixo horizontal, temos representados os valores para ξ e no eixo vertical o valor médio e o desvio padrão o tempo de processamento em 10 execuções dos operadores em segundos.

Figura 22 – Valores de AMI para conjuntos de imagens usando *ANN-SGB*. Figura 23 – Tempo de execução (em segundos) para os operadores *Range-SGB* nos conjuntos de dados sintéticos.

A Figura 24 mostra uma comparação dos tempos de execução para os operadores *KNN-SGB* variando o valor de k para os conjuntos de dados sintéticos.

Figura 24 – Tempo de execução (em segundos) para os operadores *KNN-SGB* nos conjuntos de dados sintéticos.

A Figura 25 mostra uma comparação dos tempos de execução para os operadores *ANN-SGB* variando o valor de k para os conjuntos de dados sintéticos.

A Figura 26 mostra uma comparação dos tempos de execução para os operadores

Figura 25 – Tempo de execução (em segundos) para os operadores *ANN-SGB* nos conjuntos de dados sintéticos.

Range-SGB variando o valor de ξ para os conjuntos de dados reais. A Figura 27 apresenta uma comparação entre k e o tempo de processamento para os operadores *KNN-SGB* para os conjuntos de dados reais, onde o eixo horizontal representa os valores de k . As figuras 26 e 27 mostram que para estes conjuntos de dados não existe diferenças significativas entre o tempo de execução para nenhum dos operadores *SGB-Any*, *SGB-All* e *SGB-Vote* utilizando o mesmo tipo de predicado.

Figura 26 – Tempo de execução para os operadores *Range-SGB* nos conjuntos de dados reais.

Figura 27 – Tempo de execução para os operadores *KNN-SGB* nos conjuntos de dados reais.

A Figura 28 apresenta uma comparação entre k e o tempo de processamento para os operadores *ANN-SGB* para os conjuntos de dados de imagens. Os tempos obtidos por estes operadores foram menores que os tempos obtidos pelos operadores *KNN-SGB*, mesmo com resultados obtidos sendo semelhantes.

Apesar disto, como as estruturas adotam abordagens diferentes para acesso aos arquivos de índices, é difícil mensurar com precisão a melhora no tempo de execução dos operadores *ANN-SGB* sobre os operadores *KNN-SGB*. No entanto, podemos afirmar que a utilização dos operadores *ANN-SGB* pode ser vantajosa, visto que pode reduzir o tempo de execução das consultas mantendo resultados com qualidade similar.

Figura 28 – Valores de AMI para conjuntos de imagens usando *ANN-SGB*.

O resultado geral da avaliação do tempo de execução de todas as variações de operadores e conjuntos de dados é que a nossa proposta de usar o critério de votos para os operadores de agrupamento tem o tempo de execução sempre semelhante às demais alternativas mais rápidas, e sempre geram resultados de qualidade igual ou melhor do que as demais alternativas.

5.3.3 Particionamentos Obtidos

Nesta seção apresentamos a visualização dos particionamentos com maiores valores de AMI para os conjuntos de dados sintéticos, dentre todas as execuções dos operadores, aproveitando o fato que esses conjuntos de dados são bidimensionais e utilizam a distância Euclidiana. Desta forma, podemos analisar visualmente os grupos obtidos pelos diferentes operadores de agrupamento.

A Figura 29 traz os grupos obtidos sobre o conjunto de dados *Aggregation*. O particionamento ideal para este conjunto é mostrado na Figura 29a, enquanto as demais figuras trazem os particionamentos obtidos para os diversos operadores SGB. As principais regiões onde os grupos foram encontrados de maneira incorreta estão circuladas. As legendas das figuras trazem os operadores utilizados e os parâmetros utilizados (ξ ou k), além do valor de AMI obtido.

De maneira análoga, as figuras 30, 31 e 32 Os melhores resultados para cada conjunto de dados estão destacados nas imagens. trazem os particionamentos obtidos para os conjuntos de dados *Compound*, *d31* e *R15*, respectivamente.

Como pode ser notado, os melhores resultados sempre foram obtidos utilizando os operadores que utilizam o critério de votos (*SGB-Vote*), seja usando predicados baseados em distância ou em número de vizinhos.

5.4 Considerações

Neste capítulo detalhamos como foram avaliados os operadores SGB. Os operadores foram testados em diversos conjuntos de dados reais e sintéticos de diferentes domínios. No geral, para os conjuntos de dados sintéticos o operador *Range-SGB-Vote* obteve resultados superiores aos demais operadores. Para os conjuntos de dados reais o operador *KNN-SGB-Vote* obteve resultados sempre superiores aos demais operadores.

Além disto, os operadores *ANN-SGB* tiveram um particionamentos similares aos operadores *KNN-SGB*. Desta forma, a utilização de consultas aproximadas para a realização dos agrupamentos pode resultar em redução no tempo de execução com pouca ou nenhuma perda de qualidade nos agrupamentos, tendo inclusive resultados superiores em alguns casos. Por exemplo, no conjunto de dados *R15*, com o parâmetro $k = 3$; o operador *ANN-SGB-Any* obteve um valor de $AMI=0,91$, enquanto o operador *KNN-SGB-Any* obteve um valor de $AMI=0,80$. A diferença obtida nos resultados pode ser vista nas figuras 32e e 32h.

Pode-se destacar que o operador *ANN-SGB-Any* obtém resultados diferentes dependendo da ordenação dos dados, enquanto os operadores *KNN-SGB-Any* e *Range-SGB-Any* retornam sempre os mesmos resultados, independente da ordem dos dados.

Figura 29 – Particionamentos obtidos sobre o conjunto **Aggregation**. (a) Partições ideais
Aggregation. (b) Range-SGB-Any $\xi = 1,5$; AMI=0,79. (c) Range-SGB-All $\xi = 14,5$; AMI=0,90. (d) Range-SGB-Vote
 $\xi = 6$; AMI=0,97. (e) KNN-SGB-Any $k = 11$; AMI=0,79. (f) KNN-SGB-All $k = 371$; AMI=0,82. (g) KNN-SGB-Vote $k = 91$; AMI=0,93. (h) ANN-SGB-Any $k = 11$; AMI=0,79. (i) ANN-SGB-All $k = 401$; AMI=0,81. (j) ANN-SGB-Vote $k = 91$; AMI=0,93.

Figura 30 – Particionamentos obtidos sobre o conjunto **Compound**.

(a) Partições ideais.

(b) Range-SGB-Any $\xi = 1, 5$; AMI=0,70. (c) Range-SGB-All $\xi = 12$; AMI=0,70. (d) Range-SGB-Vote $\xi = 4, 5$; AMI=0,78. (e) KNN-SGB-Any $k = 1$; AMI=0,25. (f) KNN-SGB-All $k = 211$; AMI=0,70. (g) KNN-SGB-Vote $k = 71$; AMI=0,75. (h) ANN-SGB-Any $k = 1$; AMI=0,25. (i) ANN-SGB-All $k = 211$; AMI=0,68. (j) ANN-SGB-Vote $k = 71$; AMI=0,75.

Figura 31 – Particionamentos obtidos sobre o conjunto D31.

(a) Partições ideais.

(b) Range-SGB-Any $\xi = 0,5$; AMI=0,80. (c) Range-SGB-All $\xi = 3,5$; AMI=0,70. (d) Range-SGB-Vote $\xi = 1,75$; AMI=0,93. (e) KNN-SGB-Any $k = 1$; AMI=0,25. (f) KNN-SGB-All $k = 216$; AMI=0,79. (g) KNN-SGB-Vote $k = 86$; AMI=0,95. (h) ANN-SGB-Any $k = 1$; AMI=0,25. (i) ANN-SGB-All $k = 251$; AMI=0,77. (j) ANN-SGB-Vote $k = 91$; AMI=0,95.

Figura 32 – Particionamentos obtidos sobre o conjunto R15.

(a) Partições ideais R15.

(b) Range-SGB-Any $\xi = 0,25$; AMI=0,80. (c) Range-SGB-All $\xi = 1,5$; AMI=0,93. (d) Range-SGB-Vote $\xi = 1$; AMI=0,99. (e) KNN-SGB-Any $k = 3$; AMI=0,80. (f) KNN-SGB-All $k = 73$; AMI=0,93. (g) KNN-SGB-Vote $k = 43$; AMI=0,99. (h) ANN-SGB-Any $k = 3$; AMI=0,91. (i) ANN-SGB-All $k = 81$; AMI=0,93. (j) ANN-SGB-Vote $k = 37$; AMI=0,99.

CONCLUSÕES

6.1 Contribuições

No decorrer do mestrado, propusemos o *SGB-Vote*, que consiste de uma nova forma de agrupar dados por similaridade. Também foram propostas generalizações tanto para os operadores existentes e quanto para o novo operador *SGB-Vote*, para que todas possam ser usadas também para executar consultas aos k -vizinhos mais próximos *KNN-SGB* e consultas aproximadas aos k -vizinhos mais próximos *ANN-SGB*.

Foram propostos algoritmos para executar todos os operadores propostos. Esses algoritmos são genéricos, de modo que também podem ser utilizados para os operadores existentes na literatura. Todos os algoritmos foram implementados em um arcabouço próprio desenvolvido em linguagem C++ utilizando a biblioteca *Arboretum* para as consultas exatas e a biblioteca *Annoy* para as consultas aproximadas. Para testar o desempenho dos operadores, foram utilizados conjuntos de dados sintéticos e dados reais de diferentes domínios, variando desde baixa até alta dimensionalidade e também em domínios métricos adimensionais.

Para os conjuntos de dados sintéticos, os experimentos mostraram que o operador *Range-SGB-Vote* é uma opção melhor, em termos de qualidade dos agrupamentos obtidos, do que os operadores *Range-SGB* existentes na literatura. Para os conjuntos de dados reais, os operadores *KNN-SGB-Vote* e *ANN-SGB-Vote* tiveram um desempenho superior aos demais operadores. O tempo de execução dos operadores *ANN-SGB* foram menores que os tempos de execução dos operadores *KNN-SGB*, indicando que a utilização de consultas aproximadas para a realização dos agrupamentos tende a ser uma abordagem promissora.

Além dos operadores SGB, foram propostas modificações para a linguagem SQL, de modo a permitir a incorporação dos operadores propostos aos SGBDR. Também foram apresentadas algumas funções de agregação que podem ser úteis em conjunto com os

operadores SGB.

6.2 Publicações

No decorrer deste trabalho foram publicados os seguintes artigos:

- “*On the Support of a Similarity-enabled Relational Database Management System in Civilian Crisis Situations*” (OLIVEIRA *et al.*, 2016): Trata-se de um trabalho colateral ao trabalho desenvolvido no mestrado, tendo sido desenvolvido em colaboração com outros membros do GBdI, focado no gerenciamento de situações emergenciais como incêndios através de similaridade. Para isto, foi criada uma extensão ao SGBDR PostgreSQL denominada *Kiara* que suporta alguns tipos de consultas por similaridade através de uma versão estendida do SQL. O artigo foi publicado nos anais da 18ª edição da *International Conference on Enterprise Information Systems* (ICEIS).
- “*Semantic Similarity Group By Operators for Metric Data*” (LAVERDE *et al.*, 2017): Compreende a definição dos operadores propostos *SGB-Vote* e *KNN-SGB*, junto com os algoritmos para execução. O trabalho foi apresentado na 10ª edição do *Similarity Search and Applications* (SISAP), que ocorreu em Munique na Alemanha entre os dias 4 e 6 de Outubro de 2017.

6.3 Sugestões de Futuras Pesquisas

A área de pesquisa em agrupamento por similaridade é bastante ampla. Dentre os trabalhos futuros que podem ser desenvolvidos pode-se citar:

- Inclusão dos operadores em SGBDR: A inclusão dos operadores é fundamental para que eles possam ser combinados com os demais operadores presentes nos SGBDR. Os operadores também seriam beneficiados caso existam índices construídos sobre os dados. Além disto, o planejador de consultas dos SGBDR poderia utilizar estatísticas para estimar o tempo de consulta para diversos planos de execução e escolher o plano de consulta mais promissor.
- Otimizações nos algoritmos: Algumas otimizações genéricas podem ser feitas sobre os algoritmos, como por exemplo o uso de *cache* nos resultados temporários ou a construção de índices temporários em memória. Além disto, alguns domínios de dados podem ter algumas propriedades que podem ser exploradas para agilizar a execução da etapa de junção contida nos algoritmos dos operadores.

- Desenvolvimento de novas funções de agregação: Algumas funções de agregação genéricas foram apresentadas na [Seção 4.4](#). No entanto, outros domínios ou aplicações podem demandar por outros tipos de funções de agregação.
- Explorar funções de janelamento: As funções de janelamento executam agregações sobre um conjunto de tuplas que estão relacionadas. Mas diferentemente do operador de agrupamento, que mantém apenas os valores agregados, as tuplas originais também são mantidas pelas funções de janelamento. Na [Seção 4.2](#) apresentamos uma extensão SQL, para que seja possível a utilização de funções de janelamento junto com os operadores propostos. No entanto, ainda existem diversos pontos a serem tratados.
- Combinar agrupamento por similaridade com outras relações de RI e RO: Existe na literatura diversos operadores de seleção que combinam os predicados de similaridade com predicados envolvendo RI e RO. Desta forma, pode ser interessante para algumas aplicações a inclusão deste tipo de predicado também.
- Combinar predicados de agrupamento: O operador de agrupamento padrão permite que sejam combinados diversos predicados de RI. Da mesma forma, pode ser interessante para aplicações combinar mais de um predicado de similaridade ou agrupar um predicado de agrupamento por similaridade com um predicado de RI.
- Propriedades algébricas: As propriedades algébricas são utilizadas nos SGBDR para otimização de consultas. Elas são utilizadas para transformar expressões algébricas em expressões equivalentes, aplicando um conjunto de regras de transformações. A formalização algébrica dos operadores permitiriam que as expressões sejam transformadas em expressões equivalentes.

REFERÊNCIAS

AGRAWAL, R.; GEHRKE, J.; GUNOPULOS, D.; RAGHAVAN, P. Automatic subspace clustering of high dimensional data for data mining applications. In: HAAS, L. M.; TIWARY, A. (Ed.). **SIGMOD 1998, Proceedings ACM SIGMOD International Conference on Management of Data, June 2-4, 1998, Seattle, Washington, USA**. ACM Press, 1998. p. 94–105. ISBN 0-89791-995-5. Disponível em: <http://doi.acm.org/10.1145/276304.276314>. Citado na página 49.

ANKERST, M.; BREUNIG, M. M.; KRIEGEL, H.; SANDER, J. OPTICS: ordering points to identify the clustering structure. In: DELIS, A.; FALOUTSOS, C.; GHANDEHARIZADEH, S. (Ed.). **SIGMOD 1999, Proceedings ACM SIGMOD International Conference on Management of Data, June 1-3, 1999, Philadelphia, Pennsylvania, USA**. ACM Press, 1999. p. 49–60. ISBN 1-58113-084-8. Disponível em: <http://doi.acm.org/10.1145/304182.304187>. Citado na página 49.

BARIONI, M. C. N.; KASTER, D. d. S.; RAZENTE, H. L.; TRAINA, A. J. M.; TRAINA CAETANO, J. Querying multimedia data by similarity in relational dbms. In: YAN, L.; MA, Z. (Ed.). **Advanced Database Query Systems: Techniques, Applications and Technologies**. Hershey, NY, USA: IGI Global, 2010. p. 323–359. Chapter 14. Citado nas páginas 38, 39, 40, 43 e 48.

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.; LOMET, D. B.; ALONSO, G.; LOHMAN, G. M.; KERSTEN, M. L.; CHA, S. K.; KIM, Y. (Ed.). **Proceedings of the 32nd International Conference on Very Large Data Bases, Seoul, Korea, September 12-15, 2006**. ACM, 2006. p. 1155–1158. ISBN 1-59593-385-9. Disponível em: <http://dl.acm.org/citation.cfm?id=1164232>. Citado na página 48.

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.). **20º Simpósio Brasileiro de Bancos de Dados, 3-7 de Outubro, 2005, Uberlândia, MG, Brazil, Anais/Proceedings**. UFU, 2005. p. 130–144. ISBN 85-7669-029-2. Disponível em: <http://www.sbbd-sbes2005.ufu.br/arquivos/artigo-09-BarioniRazente.pdf>. Citado na página 48.

BEDO, M. V. N.; TRAINA, A. J. M.; TRAINA JR., C. Seamless integration of distance functions and feature vectors for similarity-queries processing. **Journal of Information and Data Management**, v. 5, n. 3, p. 308–320, 2014. Disponível em: <https://seer.lcc.ufmg.br/index.php/jidm/article/view/689>. Citado na página 48.

BÖRZSÖNYI, S.; KOSSMANN, D.; STOCKER, K. The skyline operator. In: GEORAKOPOULOS, D.; BUCHMANN, A. (Ed.). **Proceedings of the 17th International Conference on Data Engineering, April 2-6, 2001, Heidelberg, Germany**. IEEE Computer Society, 2001. p. 421–430. ISBN 0-7695-1001-9. Disponível em: <http://dx.doi.org/10.1109/ICDE.2001.914855>. Citado nas páginas 64, 65 e 67.

BOZKAYA, T.; ÖZSOYOGLU, Z. M. Indexing large metric spaces for similarity search queries. **ACM Trans. Database Syst.**, v. 24, n. 3, p. 361–404, 1999. Disponível em: <http://doi.acm.org/10.1145/328939.328959>. Citado na página 44.

BUDÍKOVÁ, P.; BATKO, M.; ZEZULA, P. Query language for complex similarity queries. In: MORZY, T.; HÄRDER, T.; WREMBEL, R. (Ed.). **Advances in Databases and Information Systems - 16th East European Conference, ADBIS 2012, Poznań, Poland, September 18-21, 2012. Proceedings**. Springer, 2012. (Lecture Notes in Computer Science, v. 7503), p. 85–98. ISBN 978-3-642-33073-5. Disponível em: http://dx.doi.org/10.1007/978-3-642-33074-2_7. Citado na página 48.

CHARIKAR, M. Similarity estimation techniques from rounding algorithms. In: REIF, J. H. (Ed.). **Proceedings on 34th Annual ACM Symposium on Theory of Computing, May 19-21, 2002, Montréal, Québec, Canada**. ACM, 2002. p. 380–388. ISBN 1-58113-495-9. Disponível em: <http://doi.acm.org/10.1145/509907.509965>. Citado na página 47.

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. (Ed.). **VLDB'97, Proceedings of 23rd International Conference on Very Large Data Bases, August 25-29, 1997, Athens, Greece**. Morgan Kaufmann, 1997. p. 426–435. ISBN 1-55860-470-7. Disponível em: <http://www.vldb.org/conf/1997/P426.PDF>. Citado na página 44.

CODD, E. F. A relational model of data for large shared data banks. **Commun. ACM**, v. 13, n. 6, p. 377–387, 1970. Disponível em: <http://doi.acm.org/10.1145/362384.362685>. Citado na página 33.

CORDEIRO, R. L. F.; FALOUTSOS, C.; TRAINA JR., C. **Data Mining in Large Sets of Complex Data**. Springer London, 2013. (Springer Briefs in Computer Science). ISBN 978-1-4471-4889-0. Disponível em: <https://doi.org/10.1007/978-1-4471-4890-6>. Citado na página 49.

DATAR, M.; IMMORLICA, N.; INDYK, P.; MIRROKNI, V. S. Locality-sensitive hashing scheme based on p-stable distributions. In: SNOEYINK, J.; BOISSONNAT, J. (Ed.). **Proceedings of the 20th ACM Symposium on Computational Geometry, Brooklyn, New York, USA, June 8-11, 2004**. ACM, 2004. p. 253–262. ISBN 1-58113-885-7. Disponível em: <http://doi.acm.org/10.1145/997817.997857>. Citado na página 47.

ELMASRI, R.; NAVATHE, S. **Fundamentals of Database Systems**. Addison-Wesley, 2011. ISBN 9780136086208. Disponível em: <https://books.google.com.br/books?id=ZdhAQgAACAAJ>. Citado nas páginas 33, 34, 35, 36, 37, 38 e 43.

ESTER, M.; KRIEGEL, H.; SANDER, J.; XU, X. A density-based algorithm for discovering clusters in large spatial databases with noise. In: SIMOUDIS, E.; HAN, J.; FAYYAD, U. M. (Ed.). **Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96), Portland, Oregon, USA**. AAAI Press, 1996. p. 226–231. ISBN 1-57735-004-9. Disponível em: <http://www.aaai.org/Library/KDD/1996/kdd96-037.php>. Citado na página 49.

- FERHATOSMANOGLU, H.; TUNCEL, E.; AGRAWAL, D.; El Abbadi, A. Approximate nearest neighbor searching in multimedia databases. In: GEORGAKOPOULOS, D.; BUCHMANN, A. (Ed.). **Proceedings of the 17th International Conference on Data Engineering, April 2-6, 2001, Heidelberg, Germany**. IEEE Computer Society, 2001. p. 503–511. ISBN 0-7695-1001-9. Disponível em: <<https://doi.org/10.1109/ICDE.2001.914864>>. Citado na página 42.
- GARCIA-MOLINA, H.; ULLMAN, J.; WIDOM, J. **Database System Implementation**. Prentice Hall, 2000. ISBN 9780130402646. Disponível em: <<https://books.google.com.br/books?id=jOVQAAAAMAAJ>>. Citado nas páginas 35, 36 e 38.
- GBDI. **Arboretum**. 2017. Disponível em: <<https://bitbucket.org/gbdi/arboretum>>. Citado na página 79.
- GIONIS, A.; INDYK, P.; MOTWANI, R. Similarity search in high dimensions via hashing. In: ATKINSON, M. P.; ORLOWSKA, M. E.; VALDURIEZ, P.; ZDONIK, S. B.; BRODIE, M. L. (Ed.). **VLDB'99, Proceedings of 25th International Conference on Very Large Data Bases, September 7-10, 1999, Edinburgh, Scotland, UK**. Morgan Kaufmann, 1999. p. 518–529. ISBN 1-55860-615-7. Disponível em: <<http://www.vldb.org/conf/1999/P49.pdf>>. Citado na página 43.
- GRAY, J.; BOSWORTH, A.; LAYMAN, A.; PIRAHESH, H. Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-total. In: **ICDE**. [S.l.]: IEEE Computer Society, 1996. p. 152–159. Citado nas páginas 60 e 67.
- GULIATO, D.; MELO, E. V. de; RANGAYYAN, R. M.; SOARES, R. C. POSTGRESQL-IE: an image-handling extension for postgresql. **J. Digital Imaging**, v. 22, n. 2, p. 149–165, 2009. Disponível em: <<http://dx.doi.org/10.1007/s10278-007-9097-5>>. Citado na página 48.
- HAN, J.; KAMBER, M.; PEI, J. **Data Mining: Concepts and Techniques**. Elsevier Science, 2011. (The Morgan Kaufmann Series in Data Management Systems). ISBN 9780123814807. Disponível em: <<https://books.google.com.br/books?id=pQws07tdpjoC>>. Citado nas páginas 48, 49 e 50.
- INDYK, P.; MOTWANI, R. Approximate nearest neighbors: Towards removing the curse of dimensionality. In: VITTER, J. S. (Ed.). **Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing, Dallas, Texas, USA, May 23-26, 1998**. ACM, 1998. p. 604–613. ISBN 0-89791-962-9. Disponível em: <<http://doi.acm.org/10.1145/276698.276876>>. Citado nas páginas 42 e 46.
- ISO/IEC. Standard for programming language C++ [Working Draft]. International Organization for Standardization (ISO), 2014. Disponível em: <<https://isocpp.org/std/the-standard>>. Citado na página 51.
- JACOX, E. H.; SAMET, H. Metric space similarity joins. **ACM Trans. Database Syst.**, v. 33, n. 2, 2008. Disponível em: <<http://doi.acm.org/10.1145/1366102.1366104>>. Citado na página 42.
- KASTER, D. S.; BUGATTI, P. H.; TRAINA, A. J. M.; TRAINA JR., C. FMI-SiR: A Flexible and Efficient Module for Similarity Searching on Oracle Database. **Journal of Information and Data Management**, v. 1, n. 2, p. 229–244, 2010. Disponível em: <<http://seer.lcc.ufmg.br/index.php/jidm/article/view/37>>. Citado na página 48.

KAUFMAN, L.; ROUSSEEUW, P. **Finding Groups in Data: An Introduction to Cluster Analysis**. Wiley, 2009. (Wiley Series in Probability and Statistics). ISBN 9780470317488. Disponível em: <<https://books.google.com.br/books?id=YeFQHikNo0C>>. Citado nas páginas 48 e 49.

LAVERDE, N. A.; CAZZOLATO, M. T.; TRAINA, A. J. M.; JR., C. T. Semantic Similarity Group By Operators for Metric Data. In: BEECKES, C.; BORUTTA, F.; KRÖGER, P.; SEIDL, T. (Ed.). **Similarity Search and Applications - 10th International Conference, SISAP 2017, Munich, Germany, October 4-6, 2017, Proceedings**. Springer, 2017. (Lecture Notes in Computer Science, v. 10609), p. 247–261. ISBN 978-3-319-68473-4. Disponível em: <https://doi.org/10.1007/978-3-319-68474-1_17>. Citado na página 94.

LESKOVEC, J.; RAJARAMAN, A.; ULLMAN, J. D. **Mining of Massive Datasets, 2nd Ed.** [S.l.]: Cambridge University Press, 2014. ISBN 978-1107077232. Citado nas páginas 40, 41 e 47.

LI, C.; WANG, M.; LIM, L.; WANG, H.; CHANG, K. C. Supporting ranking and clustering as generalized order-by and group-by. In: **SIGMOD Conference**. [S.l.]: ACM, 2007. p. 127–138. Citado nas páginas 54 e 67.

LLOYD, S. P. Least squares quantization in PCM. **IEEE Transactions on Information Theory**, v. 28, n. 2, p. 129–136, 1982. Disponível em: <<http://dx.doi.org/10.1109/TIT.1982.1056489>>. Citado na página 49.

NG, R. T.; HAN, J. Efficient and effective clustering methods for spatial data mining. In: BOCCA, J. B.; JARKE, M.; ZANIOLO, C. (Ed.). **VLDB'94, Proceedings of 20th International Conference on Very Large Data Bases, September 12-15, 1994, Santiago de Chile, Chile**. Morgan Kaufmann, 1994. p. 144–155. ISBN 1-55860-153-8. Disponível em: <<http://www.vldb.org/conf/1994/P144.PDF>>. Citado na página 49.

OLIVEIRA, P. H.; FRAIDEINBERZE, A. C.; LAVERDE, N. A.; GUALDRON, H.; GONZAGA, A. S.; FERREIRA, L. D.; OLIVEIRA, W. D.; JR., J. F. R.; CORDEIRO, R. L. F.; JR., C. T.; TRAINA, A. J. M.; SOUSA, E. P. M. de. On the support of a similarity-enabled relational database management system in civilian crisis situations. In: HAMMOUDI, S.; MACIASZEK, L. A.; MISSIKOFF, M.; CAMP, O.; CORDEIRO, J. (Ed.). **ICEIS 2016 - Proceedings of the 18th International Conference on Enterprise Information Systems, Volume 1, Rome, Italy, April 25-28, 2016**. SciTePress, 2016. p. 119–126. ISBN 978-989-758-187-8. Disponível em: <<https://doi.org/10.5220/0005816701190126>>. Citado na página 94.

POLA, I. R. V.; CORDEIRO, R. L. F.; TRAINA JR., C.; TRAINA, A. J. M. A new concept of sets to handle similarity in databases: The simsets. In: BRISABOA, N. R.; PEDREIRA, O.; ZEZULA, P. (Ed.). **Similarity Search and Applications - 6th International Conference, SISAP 2013, A Coruña, Spain, October 2-4, 2013, Proceedings**. Springer, 2013. (Lecture Notes in Computer Science, v. 8199), p. 30–42. ISBN 978-3-642-41061-1. Disponível em: <http://dx.doi.org/10.1007/978-3-642-41062-8_4>. Citado nas páginas 62 e 67.

_____. Similarity sets: A new concept of sets to seamlessly handle similarity in database management systems. **Inf. Syst.**, v. 52, p. 130–148, 2015. Disponível em: <<http://dx.doi.org/10.1016/j.is.2015.01.011>>. Citado na página 63.

SAMET, H. **Foundations of Multidimensional and Metric Data Structures (The Morgan Kaufmann Series in Computer Graphics and Geometric Modeling)**. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2005. ISBN 0123694469. Citado nas páginas 43, 44, 45 e 46.

SCHALLEHN, E.; SATTTLER, K.; SAAKE, G. Advanced grouping and aggregation for data integration. In: **CIKM**. [S.l.]: ACM, 2001. p. 547–549. Citado nas páginas 53 e 67.

SEdgeWICK, R.; WAYNE, K. **Algorithms, 4th Edition**. [S.l.]: Addison-Wesley, 2011. ISBN 978-0-321-57351-3. Citado nas páginas 50 e 51.

SILVA, Y. N.; ALY, A. M.; AREF, W. G.; LARSON, P. Simdb: a similarity-aware database system. In: ELMAGARMID, A. K.; AGRAWAL, D. (Ed.). **Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2010, Indianapolis, Indiana, USA, June 6-10, 2010**. ACM, 2010. p. 1243–1246. ISBN 978-1-4503-0032-2. Disponível em: <<http://doi.acm.org/10.1145/1807167.1807330>>. Citado na página 48.

SILVA, Y. N.; AREF, W. G.; ALI, M. H. Similarity group-by. In: IOANNIDIS, Y. E.; LEE, D. L.; NG, R. T. (Ed.). **Proceedings of the 25th International Conference on Data Engineering, ICDE 2009, March 29 2009 - April 2 2009, Shanghai, China**. IEEE, 2009. p. 904–915. ISBN 978-0-7695-3545-6. Disponível em: <<http://dx.doi.org/10.1109/ICDE.2009.113>>. Citado nas páginas 56, 59 e 67.

_____. The similarity join database operator. In: LI, F.; MORO, M. M.; GHANDEHARIZADEH, S.; HARITSA, J. R.; WEIKUM, G.; CAREY, M. J.; CASATI, F.; CHANG, E. Y.; MANOLESCU, I.; MEHROTRA, S.; DAYAL, U.; TSOTRAS, V. J. (Ed.). **Proceedings of the 26th International Conference on Data Engineering, ICDE 2010, March 1-6, 2010, Long Beach, California, USA**. IEEE, 2010. p. 892–903. ISBN 978-1-4244-5444-0. Disponível em: <<http://dx.doi.org/10.1109/ICDE.2010.5447873>>. Citado na página 42.

SPOTIFY. **Annoy (Approximate Nearest Neighbors Oh Yeah)**. 2017. Disponível em: <<https://github.com/spotify/annoy>>. Citado na página 79.

TANG, M.; TAHBOUB, R. Y.; AREF, W. G.; ATALLAH, M. J.; MALLUHI, Q. M.; OUZZANI, M.; SILVA, Y. N. Similarity group-by operators for multi-dimensional relational data. **IEEE Trans. Knowl. Data Eng.**, v. 28, n. 2, p. 510–523, 2016. Disponível em: <<http://doi.ieeecomputersociety.org/10.1109/TKDE.2015.2480400>>. Citado nas páginas 58, 60 e 67.

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. **VLDB J.**, v. 16, n. 4, p. 483–505, 2007. Disponível em: <<http://dx.doi.org/10.1007/s00778-005-0178-0>>. Citado na página 44.

TRAINA JR., C.; TRAINA, A. J. M.; FALOUTSOS, C.; SEEGER, B. Fast indexing and visualization of metric data sets using slim-trees. **IEEE Trans. Knowl. Data Eng.**, v. 14, n. 2, p. 244–260, 2002. Disponível em: <<http://dx.doi.org/10.1109/69.991715>>. Citado na página 44.

- UHLMANN, J. K. Satisfying general proximity/similarity queries with metric trees. **Inf. Process. Lett.**, v. 40, n. 4, p. 175–179, 1991. Disponível em: <[https://doi.org/10.1016/0020-0190\(91\)90074-R](https://doi.org/10.1016/0020-0190(91)90074-R)>. Citado na página 44.
- VINH, N. X.; EPPS, J.; BAILEY, J. Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. **Journal of Machine Learning Research**, v. 11, p. 2837–2854, 2010. Disponível em: <<http://portal.acm.org/citation.cfm?id=1953024>>. Citado na página 50.
- WANG, J.; SHEN, H. T.; SONG, J.; JI, J. Hashing for similarity search: A survey. **CoRR**, abs/1408.2927, 2014. Disponível em: <<http://arxiv.org/abs/1408.2927>>. Citado na página 47.
- WANG, W.; YANG, J.; MUNTZ, R. R. STING: A statistical information grid approach to spatial data mining. In: JARKE, M.; CAREY, M. J.; DITTRICH, K. R.; LOCHOVSKY, F. H.; LOUCOPOULOS, P.; JEUSFELD, M. A. (Ed.). **VLDB'97, Proceedings of 23rd International Conference on Very Large Data Bases, August 25-29, 1997, Athens, Greece**. Morgan Kaufmann, 1997. p. 186–195. ISBN 1-55860-470-7. Disponível em: <<http://www.vldb.org/conf/1997/P186.PDF>>. Citado na página 49.
- YIANILOS, P. N. Data structures and algorithms for nearest neighbor search in general metric spaces. In: RAMACHANDRAN, V. (Ed.). **Proceedings of the Fourth Annual ACM/SIGACT-SIAM Symposium on Discrete Algorithms, 25-27 January 1993, Austin, Texas**. ACM/SIAM, 1993. p. 311–321. ISBN 0-89791-562-3. Disponível em: <<http://dl.acm.org/citation.cfm?id=313559.313789>>. Citado na página 44.
- ZEZULA, P.; AMATO, G.; DOHNAL, V.; BATKO, M. **Similarity Search - The Metric Space Approach**. Kluwer, 2006. v. 32. (Advances in Database Systems, v. 32). ISBN 978-0-387-29146-8. Disponível em: <<http://dx.doi.org/10.1007/0-387-29151-2>>. Citado nas páginas 38, 39, 40, 41, 42, 45 e 46.
- ZHANG, C.; HUANG, Y. Cluster by: a new sql extension for spatial data aggregation. In: SAMET, H.; SHAHABI, C.; SCHNEIDER, M. (Ed.). **15th ACM International Symposium on Geographic Information Systems, ACM-GIS 2007, November 7-9, 2007, Seattle, Washington, USA, Proceedings**. ACM, 2007. p. 53. ISBN 978-1-59593-914-2. Disponível em: <<http://doi.acm.org/10.1145/1341012.1341077>>. Citado nas páginas 55, 57 e 67.
- ZHANG, N.; LI, C.; HASSAN, N.; RAJASEKARAN, S.; DAS, G. On skyline groups. **IEEE Trans. Knowl. Data Eng.**, v. 26, n. 4, p. 942–956, 2014. Disponível em: <<http://dx.doi.org/10.1109/TKDE.2013.119>>. Citado na página 65.
- ZHANG, X.; CHOMICKI, J. Preference queries over sets. In: ABITEBOUL, S.; BÖHM, K.; KOCH, C.; TAN, K. (Ed.). **Proceedings of the 27th International Conference on Data Engineering, ICDE 2011, April 11-16, 2011, Hannover, Germany**. IEEE Computer Society, 2011. p. 1019–1030. ISBN 978-1-4244-8958-9. Disponível em: <<http://dx.doi.org/10.1109/ICDE.2011.5767866>>. Citado na página 65.