Optimizing similarity queries in metric spaces meeting user's expectation

Monica Ribeiro Porto Ferreira Ribeiro Porto Ferreira

Optimizing similarity queries in metric spaces meeting user's expectation

Mônica Ribeiro Porto Ferreira

Optimizing similarity queries in metric spaces meeting user's expectation[1]

Mônica Ribeiro Porto Ferreira

Advisors: **Prof. Dr. Caetano Traina Jr.**
Prof. Dr. Richard Chbeir

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

USP – São Carlos
November 2012

[1] Financial supports: FAPESP (Process Number 2008/00210-7), CAPES (Process Number PDEE BEX 2451/09-3, CNPq, FAPESP-Microsoft Research and CNRS.

Ficha catalográfica preparada pela Seção de Tratamento
da Informação da Biblioteca Prof. Achille Bassi – ICMC/USP

F383o

Ferreira, Mônica Ribeiro Porto
Optimizing similarity queries in metric spaces meeting user's expectation / Mônica Ribeiro Porto Ferreira ; orientadores Caetano Traina Jr. e Richard Chbeir. –- São Carlos, 2012.
124 p.

Tese (Doutorado Duplo - Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional e *Doctorat en Informatique*) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo; *Université de Bourgogne*, 2012.

1. Similarity queries. 2. Similarity algebra. 3. Similarity query optimization. 4. User's expectation. 5. Metric spaces. I. Traina Jr., Caetano, orient. II. Chbeir, Richard, orient. III. Título.

Otimização de operações de busca por similaridade em espaços métricos atendendo à expectativa do usuário[1]

Mônica Ribeiro Porto Ferreira

Orientadores: **Prof. Dr. Caetano Traina Jr.**
Prof. Dr. Richard Chbeir

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

USP – São Carlos
Novembro de 2012

[1] Apoio financeiro: FAPESP (Processo Nº 2008/00210-7), CAPES (Processo Nº PDEE BEX 2451/09-3), CNPq, FAPESP-Microsoft Research e CNRS.

Ficha catalográfica preparada pela Seção de Tratamento
da Informação da Biblioteca Prof. Achille Bassi – ICMC/USP

Ferreira, Mônica Ribeiro Porto
F383o Otimização de operações de busca por similaridade em espaços métricos atendendo à expectativa do usuário / Mônica Ribeiro Porto Ferreira ; orientadores Caetano Traina Jr. e Richard Chbeir. –- São Carlos, 2012.
124 p.

Tese (Doutorado Duplo - Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional e *Doctorat en Informatique*) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo; *Université de Bourgogne*, 2012.

1. Consultas por similaridade. 2. Álgebra por similaridade. 3. Otimização de consultas por similaridade. 4. Expectativa do usuário. 5. Espaços métricos. I. Traina Jr., Caetano, orient. II. Chbeir, Richard, orient. III. Título.

Acknowledgments

I would like to thank my advisor and friend Prof. Dr. Caetano Traina Jr., who believed in me from the beginning when invited me to do undergraduate research. He afforded me opportunities and challenges, support and encouragement at all time, and mainly trust. My thanks to my French advisor, Prof. Dr. Richard Chbeir, who also contributed to guide my work.

My special thanks to my dear husband Leandro, for always being there, for sharing happy moments and encouraging me to overcome difficult times, for his comprehension during my absences in countless weekends and holidays that I had to work, for his sweet words that make everything seems so much easier.

I would like to thank my parents, José Maria and Mayra, my grandmother Ladice and all my family that is very large and I can not mention one by one, for their unconditional love, support and encouragement dedicated throughout my life, for their comprehension during my absences in holidays and family gatherings, for their hug and cheer all the time. I also thank my brothers José Maria Jr. and José Guilherme, my sister-in-law Josélia, my brother-in-law Rudinei, my nephews João Pedro and Diego, and my parents-in-law Maria Helena and Celso for their attention, encouragement words and for always being around when I needed. My sincere thanks to my sister-in-law Elaine for helping and teaching me many things throughout my academic life.

My special thanks to my grandmother and heroine Carminda (in memoriam) and to my aunt and godmother Dama (in memoriam) for their love and support, for their life examples, for their unconditionally encouragement dedicated throughout my life, for hugging and cheering me all the time.

My gratitude to Profa. Agma Traina at ICMC-USP for her time and effort on my thesis, for her collaborative work, ideas, words of encouragement and for her affection demonstrated always so kind. I also thanks Profa. Ires Dias for the collaborative work, time and effort invested with the algebra. My grateful to Profa. Sandra de Amo and Prof. Renato Fileto for their contribution to my work.

I dedicate my sincere thanks to my friends and colleagues of the GBdI-USP in São Carlos-SP-Brazil, especially I am grateful to Prof. Junior, Robson, Carolina, Letrícia, Jaqueline, Willian, Lucio, Sérgio and Daniel C. for their important collaboration in the lab and meetings. I also thanks my colleagues of the Le2i-uB in Dijon-France. I am grateful to my friends Marcela Ribeiro, Luciana Romani, Fekade Getahum and Elie Raad for their contributions to my work and for incentive words in moments of despair and anguish.

My thanks to Laura, Glaucia, Ana Paula, Lhais and Carolina for helping me with bureaucracies. I also thank the *Instituto de Ciências Matemáticas e de Computação* of USP in São Carlos-SP-Brazil and *Université de Bourgogne* in Dijon-France for their academic structures that became possible the development of this cotutlle work.

Finally, I acknowledge the funding agencies FAPESP and CAPES for the financial support during this doctorate. Additionally, I thanks the funding agencies CNPq and FAPESP-Microsoft Research that also supported the research undergoing at the GBdI laboratory, and CNRS that supported the research undergoing at the LE2I laboratory.

Abstract

The complexity of data stored in large databases has increased at very fast paces. Hence, operations more elaborated than traditional queries are essential in order to extract all required information from the database. Therefore, the interest of the database community in similarity search has increased significantly. Two of the well-known types of similarity search are the Range (R_q) and the k-Nearest Neighbor ($k\text{NN}_q$) queries, which, as any of the traditional ones, can be sped up by indexing structures of the Database Management System (DBMS). Another way of speeding up queries is to perform query optimization. In this process, metrics about data are collected and employed to adjust the parameters of the search algorithms in each query execution. However, although the integration of similarity search into DBMS has begun to be deeply studied more recently, the query optimization has been developed and employed just to answer traditional queries.

The execution of similarity queries, even using efficient indexing structures, tends to present higher computational cost than the execution of traditional ones. Two strategies can be applied to speed up the execution of any query, and thus they are worth to employ to answer also similarity queries. The first strategy is query rewriting based on algebraic properties and cost functions. The second technique is when external query factors are applied, such as employing the semantic expected by the user, to prune the answer space. This thesis aims at contributing to the development of novel techniques to improve the similarity-based query optimization processing, exploiting both algebraic properties and semantic restrictions as query refinements.

Title: *Optimizing similarity queries in metric spaces meeting user's expectation.*
Doctoral dissertation submitted to the *Instituto de Ciências Matemáticas e de Computação* – ICMC-USP, in partial fulfillment of the requirements for the degree of the Doctorate Program in Computer Science and Computational Mathematics. FINAL VERSION.

Resumo

A complexidade dos dados armazenados em grandes bases de dados tem aumentado sempre, criando a necessidade de novas operações de consulta. Uma classe de operações de crescente interesse são as consultas por similaridade, das quais as mais conhecidas são as consultas por abrangência (R_q) e por k-vizinhos mais próximos ($k\text{NN}_q$). Qualquer consulta é agilizada pelas estruturas de indexação dos Sistemas de Gerenciamento de Bases de Dados (SGBDs). Outro modo de agilizar as operações de busca é a manutenção de métricas sobre os dados, que são utilizadas para ajustar parâmetros dos algoritmos de busca em cada consulta, num processo conhecido como otimização de consultas. Como as buscas por similaridade começaram a ser estudadas seriamente para integração em SGBDs muito mais recentemente do que as buscas tradicionais, a otimização de consultas, por enquanto, é um recurso que tem sido utilizado para responder apenas a consultas tradicionais.

Mesmo utilizando as melhores estruturas existentes, a execução de consultas por similaridade tende a ser mais custosa do que as operações tradicionais. Assim, duas estratégias podem ser utilizadas para agilizar a execução de qualquer consulta e, assim, podem ser empregadas também para responder às consultas por similaridade. A primeira estratégia é a reescrita de consultas baseada em propriedades algébricas e em funções de custo. A segunda técnica faz uso de fatores externos à consulta, tais como a semântica esperada pelo usuário, para restringir o espaço das respostas. Esta tese pretende contribuir para o desenvolvimento de técnicas que melhorem o processo de otimização de consultas por similaridade, explorando propriedades algébricas e restrições semânticas como refinamento de consultas.

Titulo: *Otimização de operações de busca por similaridade em espaços métricos atendendo à expectativa do usuário.*
Tese apresentada ao Instituto de Ciências Matemáticas e de Computação - ICMC-USP, como parte dos requisitos para obtenção do título de Doutor em Ciências - Ciências de Computação e Matemática Computacional. VERSÃO REVISADA.

Résumé

La complexité des données contenues dans les grandes bases de données a augmenté considérablement. Par conséquent, des opérations plus élaborées que les requêtes traditionnelles sont indispensable pour extraire toutes les informations requises de la base de données. L'intérêt de la communauté de base de données a particulièrement augmenté dans les recherches basées sur la similarité. Deux sortes de recherche de similarité bien connues sont la requête par intervalle (R_q) et par k-plus proches voisins ($k\text{NN}_q$). Ces deux techniques, comme les requêtes traditionnelles, peuvent être accélérées par des structures d'indexation des Systèmes de Gestion de Base de Données (SGBDs). Une autre façon d'accélérer les requêtes est d'exécuter le procédé d'optimisation des requêtes. Dans ce procédé les données métriques sont recueillies et utilisées afin d'ajuster les paramètres des algorithmes de recherche lors de chaque exécution de la requête. Cependant, bien que l'intégration de la recherche de similarités dans le SGBD ait commencé à être étudiée en profondeur récemment, le procédé d'optimisation des requêtes a été développé et utilisé pour répondre à des requêtes traditionnelles.

L'exécution des requêtes de similarité a tendance à présenter un coût informatique plus important que l'exécution des requêtes traditionnelles et ce même en utilisant des structures d'indexation efficaces. Deux stratégies peuvent être appliquées pour accélérer l'execution de quelques requêtes, et peuvent également être employées pour répondre aux requêtes de similarité. La première stratégie est la réécriture de requêtes basées sur les propriétés algébriques et les fonctions de coût. La deuxième stratégie est l'utilisation des facteurs externes de la requête, tels que la sémantique attendue par les usagers, pour réduire le nombre des résultats potentiels. Cette thèse vise à contribuer au développement des techniques afin d'améliorer le procédé d'optimisation des requêtes de similarité, tout en exploitant les propriétés algébriques et les restrictions sémantiques pour affiner les requêtes.

Sujet de These: *Optimisation des requêtes de similarité dans les espaces métriques répondant aux besoins des usagers.*
Thèse présentée à l'*Instituto de Ciências Matemáticas e de Computação* - ICMC-USP, dans le cadre des exigences pour l'obtention du titre de Docteur en Informatique - *Ciências de Computação e Matemática Computacional.* VERSION FINALE.

Contents

List of Figures xi

List of Tables xiii

List of Abbreviations and Acronyms xv

List of Symbols xvii

1 Introduction 1
- 1.1 Motivation . . . 1
- 1.2 Problem Definition . . . 3
- 1.3 Work Goals . . . 5
- 1.4 Main Contributions . . . 5
- 1.5 Work Organization . . . 6

2 Traditional Query Optimization Process 9
- 2.1 Introduction . . . 9
- 2.2 Access Methods . . . 10
- 2.3 The Relational Algebra . . . 11
- 2.4 Query Optimization . . . 12
  - 2.4.1 Query Rewriting . . . 16
  - 2.4.2 Cost and Condition Selectivity Model . . . 17
- 2.5 Final Comments . . . 19

3 Similarity Queries 21
- 3.1 Introduction . . . 21
- 3.2 Metric Access Methods . . . 24
- 3.3 Searching for Similarity . . . 25
- 3.4 Similarity Algebra . . . 26
- 3.5 Query Optimization . . . 28
- 3.6 Query Rewriting . . . 30
- 3.7 Cost and Condition Selectivity Model . . . 32
- 3.8 Users' Preferences . . . 35
- 3.9 Data Mining . . . 37
- 3.10 Final Comments . . . 38

4 A novel approach for Similarity Query Optimization Process in DBMSs 41
- 4.1 Introduction . . . 41
- 4.2 Including similarity-based operators into the Relational Model . . . 42
- 4.3 Canonical Plan Algorithm . . . 43

4.4 Query Optimization . . . 47
4.5 Similarity Algebra for metric spaces . . . 49
4.5.1 Similarity Operations - Definitions . . . 49
4.5.2 Properties of the Range Selection . . . 51
4.5.3 Properties of the k-Nearest Neighbor Selection . . . 57
4.6 Semantic Restrictions . . . 73
4.6.1 Preference Model Module . . . 75
4.6.2 Data Mining Model Module . . . 76
4.7 Final Comments . . . 77

5 Similarity Retrieval Engine - Case Study **79**
5.1 Introduction . . . 79
5.2 The SIREN Query Optimizer . . . 84
5.2.1 Experimental Evaluation . . . 86
5.3 The SIREN Preference Model . . . 90
5.3.1 Experimental Evaluation . . . 92
5.4 The SIREN Data Mining Model . . . 96
5.4.1 Experimental Evaluation . . . 98
5.5 Final Comments . . . 102

6 Conclusion **105**
6.1 Final Considerations . . . 105
6.2 Main Contributions . . . 105
6.3 Future Works . . . 108
6.3.1 Future Applied Research . . . 108
6.3.2 Future Theoretical Research . . . 109

Bibliography **111**

A The CoPhIR Dataset **123**

List of Figures

2.1 Example of a (a) logical and a (b) physical query plans, represented as query trees for Query **Q**1. . . . 14
2.2 Query optimizer architecture according to Ioannidis [1996] and Garcia-Molina et al. [2000]. . . . 15

3.1 Examples of similarity queries in bidimensional space with Euclidean distance L_2: (a) Range query – R_q, and (b) k-nearest neighbor query – $k\text{NN}_q$, with $k = 3$ elements. . . . 23
3.2 Time line for the existing similarity algebra works, following the four approaches used in MIS. . . . 29
3.3 Time line for the existing query optimization works, following the four approaches used in MIS. . . . 31
3.4 Time line for the existing query rewriting works, following the four approaches used in MIS. . . . 33
3.5 Time line for the existing cost and condition selectivity estimation works, following the four approaches used in MIS. . . . 36

4.1 A relation composed of both simple and complex attributes. . . . 43
4.2 General form of an SQL-like query. . . . 44
4.3 (a) A query expressed in the SIREN extension of SQL to support similarity, and (b) the canonical query plan, represented as tree, for Query **Q**2. . . . 46
4.4 Similarity query optimizer architecture. . . . 48
4.5 Venn diagram representation of inclusion property of Equation 4.28. . . . 67
4.6 Generic flowchart to prepare and execute similarity queries considering of preference models. . . . 76
4.7 Generic flowchart to prepare and execute similarity queries considering of data mining models. . . . 77

5.1 SIREN_{op} architecture. . . . 80
5.2 'Similarity-first', canonical, alternative plans and execution time of Query Q3. . . . 85
5.3 'Similarity-first' plan, canonical tree, alternative plans and execution time of Query Q7. . . . 89
5.4 Processes to prepare and execute similarity queries considering of preference models. . . . 93
5.5 (a) Percentage of correct answer in the similarity-only and preference similarity queries; (b) Precision vs. Percentage (%) Interesting Answers. . . . 95
5.6 Processes to prepare and execute similarity queries considering of data mining models. . . . 98

5.7 Data flow showing how the mining rules are enabled using the `CREATE MINING MODEL` and `SET MODIFICATION` commands. 100
5.8 Results of 10NN over query enabling and disabling the use of a data mining model. The training set example was obtained from the `WondersWorld` relation with the attribute `Training = 'True'` and used to generate the rules. 104

List of Tables

4.1 Summary of unary similarity operators. 51
4.2 Summary of algebraic properties to range similarity queries. 72
4.3 Summary of algebraic properties to kNN similarity queries. 73
4.4 Summary of algebraic equivalence properties to traditional queries invalid to kNN similarity queries. 74

5.1 Real dataset descriptions used in the experiments. 83
5.2 Canonical plan, represented as a table, of the Query Q3. 85
5.3 Performance of Queries **Q**4, **Q**5 and **Q**6 (total time in milliseconds). 88
5.4 Results from several range values - range queries (average) 101
5.5 Results from several k values - kNN queries (average) 102

List of Abbreviations and Acronyms

AM	Access Method.
bu-Tree	Bottom-up index tree.
CM-tree	Clustered Metric tree.
CoPhIR	Content-based Photo Image Retrieval.
cp-rules	Conditional Preference Rules.
CPU	Central Processing Unit.
DBM-tree	Density-Based Metric tree.
DBMS	Database Management System.
DDSM	Digital Database for Screening Mammography.
DMM	Data Mining Model.
DF-tree	Distance Fields tree.
EGNAT	Evolutionary Geometric Near-neighbor Access tree.
EBNF	Extended Backus-Naur Form.
EXIF	Exchangeable Image File Format.
FQ-tree	Fixed Queries tree.
GH-tree	Generalized Hyperplane tree.
GIS	Geographic Information System.
GNAT	Geometric Near-Neighbor Access Tree.
I/O	Input/Output.
KDD	Knowledge Discovery in Databases.
$k\mathbf{FN}_q$	k-Farthest Neighbor Query.
$k\mathbf{NN}_q$	k-Nearest Neighbor Query.
MAM	Metric Access Method.
MIS	Multimedia Information Systems.
MM-tree	Memory-based Metric tree.
MPEG-7	Moving Picture Experts Group-7.

MSA	Multi-Similarity Algebra.
MVP-tree	Multi-Vantage-Point tree.
PM	Preference Model.
PM-tree	Pivoting M-tree.
RA	Relational Algebra.
RDBMS	Relational Database Management System.
R_q	Range Query.
R_q^{-1}	Reversed Range Query.
SA-tree	Spatial Approximation tree.
SA	Similarity Algebra.
SAME^W	Similarity Algebra for Multimedia Extended with Weights.
SimDB	Similarity-aware Database System.
SIREN	Similarity Retrieval Engine.
SIREN$_{op}$	Similarity Retrieval Engine with Query Optimizer.
SQL	Structured Query Language.
TOR	Total Order Relation.
VP-tree	Vantage-Point tree.
XML	Extensible Markup Language.

GBdI	*Grupo de Bases de Dados e Imagens.*
ICMC	*Instituto de Ciências Matemáticas e de Computação.*
SGBD	*Sistema de Gerenciamento de Bases de Dados.*
uB	*Université de Bourgogne.*
UFU	*Universidade Federal de Uberlândia.*
USP	*Universidade de São Paulo.*

List of Symbols

$\mathbb{A}, \mathbb{A}_h, \mathbb{A}_m$	Traditional (or simple) data domain ($dom(A_h)$).
A, A_h, A_m	Traditional (or simple) attribute defined in a traditional domain $\mathbb{A}, \mathbb{A}_h, \mathbb{A}_m$ ($A \subset \mathbb{A}, A_h \subset \mathbb{A}_h, A_m \subset \mathbb{A}_m$).
a, b, a_h	Constant in the domain of A or the value of another attribute from the same domain $\mathbb{A}$ in the same tuple ($a_h \in \mathbb{A}_h$, $h \in \mathbb{N}^*$).
$Dom^*(A)$	Active domain of attribute A.
$\mathbb{T}$	Data domain.
T, T_i	Relations or datasets ($T, T_i \in \mathbb{T}$, $i \in \mathbb{N}^*$).
$t = \langle a_1, \ldots, a_m, s_1, \ldots, s_p \rangle$	Tuple.
$t_i(A_h)$	The value of the i^{th} tuple on simple attribute A_h ($i \in \mathbb{N}^*$).
$t_i(S_j)$	The value of the i^{th} tuple on complex attribute S_j ($i \in \mathbb{N}^*$).
θ	Traditional comparison operator (exact matching and relational).
$=, \neq$	Exact matching comparison operators.
$<, \leq, >, \geq$	Relational comparison operators.
$\cup$	Union operator.
$\cap$	Intersection operator.
$-$	Difference operator.
π	Projection operator.
$\times$	Cross-product or Cartesian product operator.
$\bowtie$	Join operator.
$\overset{c}{\bowtie}$	θ-join operator.
σ	Selection operator.
c, c_i	Selection condition ($i \in \mathbb{N}^*$).
d	Distance function or metric ($d : \mathbb{S} \times \mathbb{S} \rightarrow \mathbb{R}^+$).
$d(s_i, s_j)$	Distance function or metric ($d : \mathbb{S} \times \mathbb{S} \rightarrow \mathbb{R}^+$, $s_i, s_j \in \mathbb{S}$, $i, j \in \mathbb{N}^*$).
$M =< \mathbb{S}, d >$	Metric space.

$\mathbb{S}, \mathbb{S}_j, \mathbb{S}_p$	Similarity (or complex) data domain ($dom(S_j)$).
S, S_j, S_p	Similarity (or complex) attribute defined in a similarity domain $\mathbb{S}, \mathbb{S}_j, \mathbb{S}_p$ ($S \subset \mathbb{S}, S_j \subset \mathbb{S}_j, S_p \subset \mathbb{S}_p$).
$Dom^*(S)$	Active domain of attribute S.
s_i, s_j	Similarity (or complex) domain elements ($s_i, s_j \in \mathbb{S}$, $i, j \in \mathbb{N}^*$).
s_q	Query element ($s_q \in \mathbb{S}$).
ξ	Similarity threshold.
θ_c	Similarity operator.
σ_c	Similarity selection operator.
k	Number of elements returned by the query ($k \in \mathbb{N}^*$).
L_p	Minkowski distance function family.
L_1	Manhattan distance function.
L_2	Euclidean distance function.
r	Preference Rule.
$I = \{i_1, \ldots . i_n\}$	Set of data items.
X, Y	Itemset ($X, Y \in I$).
$f_{r_1}, \ldots, f_{r_n}$	Image features.
$[l_{1_0} - l_{1_1}], \ldots, [l_{n_0} - l_{n_1}]$	Image feature intervals.
$Class_{R_1}, \ldots, Class_{R_m}$	Image classes.
$max_{feature}$	Maximum number of features extracted from an image.
max_{class}	Maximum number of classes found in a relation.
Ri	Relevant images.
Tsc	Total number of images of the same class.
Ti	Total number of images.

Chapter

1

Introduction

1.1 Motivation

Database Management Systems (DBMSs) were developed to store and retrieve large amounts of data, pursuing efficient query execution and guaranteeing exact answers. The great majority of the current DBMSs are based on the relational technology for data management, which has been developed since the 70's [Codd, 1970]. Usually, these systems support only scalar data domains, such as numbers, dates and short character string. Notwithstanding, with the continuous evolution of the technology, it emerged the need to store what is called **complex data**, such as multimedia (image, audio and video), large texts, multi-dimensional arrays, time series and genomic sequences, and, consequently, organizing them in databases has became an important research target.

When dealing with complex data, comparing elements based on exact matching ($=$ and $\neq$) is not a useful operation, because two exactly equal elements are rare in those domains. Relational comparison, based on the Total Order Relation (TOR), usually expressed by $<$, $\leq$, $>$ and $\geq$, is also not generally applicable.

Often, the retrieval of elements from sets of complex data is based on similarity comparisons, thus complex datasets can be represented in a metric space. Therefore, metric space properties (instead of identity and TOR ones) are employed to build data structures, which are used to speed up query execution. A metric space is a pair $M =< \mathbb{S}, d >$, where $\mathbb{S}$ defines the complex data domain and d is a distance function $d : \mathbb{S} \times \mathbb{S} \rightarrow \mathbb{R}^+$ that satisfies the symmetry, the non-negativity and the triangular inequality properties [Bozkaya and Özsoyoglu, 1999]. A dataset S is in a metric space when $S \subset \mathbb{S}$.

The distance function can be employed to quantify the similarity between two elements, such that two elements are more similar as closer they are from each other, in such a way that a distance equal to zero means that the two elements are the same. Thus, a distance function allows the expression of queries based on predicates that comply with the three properties of metric spaces. Vectorial spaces with any distance function L_p, such as Euclidean (L_2) or Manhattan (L_1) distances, are special cases of metric spaces. Similarity operators can be applied to many complex data types, including spatial data, as long as a distance function that respects the metric properties is properly defined. For instance, Geographic Information Systems (GIS) adopt the Euclidean distance function, so points can be queried by similarity, assuming that closer points are more similar. In database applications, the distance function is usually considered as a "black box", commonly defined by the application domain specialist.

With the rising demand to support multimedia data in relational database management systems (RDBMSs), similarity query operations have attracted increasing interest, specially in applications that require to retrieve complex data by their content. Similarity query operators recover elements that meet a similarity criteria, which are expressed with reference to a data domain element $s_q \in \mathbb{S}$, called the "query center". The two most common similarity queries are the range and the k-nearest neighbor queries [Chakrabarti et al., 2004; Korn et al., 1996]. A Range query (R_q) returns every database element that differs from the query center by at most a given similarity threshold. An example of R_q in a genomic sequence database is: "*Select the DNA sequences that differ from a given sequence p for up to 5 nucleotides*". The k-Nearest Neighbor query ($k\text{NN}_q$) returns the k elements nearest to the query center. An example of $k\text{NN}_q$ query in a genomic sequence database is: *"Select the 3 proteins more similar to the given protein p"*.

Index structures use the properties of the stored data domain to speed up data retrieval in the RDBMS. Several access methods based on index structures have been developed for traditional data, mostly dependent on the TOR property, such as the B-tree and its variations [Zisman, 1993]. For complex data, existing access methods called Metric Access Method (MAM) explore the distance function properties, specially the symmetry and the triangular inequality ones. Examples of index structures able to index metric data in DBMSs and answer similarity queries include the M-tree [Ciaccia et al., 1997], the Slim-tree [Traina Jr. et al., 2002] and the OMNI family methods [Traina Jr. et al., 2007].

Besides access methods based on index structures, the DBMS also use "Query Optimization" techniques to speed up the retrieval operations. Those techniques are based on the maintenance of measurements, the so called "statistics", which are employed in two situations: to estimate the execution cost of different but equivalent algebraic expressions that answer the same query; and to adjust the index method parameters when they are executed to index data or to answer a given query.

Both the query optimization and the access methods based on index structures are employed successfully by DBMSs handling scalar data. However, since similarity search began to be studied more recently than traditional ones, there are fewer appropriate techniques to optimize similarity queries. Moreover, there are few statistics available for metric data that can be stored in a compact way and are generally effective to estimate the cost of queries involving similarity-based predicates and/or MAM parametrization.

Several research works have analyzed how the meaning of "similarity" can be defined in a flexible way, ideally considering also semantic restrictions, such as the identifiable user's interest when posing a query, or particular conditions that restricts the context where each query should be executed or even particular knowledge about the data and it distribution over the metric space. Thus, besides performing query optimization based on algebraic properties of the query operators, it is conceivable to modify a query including the representation of conditions that allow to speed up answer evaluation in situations that are specific to the query environment.

Although some works employ useful parameters to estimate the selectivity of environment conditions and access cost, they always consider the complete database, without taking advantage of significant "local" variations in the data distribution. As accessing metric data tends to be much more expensive than accessing traditional ones, the precise identification of parameters that affect the search in the similarity query region becomes even more important. Moreover, these parameters must consider the DBMS operational environment when a query is executed and the user's interest when the query is posed, allowing integrating new similarity-related query optimization techniques based on algebraic properties into the existing RDBMS. This thesis aims at contributing to further develop the techniques to improve the similarity-based query optimization processing, exploiting both algebraic properties and semantic restrictions as query refinements.

1.2 Problem Definition

There are basically three main techniques to speed up the DBMS similarity query:

(i) metric access methods based on index structures;

(ii) query optimization; and

(iii) semantic restriction occurring in the search space based on local conditions, such as users' interests and stored data distribution at the query time.

An index structure can be developed independently of the system where it will be used. Query optimization is a DBMS technique that aims at finding an adequate execution plan among all the equivalent possibilities that bears the minimum cost. Semantic restrictions are known properties existing among particular subsets of the data domain that either

is effectively stored in the database or meets users' interests. These restrictions embody several assumptions that can be employed as conditions to filter out whole subspaces of the data distribution and thus helps speeding up data retrieval. At least the following three mechanisms and their supporting data structures are required to create a useful operational environment for similarity query processing:

1. **Query rewriting rules -** it is a mechanism that rewrites query expressions into equivalent expressions (or into expressions that can help finding the same answer) but that can be executed in different amounts of time, which is based on rewriting rules derived from the algebra. Every query can be expressed as a combination of operators, and the relational algebra is used to identify equivalent expressions for the same query.

2. **Cost estimation -** it is a mechanism to estimate the execution cost of each alternative way to execute the query. It should be possible to estimate the cost of each operation using each of the available access methods. The cost is evaluated in terms of the execution total time, of the required memory space and of the necessary number of disk accesses, but other factors such as data transmission cost (for a distributed database) or the number of distance calculations (for similarity queries) can be used too.

3. **Selectivity estimation -** it is a mechanism to estimate the selectivity of each condition in selection and join operators. This mechanism is based on a model to predict the resources that each alternative plan needs to evaluate the query, which must require a low amount of memory to store the "statistics" over each attribute involved in the predicates.

These three mechanisms must cooperate among themselves to choose the best access methods, the operations and their best configurations to define what a good (ideally, the best) plan to execute a query is. The alternative plans for each query are generated based on query rewriting rules that rely on algebraic properties to guarantee the equivalence of each generated plan. The cost estimation of each alternative plan is based on the cost estimated for each of its composing algebraic operators. The statistics should be stored in a compact way and must require both a small processing cost, in order to reduce the overhead that they impose during the query processing, and a small amount of valuable memory space to be stored. Moreover, the query structure representation must specify every parameter required to execute the query. Thus, using these mechanisms enables the definition of adequate ways to provide an proper environment to represent the state of the database when the query is executed, the query context, and also the user's interest.

1.3 Work Goals

The work presented in this thesis was developed to create a conceptual foundation to include similarity queries in the relational model, addressing the specification of the user's expectation in these queries. With this purpose, this thesis answers the following main questions:

- How to improve the efficiency and efficacy of similarity queries regarding users' expectations?
- How to improve the execution performance of queries involving similarity search operators?
- How to include similarity-based operators into relational model and how to optimize them?
- How to represent the user's expectation tailored to similarity queries?
- What are the properties of each similarity operator and the rules it meets? Can this similarity operator be employed to optimize a query expression that mixes similarity-based and any other identity- and TOR-based operators?
- How to use the semantic restriction such as users' interests and knowledge mined from complex data to optimize similarity queries?
- How to extend the traditional query optimization architecture to handle also similarity-based operators, semantic restrictions and user's interest requirements?

1.4 Main Contributions

This thesis contributes to the research fields of databases, data mining and users' preferences analysis. Its main contributions can be divided in two categories:

Theoretical:

- The inclusion of similarity operators in the relational model, defining the Similarity Algebra to express similarity queries. This algebra is composed of both equivalence and inclusion-based algebraic rules aiming at optimizing unary similarity operators combined either with other similarity operators or with the traditional ones;
- The adaptation of the RDBMS query rewriting techniques to manage similarity predicates either alone or mixed with traditional ones;

- A definition of a new technique based on semantic restrictions to identify probable regions where the answers of a query should be found, pruning those where answers cannot be found and thus improving query answering efficiency;

Applied techniques:

- The definition of an environment to represent conditions that may indicate the context of a query, the content of a database in the query and the users' preferences as well as a way to speed up query execution and to obtain answers that best meet the user's expectation at the same time, improving the query efficacy;
- The extension of the SIREN (the Similarity Retrieval Engine) [Barioni et al., 2006] to optimize similarity queries meeting user's expectation;
- The extension of the SQL-based SIREN query language to manage users' preferences and data mining processing expressed in a relational, SQL-like language.

1.5 Work Organization

This monograph is organized as follows:

Chapter 1 - Introduction. This chapter describes the motivation, problem definition, objectives and the main contributions of the work developed in this doctorate program.

Chapter 2 - Traditional Query Optimization Process. That chapter presents the main definitions and concepts of traditional queries and their query optimization processes.

Chapter 3 - Similarity Queries. That chapter presents the basic concepts of similarity queries, formalizing the main types of similarity operators (the so called range and k-nearest neighbor queries), and the four approaches found in the literature to treat similarity (the rank, fuzzy, exact and hybrid approaches). Forthwith, the similarity algebra, the query optimization, the query rewriting and the cost and condition selectivity estimation for these four approaches are presented. The concepts of users' preferences and data mining techniques used in this thesis are also presented.

Chapter 4 - A Novel Approach for Similarity Query Optimization Process in DBMSs. That chapter presents the main concepts that allow the inclusion of similarity-based operators into a RDBMS. The extension of traditional query

optimization architecture and similarity algebra are also described in that chapter. Furthermore, semantic restrictions are included in query optimization process to improve the performance of similarity query execution.

Chapter 5 - Similarity Retrieval Engine - Case Study. That chapter presents techniques developed in this thesis in a specific similarity-enabled query interpreter and executor called SIREN, which was extended to include a query optimizer based on the rules developed to help the query rewriting. The SIREN query language extended to handle users' preferences and data mining is also presented. In addition, we show experiments performed over real databases to evaluate every technique developed throughout this thesis.

Chapter 6 - Conclusion. That chapter presents the final considerations, main contributions and suggestions for future works.

Appendix A - The CoPhIR Dataset. That appendix describes the real database employed to exemplify several of the concepts presented in this monograph.

Chapter

2

Traditional Query Optimization Process

2.1 Introduction

Database Management Systems (DBMSs) were developed to store and to retrieve large amounts of data, guaranteeing exact answers and efficient query execution. As the great majority of current DBMSs are based on the relational technology for data management, which has been developed since the 70's [Codd, 1970], the relational model was chose to be the base of our research. In fact, the foundations of that model — namely, set theory and predicate logic — are themselves the base of almost all the other developed since them [Date, 2009]. From the beginning, these systems where conceived to support only scalar data types, such as numerical and short character strings. Such data types rely on two main comparison operator types: exact matching and relational. Exact matching comparison operators ($=$ and $\neq$) can be applied universally for any kind of data types, since it is always possible to decide whether two elements are equal or not. The relational comparison operators ($<$, $\leq$, $>$ and $\geq$) need that the elements are represented in a data domain that meets the Total Order Relation (TOR) property, which allows the comparison of any pair of elements and decide which element precedes or succeeds the other in the pair. The relational and exact matching operators are by far the most common operators found in relational database management systems (RDBMSs). In this thesis, the operators based on exact matching and relational comparisons are called "traditional operators".

RDBMSs use the TOR property existing between elements of scalar data domains to speed up the execution of queries. In this monograph, we use the term "**retrieval operator**" or "**algebraic operator**" to refer to the building blocks of the query execution algorithms over datasets or of the algebraic query representation respectively, whereas we use the term "**comparison operator**" to refer to the operator that performs the

comparison of a pair of elements of a given data domain that are used to implement/specify the comparisons 'inside' a retrieval operator or algebraic operator. In a DBMS, queries are expressed using predicates based on the comparison operators and are executed using retrieval operators. Predicates specify how an algebraic operator sifts data to obtain the results of the query execution. Both unary and binary operations may require conditions. The unary operation based on comparisons is the selection operation, which is represented as $\boxed{\sigma_{(A\,\theta\,a)}\,T}$, where A is an attribute of the relation T defined over a scalar domain $\mathbb{A}$, θ is one of the valid comparison operators in the domain $\mathbb{A}$ of the attribute A, i.e. one of the traditional operators, and 'a' is either a constant (or an expression that returns a constant) taken in the domain of A or the value of another attribute from the same domain $\mathbb{A}$ in the same tuple. Comparison operators are used also in binary retrieval operators, such as the join operation. Unless otherwise stated, the query operators considered in this monograph always refer to the selection operator.

Queries may be answered either evaluating every element in the dataset or not. Index structures, which use properties from the data domain, can be employed to speed up search queries in a DBMS. If there is no index structure, the sequential scan, whose complexity for selection operators is linear regarding time, is the only way to answer a query. Otherwise, if there is an index structure, the number of disk accesses during the query processing can be minimized, resulting in better performance. The applicability of traditional operators to select data allows the development of more efficient indexing techniques.

This chapter presents an overview of the most common access methods used in the RDBMS in Section 2.2. Section 2.3 presents some of the fundamental concepts of the Relational Algebra. The query optimizer based on query rewriting is presented in Section 2.4. The final considerations are presented in Section 2.5

2.2 Access Methods

An access method (AM) is based on a data structure that considers the properties of each data domain to support efficient data access. This structure is the main technique used to reduce the computational cost and to accelerate the search for data in DBMS. Among the most important AM are the B-tree and its variations, and the hash tables [Zisman, 1993].

The B-tree and its B^+-tree variant are the most common AM used in DBMS for being suitable structures for large amounts of data and also for keeping their efficiency in data stored in secondary memory (disk or flash memory) [Graefe, 2011]. B-trees are essentially balanced and multilevel indexes, with graceful growth capabilities. Blocks with x keys and $x + 1$ pointers, $x \in \mathbb{N}^*$, are organized into a tree, whose sorted leaves point to data

records. All blocks are from half to completely full anytime [Garcia-Molina et al., 2000; Graefe, 2011].

A hash table is the AM used only in equality search, because it has a hash function to map (ideally) uniform and randomly search-key values to buckets, without preserve any ordering between indexed elements [Liu and Özsu, 2009]. A hash function h maps each value of the active domain of the attribute A ($Dom^*(A)$) to one bucket. Buckets store data into a memory blocks and possibly by one or more overflow blocks. Thus, the hash table is kept mainly in secondary storage [Cormen et al., 2001; Garcia-Molina et al., 2000].

2.3 The Relational Algebra

The Relational Algebra (RA), proposed by Codd in 1972 [Codd, 1972], is a collection of operations over relations suitable for manipulating data in relational databases. Every operator accepts one or two relations as arguments and returns one relation as the result. This property facilitates expressing complex queries, composing operators using the Boolean operators to create the relational algebra expressions, which can involve as many comparison predicates as required [Ramakrishnan and Gehrke, 2003]. Some operators of the relational algebra are: union ($\cup$), intersection ($\cap$), difference ($-$), selection (σ), projection (π), cross-product or Cartesian-product ($\times$) and join ($\bowtie$). The rename 'pseudo-operator' (ρ) is often employed too.

The selection and the projection operators, both unary operators, manipulate data from a single relation. The selection operator chooses the tuples from the input relation that meets a given selection condition, while the projection picks out some of the relation attributes [Garcia-Molina et al., 2000].

Other operators manipulate data from two relations, and therefore are called binary operators. The union, intersection, difference and cross-product are the standard set operations available in relational algebra. The two relations participating in a union, intersection or difference operations must be union-compatible, i.e., they must have the same number of attributes and each pair of corresponding attributes must have the same domain [Yu and Meng, 2002]. The join operator combines two relations on their common attributes [Maier, 1983].

A relational query expresses "what" the user intends to retrieve, following the relational expression paradigm. The query can be converted into a step-by-step procedure following the imperative paradigm to compute the desired answer, based on the order in which operators are applied in the query. However, usually there are several ways to express the same relational query into a procedural representation, and each way can lead to executions with distinct costs [Ramakrishnan and Gehrke, 2003]. Choosing the

procedure that lead to the fastest, or at least one of the fastest, execution is the objective of the query optimization techniques.

2.4 Query Optimization

The definitions and properties presented in this section, and in Subsections 2.4.1 and 2.4.2 are based on the works of Chaudhuri [1998]; Garcia-Molina et al. [2000]; Ioannidis [1996]; Yu and Meng [2002] and Ramakrishnan and Gehrke [2003]. For illustration of the concepts presented in this section and in Subsections 2.4.1 and 2.4.2, we use a subset of the Content-based Photo Image Retrieval[1] (CoPhIR) database. A detailed description of the dataset is presented in Appendix A. The relational schema used in the examples is shown in Example 2.1. The primary key attributes are underlined. In this example, the traditional attributes are colored in red, while the complex attributes are colored blue. The existing RDBMSs usually do not support complex attribute. As we will use this same relation scheme to illustrate our approach to include complex attributes and similarity queries over them, we are showing the complete scheme from the beginning, but they will be fully explained only in Chapter 4.

Example 2.1:

$$\texttt{CoPhIRdb} = \{\underline{\texttt{UserId}}, \underline{\texttt{PhotoId}}, \texttt{Title}, \texttt{Description}, \texttt{Tags}, \texttt{Lat}, \texttt{Long}, \texttt{Country}, \texttt{Image}, \texttt{Coordinate}\}$$

User queries written in Structured Query Language (SQL) are received by a RDBMS, translated into relational algebra expressions and presented to the query optimizer, which uses information about how the data are stored to generate efficient execution plans to evaluate these queries. An execution plan is a tree with relational operators at the intermediate nodes and relations at the leaf nodes, defining a sequence of steps for query evaluation. Each step in the plan corresponds to one relational operation in the logical query plan, and to one relational operation plus the access method to be used for the operation evaluation in the physical query plan.

For example, suppose that a user wants to know other users that have photos of beaches from the tropical climate. The user can write the SQL Query **Q**1, shown in Example 2.2, and send it to a RDBMS.

Example 2.2:

Q1: "Select the users that have photos with tags '*beach* and *sea*' and whose photos were taken on tropical climate beaches".

[1] CoPhIR website. Available at: `http://cophir.isti.cnr.it/`. Accessed in: July 02, 2012.

```
SELECT DISTINCT Codb1.UserId
FROM CoPhIRdb Codb1,
     (SELECT Codb.UserId, Codb.PhotoId
      FROM CoPhIRdb Codb
      WHERE UPPER(Codb.Tags) LIKE '%BEACH%'
      AND Codb.Lat BETWEEN -23.43 AND +23.43) Codb2
WHERE UPPER(Codb1.Tags) LIKE '%SEA%'
AND Codb1.UserId = Codb2.UserId
AND Codb1.PhotoId = Codb2.PhotoId
```

When the Query **Q**1 is received by a RDBMS, it is compiled, optimized and then executed. After the Query Compiler finalizes its analysis, Query **Q**1 is parsed into an expression tree following the relational algebra, which is the logical query plan (presented in Figure 2.1(a)) represented in a 'canonical' format, and submitted to the Query Optimizer. The Query Optimizer receives this 'canonical plan' as input. A logical query plan uses algebraic operators to represent the query. Equivalent plans can be obtained following equivalence rules that can be used to rewrite a plan in different but equivalent ways.

Thereafter, the logical plan is converted into a physical query plan, which is a sequence of operations that can be implemented by the query evaluation engine. A physical query plan uses retrieval operators to represent the query. A physical query plan is obtained exchanging each algebraic operator (or a sequence of algebraic operators) by a retrieval operator that executes the intended action of the exchanged algebraic operator(s) over a dataset stored in a RDBMS. The physical query plan also indicates the access method that must be employed to access each relation involved in the query and selects an execution alternative for each of the algebra operations exchanged from the logical plan. Figure 2.1(b) shows the logical plan transformed into a physical plan that uses the '`table-scan`' and the '`nested-loop join`' physical operators. The `table-scan` reads the entire relation corresponding to the `FROM` subquery and filters out the tuples according to its selection conditions. Then, the result of this subquery is joined with selected tuples of the inner relation by the join operator (right child), using the `nested-loop join` physical operator.

For the same query, there are different equivalent execution plans that produce the same result. However, different equivalent plans are usually evaluated with different costs. The goal of the query optimization is to find an execution plan, among all possible equivalent plans, that has the best performance, i.e., that can be evaluated with the minimum cost, the so called 'optimal plan'. Indeed, the time required to find "the best"

plan is usually rather large, so real optimizers strive to find a "good enough" plan with in an acceptable delay.

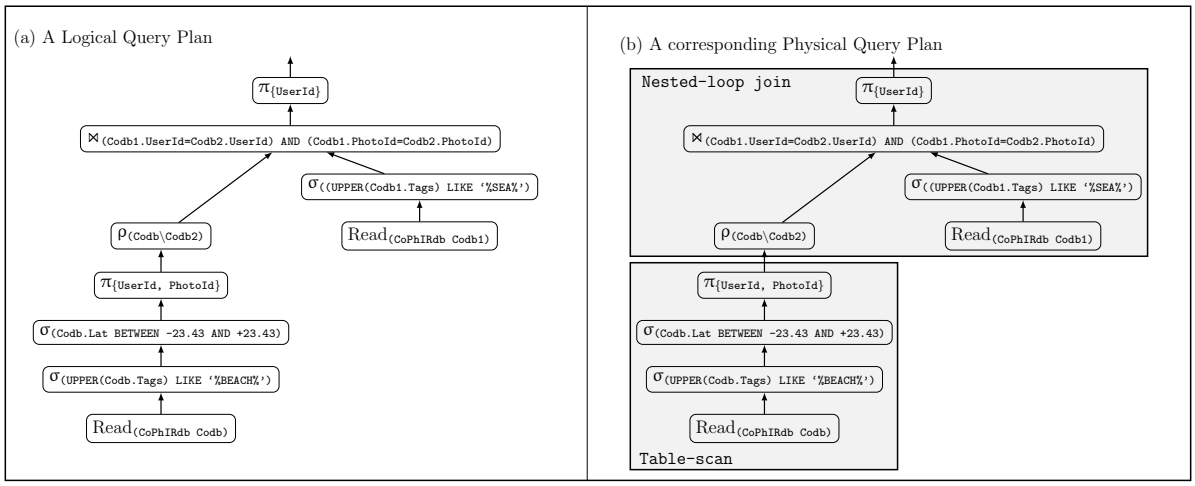


Figure 2.1: Example of a (a) logical and a (b) physical query plans, represented as query trees for Query **Q**1.

The query optimization architecture adopted in this monograph is based on query rewriting, as presented in Figure 2.2. This type of optimization is based on the fact that several algebraic expressions have equivalent results, but each expression has a different execution cost. The goal of this kind of query optimization is identifying an equivalent algebraic expression that may be evaluated with low computational cost. According to Ioannidis [1996] and Garcia-Molina et al. [2000], this kind of query optimizer can be divided in two parts: the Logical Query Plan Generator and the Physical Query Plan Generator.

The Logical Query Plan Generator is responsible for applying transformations to a given query represented as an algebraic expression and for producing equivalent queries intended to be executed in a more efficient way. For this intend, it uses the Algebraic Space Enumerator and the Method-Structure Space modules. The Algebraic Space Enumerator module determines the ordering of the necessary operators considered in each query. The Method-Structure Space module selects the existing implementation choices for the execution of each operator ordering specified by the Algebraic Space Enumerator. In sum, the Logical Query Plan Generator receives the canonical plan as input and uses the algebraic transformations generated by the Algebraic Space Enumerator module to transform the logical query plan into a better one. Thereafter, the Method-Structure Space module is used to lead the conversion of the logical query plan into an efficient physical plan. The physical query plan fond to be the better one is the input to the Physical Query Plan Generator.

The Physical Query Plan Generator pursues a search strategy exploring the space of execution plans determined by the Algebraic Space Enumerator and the Method-Structure Space modules for each query produced by the Logical Query Plan Generator. It

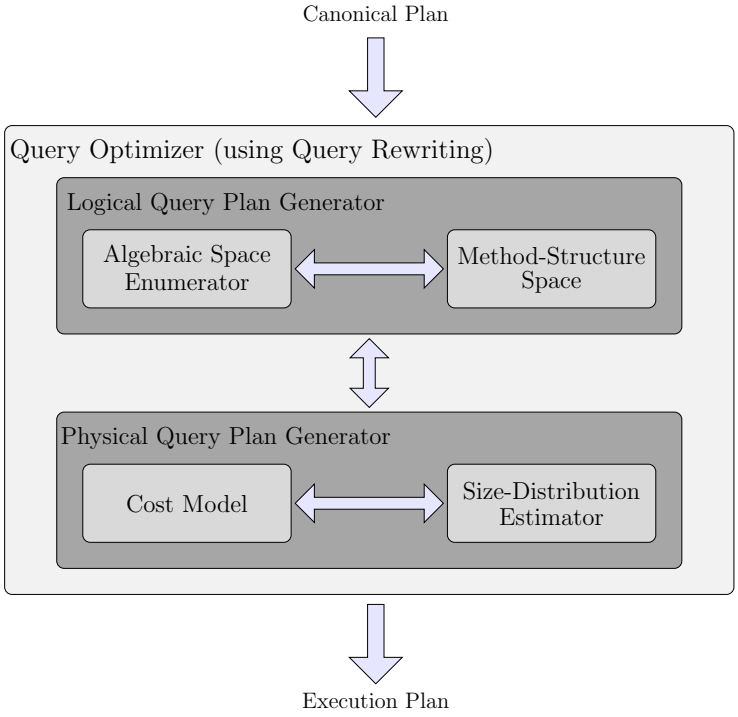


Figure 2.2: Query optimizer architecture according to Ioannidis [1996] and Garcia-Molina et al. [2000].

compares plans based on each plan cost estimate generated by the Cost Model and the Size-Distribution Estimator modules, selecting the cheapest one to be used to generate the answer for the original query. The Cost Model module specifies the arithmetic formulas employed to estimate the cost of the execution plan. The Size-Distribution Estimator module estimates the sizes of the queries result (or of the subqueries) and the frequency distributions of values assumed by the attribute ("statistics"), which are needed for the Cost Model. Once the best plan is chosen, the 'execution plan' is submitted to the execution by the Query Executer and the answer of the query is sent back to the user.

Optimizing a SQL query converted to a relational algebra expression involves two steps: enumerating the alternative plans available to evaluate the expression, and estimating the cost of each enumerated plan, choosing the plan with the least estimated cost. The operation ordering has a significant impact on the cost of query execution. There are two main techniques to determine the "best" ordering for query execution: the algebraic-based optimization technique, which uses a set of heuristic rules to guide the transformation from one execution plan to another; and the cost estimation-based optimization technique, which estimates the cost of every possible execution plan for each query and chooses the execution plan with the lowest estimated cost. Both techniques use a set of rules that can transform an execution plan into another, represented as a relational algebra expression. The algebraic-based optimization technique, or query rewriting, is described in Subsection 2.4.1 and the cost estimation-based technique, or cost and condition selectivity model, is described in Subsection 2.4.2.

2.4.1 Query Rewriting

The basic idea of the algebra-based optimization technique is first to represent each relational query as a relational algebra expression and then to transform it into an equivalent but more efficient relational algebra expression. In the literature, there are several algebraic laws that can transform a relational algebra expression into another equivalent one. Two relational algebra expressions are said to be equivalent if they produce the same result over any instance of the input relations. Hence, several equivalence expressions allow modifying a relational algebra expression to obtain an expression with a cheaper plan. The existence of equivalent expressions implies a choice of evaluation strategies.

The algebraic laws most commonly used in query optimization are described below. Let T_1, T_2 and T_3 be three relations, then:

- Cascade of selections: let c_1 and c_2 be two selection conditions on T_1, then

$$\sigma_{(c_1 \text{ and } c_2)} T_1 = (\sigma_{c_1} T_1) \cap (\sigma_{c_2} T_1) = \sigma_{c_1}(\sigma_{c_2} T_1) = \sigma_{c_2}(\sigma_{c_1} T_1) \ . \tag{2.1}$$

- Commuting selection with join: if condition c involves attributes of only T_1, then:

$$\sigma_c(T_1 \bowtie T_2) = (\sigma_c T_1) \bowtie T_2 \ , \tag{2.2}$$

  if c_1 only involves attributes from T_1 and condition c_2 only involves attributes from T_2, then:

$$\sigma_{(c_1 \text{ and } c_2)}(T_1 \bowtie T_2) = (\sigma_{c_1} T_1) \bowtie (\sigma_{c_2} T_2) \ . \tag{2.3}$$

  The commuting selection is also applied to the cross-product of relations T_1 and T_2.

- Associativity of θ-join and natural join: the θ-join and natural join operations can not be mixed in the same rule, because they yield an incorrect result, that is, $T_1 \overset{c_1}{\bowtie} (T_2 \bowtie T_3) \neq (T_1 \overset{c_1}{\bowtie} T_2) \bowtie T_3$. Nevertheless,

$$T_1 \overset{c_1}{\bowtie} (T_2 \overset{c_2}{\bowtie} T_3) = (T_1 \overset{c_1}{\bowtie} T_2) \overset{c_2}{\bowtie} T_3 \ , \tag{2.4}$$

  provided that c_1 involves only attributes from T_1 and T_2, and c_2 involves only attributes from T_2 and T_3;

$$T_1 \bowtie (T_2 \bowtie T_3) = (T_1 \bowtie T_2) \bowtie T_3 \ . \tag{2.5}$$

- Replacing $\times$ and σ by $\bowtie$: if c is a selection condition of the form $\boxed{T_1.a\ \theta\ T_2.b}$ or the conjunction of terms following this same format, and it is preceded by a

cross-product operation, then:

$$\sigma_c(T_1 \times T_2) = (T_1 \overset{c}{\bowtie} T_2) \ . \tag{2.6}$$

The transformation of equivalent relational algebra expressions is guided by heuristics optimization laws. The following four rules are commonly used:

1. Perform selections as early as possible, because they often can substantially reduce the size of the relations. As a result, if they are performed early, later operations such as joins can be evaluated more efficiently, processing a reduced input.

2. Replace cross-products by joins whenever possible, because a cross-product is typically much more expensive than a join.

3. If there are several joins, perform the most restrictive joins first. A join is more restrictive than another if it yields a smaller result. Finding which join is the most restrictive is based on selectivities and other statistical information.

4. Project out useless attributes early, so smaller input relations can be used in the next operations.

Those heuristic optimization rules can be represented graphically using the concept of a query tree. In the query tree, each input relation is the leaf node and each operation is represented as an internal node. The operation in a higher node can be evaluated only if all of its descendant operations have been evaluated.

2.4.2 Cost and Condition Selectivity Model

The objective of the cost estimation-based optimization techniques is to choose, among all possible execution plans, the one that has the lowest estimated cost. The estimation of the query evaluation cost in a RDBMS is the sum of two components: the cost to access secondary memory (the input/output - I/O cost) and the computational cost (use of the central processing unit - CPU). The I/O cost is derived by the data transfer between the main memory and the secondary storage, which can be computed by the number of page reads and writes. The CPU cost is determined by the execution of the operations over data stored in main memory. For most database operations, including selection, projection and join operations, the I/O cost is the dominant. Therefore, several access methods, such as B^+-trees, are employed to reduce the I/O cost.

This optimization technique works as follows: for each query, enumerate all possible (or worth considering) execution plans; for each plan, estimate its cost; and choose the one with the lowest estimated cost. If every execution plan cost can be estimated accurately, then an optimal plan can be found. However, there are two difficulties to use this technique:

1. The number of possible execution plans is an exponential function of the number of relations referenced in a query;

2. An accurate cost estimation for the execution plan may be difficult to obtain, because it is necessary to correctly estimate the intermediate result sizes.

The first difficulty is tackled using heuristics to enumerate only a subset of all possible execution plans, instead of all possible plans. On the other hand, the second difficulty demands using information about the stored data, the so called "statistics", to estimate the selectivity and cost of the several conditions used in the query. The most studied methods can be roughly classified into three categories, as follows: (1) histogram-based methods, which use pre-stored detailed statistics about relations to estimate the sizes of intermediate results; (2) sampling methods, which estimate the sizes of intermediate results based on the information collected from a small fraction of current data stored in the relations; and (3) parametric methods, which use analytical and/or statistical techniques to estimate the size of intermediate results, making assumptions about the distribution of data values (e.g. uniform distribution) and about the correlation between the values of different attributes (e.g. independent attributes).

A fundamental property of a database system is that it maintains a description of all the data that it contains. This information is stored in a collection of relations, maintained by the system, called the 'system catalog'. Statistics (cardinality, size, etc.) about relations and indexes are stored in the system catalog and updated periodically. The catalog also contains information about users, such as the accounting and authorization information.

Information stored in the system catalog are used by the Query Optimizer to estimate plan costs. At the beginning of the evaluation, the operands are the existing data structures of known sizes, such as relations, available indexes and number of pages. However, in later stages, as most operands have been results of preceding operations, the cost model must estimate their sizes using information about the original data structure and the selectivity of operations already performed on them [Jarke and Koch, 1984]. The selectivity corresponds to the expected fraction of tuples that will satisfy the condition [Selinger et al., 1979]. The most selective operation is the one that retrieves the fewest pages, and using it tends to minimize the data retrieval cost.

There are some techniques to estimate the result size of relational operations. For example, in the selection operation, the key is to have an accurate estimation of the selectivity. When the selection condition c is of the form $\boxed{c = A\ \theta\ a}$, the selectivity depends largely on the distribution of the values of A in T, and sometimes from the characteristics of the attribute A; when c is a conjunction or disjunction of several simple conditions, then the selectivity also depends on the dependencies among the involved attributes.

2.5 Final Comments

This chapter presented an overview of the traditional query processing performed by the relational database management systems. It was shown that every SQL query is received by the DBMS, translated into a tree using relational algebra operators and presented to the query optimizer as a logical query plan.

In the query optimizer, this tree is submitted to the Logical Query Plan Generator, which produces several algebraic equivalent plans. The equivalent plans are analyzed by the Physical Query Plan Generator, where their cost are evaluated and compared. The goal is to identify an algebraic equivalent plan that may be evaluated with low computational cost. The (ideally) best logical plan is transformed into a physical plan exchanging the algebraic operators (or sequence of operators) by retrieval operators that implements the corresponding functionality. The algebraic laws and heuristics presented in Subsection 2.4.1, and the cost estimation techniques present in Subsection 2.4.2 are used, respectively, to generate equivalent plans and to choose the better (cheaper) physical plan to be executed.

The query optimization process for traditional data is well consolidated, although there is not a research consensus on its modules and level details. However, to support complex data in a RDBMS, the query optimizer should be able to rewrite similarity queries and to estimate their cost. In Chapter 3 we present some related work on similarity query optimization.

Chapter

3

Similarity Queries

3.1 Introduction

In contrast with the traditional queries, which use exact matching and relational operators to manipulate scalar data in relational database management systems (RDBMSs), similarity queries search for elements that are more "similar to" or "distinct from" a given query element, following some similarity condition. In other words, similarity queries compare every element of a set with a query element and select those that meet the similarity criterion.

Similarity between two elements is defined based on a distance function d [Wang and Shasha, 1990]. The distance function d calculates the distance between two elements and returns a real non-negative value, which assesses the dissimilarity degree between them. The distance function returns values near to zero for element pairs more similar, and returns larger values when comparing two elements rather dissimilar [Braunmüller et al., 2000]. The distance function d , also called a metric, is the basis to create a metric space $M =< \mathbb{S}, d >$, where $\mathbb{S}$ denotes the universe of valid elements (i.e. the complex data domain) and d is a function $d : \mathbb{S} \times \mathbb{S} \rightarrow \mathbb{R}^+$ that expresses the "distance" between two elements of $\mathbb{S}$. The metric d must satisfy the following properties [Lima, 1993]:

- Symmetry: $d(s_1, s_2) = d(s_2, s_1)$;
- Non-negativity: $0 < d(s_1, s_2) < \infty$ if $s_1 \neq s_2$ and $d(s_1, s_1) = 0$;
- Triangular inequality: $d(s_1, s_2) \leq d(s_1, s_3) + d(s_3, s_2)$, $\forall\, s_1, s_2, s_3 \in \mathbb{S}$.

A metric dataset $S \in \mathbb{S}$ is the set of elements from the domain $\mathbb{S}$ stored in a database [Braunmüller et al., 2000]. Similarity queries are the most important queries

to retrieve data from metric datasets. A similarity query should find efficient ways to locate user-relevant information in a collection of elements whose similarity has been quantified using a pairwise metric between element instances [Zezula et al., 2006]. The basic similarity queries retrieve elements in a metric dataset that meet the similarity predicate comparing a given element provided as a query parameter called "query center" using the distance function of the corresponding metric space to evaluate the predicate.

The majority of the literature regarding search techniques in metric spaces focuses in queries over a single metric dataset, without considering that the metric data is related to other data that often are represented in scalar domains. In this monograph, we assume that a metric dataset is the active domain $Dom^*(S)$ of an attribute S of a relation T in a relational database, that is, we assume that the metric dataset S is the set of values existing in attribute, which is indistinct from attribute S. Therefore, each element $s_i \in S$ is the value of attribute S in a tuple in relation T, and in this way each element s_i is associated to the values of the other attributes of T in the same tuples. When the user poses queries over T, some of the predicates could be similarity ones over attribute S, whereas others can be identity or TOR-based over any attribute of T, and the query will therefore be composed of identity, relational and similarity criteria.

Just like the traditional ones, the similarity-based criteria employ comparison operators called **similarity comparison operators** or just similarity predicates. When a relational selection operator employs a similarity comparison operator to filter the input dataset, it is called a "similarity selection operator". The similarity selection is the fundamental similarity operator to perform queries over similarity datasets. The syntax to express similarity selections follows the same format of the traditional ones: $\boxed{\sigma_{c\,(S\;\theta_c\;s_q)}\,T}$, where σ_c represents a similarity selection, S is a metric attribute defined in relation T whose values are taken from the metric domain $\mathbb{S}$, θ_c is a similarity operator valid in the domain $\mathbb{S}$ of the attribute S and $s_q \in \mathbb{S}$ is a query element, which can be either a constant (or an expression that returns a constant) or the value of another attribute in the same tuple of the relation T, which is also taken from the same domain $\mathbb{S}$. In this monograph, we always assume that every "complex" dataset is in a metric domain, thus a complex attribute is also a metric attribute, and we use the words "complex attribute" and "metric attribute" interchangeably.

There are two similarity operators commonly employed: the range and the k-nearest neighbor ones. They are defined as follows.

Range Query - R_q: Given a query center $s_q \in \mathbb{S}$ and a similarity threshold ξ, the range query returns all elements that differ from the query center s_q at most the similarity threshold. Figure 3.1(a) shows an example of a range query in a bi-dimensional space with the Euclidean distance L_2 and the threshold ξ shown. The blue elements (inside the circumference) belong to the answer dataset.

k-Nearest Neighbor Query - $k\text{NN}_q$: Given a query center $s_q \in \mathbb{S}$ and a similarity threshold $k \in \mathbb{N}^*$, the k-nearest neighbor query returns the k elements nearest to the query center s_q. Figure 3.1(b) illustrates an example of a k-nearest neighbor query in a bi-dimensional space with Euclidean distance L_2 and $k = 3$. The blue elements connected to the query center s_q belong to the answer dataset.

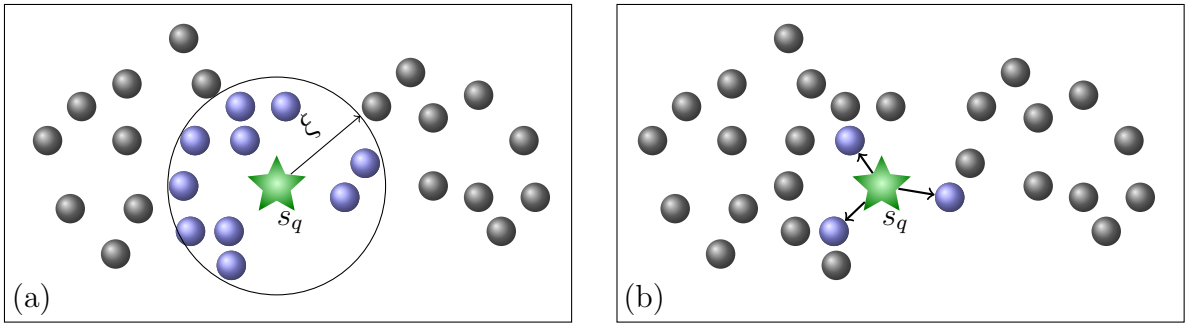


Figure 3.1: Examples of similarity queries in bidimensional space with Euclidean distance L_2: (a) Range query – R_q, and (b) k-nearest neighbor query – $k\text{NN}_q$, with $k = 3$ elements.

To evaluate a similarity query using a sequential scanning, every dataset element must be compared to the query center. Index structures are employed to accelerate that processing, pruning regions of the space where answers surely can not be found. Although the sequential scan can always be used, even when there is no data index structure, this strategy is not adequate for large datasets due to the high computational costs involved. On the other hand, if there is a data index structure, the number of comparisons (number of distance calculations) and the number of disk accesses during the query processing are reduced, leading to a better performance. However, the index structures usually available in the DBMS to execute queries using comparisons over scalar data can not be used to retrieve complex data. Metric access methods (MAMs) are the most adequate index structure for datasets represented just by the elements and by the distances between them [Traina and Traina Jr., 2003]. An overview of MAMs is presented in Section 3.2.

Similarity queries are found and discussed in literature following four approaches: rank, fuzzy, exact and hybrid, which are summarized in Section 3.3. Sections 3.4, 3.5, 3.6 and 3.7 present brief overviews, respectively, of the similarity algebra, the query optimization process, the query rewriting techniques and the cost and selectivity estimation of query condition, when querying complex data based on these four approaches. Sections 3.8 and 3.9 discuss techniques existing to process the user's preference and to explore data mining tasks over complex data, respectively. Finally, Section 3.10 presents some comments about this chapter material.

3.2 Metric Access Methods

Metric access methods (MAMs) are based on index structures that organize the elements in a stored dataset using only a distance function that satisfies the three properties of distance functions, namely symmetry, non-negativity and triangular inequality. These methods are of particular importance to index complex data, usually organizing them as a tree, as the usual total ordering relationship among the elements does not apply. The objective of a MAM is to minimize the number of comparisons (distance functions calculations) and the number of disk accesses during the query processing [Traina and Traina Jr., 2003].

There are several research works aiming at answering similarity queries efficiently. The main MAMs found in the literature are briefly presented here. Detailed and comprehensive surveys about MAMs can be found in Chávez et al. [2001], Hjaltason and Samet [2003] and Zezula et al. [2006].

The paper of Burkhard and Keller [1973] is the landmark in development involving data index in metric domains. It describes three techniques for recursive partitioning of a metric space that allow the creation of MAMs, which are materialized as trees. The first technique partitions a dataset by choosing a representative for subsets of elements that are close to each other and grouping them based on their distances to the representative. The second technique divides the original set into a fixed number of subsets and chooses a representative to each subset. Each representative and the maximum distance from the representative to some element are also maintained in the structure to improve similarity queries. The third is similar to the second technique, with the additional requirement that the maximum distance between any two elements in the same subset is not greater than a given constant c. This constant can be distinct for each level of the structure and its value guarantees that every element is in at least one of the subsets in that level. In the three techniques, the representatives are used to prune elements and subtrees during a query.

Following the techniques presented in Burkhard and Keller [1973], several MAMs were proposed, such as: the Generalized Hyperplane tree (GH-tree) [Uhlmann, 1991], the Ball Decomposition [Uhlmann, 1991], the Vantage-Point tree (VP-tree) [Yianilos, 1993], the Fixed Queries tree (FQ-tree) [Baeza-Yates et al., 1994], the Geometric Near-Neighbor Access Tree (GNAT) [Brin, 1995], the Multi-Vantage-Point tree (MVP-tree) [Bozkaya and Özsoyoglu, 1997, 1999], the Spatial Approximation tree (SA-tree) [Navarro, 1999, 2002] and the bottom-up index tree (bu-tree) [Liu et al., 2006]. However, all of these MAMs are considered static access methods, because they require to have the full dataset already available during the index creation process, and they do not support further insertions and deletions after the tree creation.

The first dynamic MAM presented in the literature was the M-tree [Ciaccia et al., 1997]. It is a height-balanced tree that stores the data in the leaf nodes. However, it often produces trees where the nodes largely overlap each other in the same level, drastically reducing the pruning ability of the query algorithms. The Slim-tree [Traina Jr. et al., 2000b] is an evolution of the M-tree that presented the first technique able to measure and reduce overlaps between subtrees that work in a metric space. Other examples of dynamic MAM are the Omni-family [Santos Filho et al., 2001], the Distance Fields tree (DF-tree) [Traina Jr. et al., 2002], the Density-balanced Metric tree (DBM-tree) [Vieira et al., 2010, 2004], the Pivoting M-tree (PM-tree) [Skopal et al., 2004], the Evolutionary Geometric Near-neighbor Access Tree (EGNAT) [Navarro and Paredes, 2011; Paredes and Navarro, 2009; Paredes et al., 2006], the DBM*-tree [Ocsa and Cuadros-Vargas, 2007], the MM Metric tree (MM-tree) [Pola et al., 2007], the Clustered Metric tree (CM-tree) [Aronovich and Spiegler, 2007] and the Onion-tree [Carélo et al., 2009, 2011].

3.3 Searching for Similarity

Usually, Multimedia Information Systems (MIS) treat similarity using four different approaches: the rank, fuzzy, exact and hybrid approaches.

The rank approach is based on the establishing an ordering among the stored tuples or elements. Ranking queries (or top-k queries) aim at providing only the top k results, according to a user-specified ranking criterion. The answer of a top-k selection query is an ordered set of tuples, where the ordering criterion is how well each tuple matches the given query. It is true that this approach is consistent to the relational model and can be applied to similarity queries considering the distance functions as the ranking criterion, but it depends on the existence of a ranking criterion that is independent from the queries. This requirement departs from the fact that the ranking criterion of a similarity query depends on each query, that is, the ranking criterion varies with the query.

The fuzzy approach associates similarity to an uncertainty, or imprecision grade, to every comparison evaluation between a pair of elements of the dataset, often providing fuzzy logic-based methods to solve queries. The problem of this approach is that it assumes that although complex data manipulation involves similarity evaluation, this does not mean that the similarity evaluation is uncertain or imprecise (as only exact match comparisons are useless in these domains). In fact, it is possible to execute similarity queries resulting in either approximated or exact answers. The fuzzy approach aims at obtaining results where there is not exact definition for how the intended results are obtained but rather only a fuzzy definition exists about what is intended.

The hybrid approach mixes the rank and the fuzzy approaches. Therefore, the resultant ordering of the elements depends of a final ranking condition, where each element matches the fuzzy condition to a different degree. The problem of this approach is that it

assumes the global ranking criterion to evaluate fuzzy conditions, therefore it yet requires a well-defined ranking criterion, but the results are harder to evaluate, as several sequences must be compared and thus a sequence metric need also to be defined.

Finally, the exact approach evaluates each element according to how well it fits a similarity criteria given for the query. The similarity varies for each query, because the similarity is always evaluated regarding elements that are specified in the query, not to a global ranking criterion, as in the rank approach. Although this approach is the most expensive ones, it always retrieve the complete answer (considering the similarity criterion), thus the correct answer is always obtained. As our goal aims at improving the query answer, speeding up the exact approach may lead to a technique that both provides good answers and do it in acceptable times. Thus, we target the exact approach to develop the research presented in this monograph.

Works on similarity algebra, query optimization, query rewriting and cost and condition selectivity model following these four approaches are presented in Sections 3.4, 3.5, 3.6 and 3.7, respectively.

3.4 Similarity Algebra

There are several extensions in the literature to the relational algebra aimed at including similarity functionality in RDBMSs, each following varying perspectives. The first algebra to consider this issue was the Multi-Similarity Algebra (MSA), presented in Adali et al. [1998]. It has been designed to integrate multiple similarity measures coming from several similarity assumptions, which use the notion of similarity ranking to return the search elements, in a common framework. However, MSA is defined at a high abstraction level and does not address the problem of an "operational" algebra usable for modeling, optimizing and processing queries with similarity-based operations [Atnafu et al., 2004]. Therefore, it is not fully consistent with the relational model.

Following the same perspective, i.e. the rank approach, Adali et al. [2004] introduced another algebra for querying ranked relations and proved various coherence preservation properties for that algebra, which shows when different rank columns are guaranteed to induce the same ordering among tuples in the answer, what can be advantageous to produce approximate early returns.

In another paper, Li et al. [2005] extended the relational algebra into a "rank-relational algebra". It captures the ranking property introducing a rank operator and extending other relational operators to support ranking as a first-class concept in the algebra. According to the authors, the relations, operators and algebraic laws respect and take advantage of the notion of ranking. Following the same approach, Adali et al. [2007] also presented an algebra that treats ranks and the element ordering imposed by ranks as first-class elements, but aims at supporting complex mining and data fusion tasks.

Following the fuzzy approach, several fuzzy relational algebras have been proposed in the literature. One of the firsts was the paper of Montesi and Trombetta [1999], which extended the classical relational algebra including new operators (top and ϵ-similar operators) and user preferences (weights) to formulate queries that take into account the similarity of elements represented in the fuzzy relational model.

The Similarity Algebra for Multimedia Extended with Weights (SAME^W) [Ciaccia et al., 2000; Penzo, 2005] generalizes the relational algebra to allow the formulation of similarity queries over multimedia databases, introducing two new operators, called *cut* and *top*, that are useful for range and kNN queries, respectively. They work providing a criteria that respectively limits the answer's cardinality, and discards tuples whose similarity degree is lower than a specified threshold. This algebra also incorporates weights to assign relevance to the user preferences.

Picariello and Sapino [2002] developed a fuzzy model for image datasets, providing an algebra for dealing with fuzziness at the attribute level of features extracted from the images.

In another paper, Montesi et al. [2003] proposed a fuzzy-based algebra to represent the imprecision related to several kinds of Web and multimedia data. The proposed fuzzy algebra extends the classical relational algebra to be applicable to fuzzy relations using the new operators *top* and *cut*. Both algebras allow taking into account user's preferences in the form of weights that can be attached to predicates and operators. This model allows the representation of imprecision at the attribute as well as at the tuple level.

Schmitt and Schulz [2004] introduced the similarity calculus and a similarity algebra (SA), bringing vagueness, weighting and user preferences to the traditional relational calculus and algebra, respectively. The authors show also how to map similarity calculus expressions into a corresponding similarity algebra ones, which is adequate for efficient query processing.

For the hybrid approach, Belohlavek et al. [2007] presented an extension of relational algebra by adding the concept of similarity to ranked tables, which essentially corresponds to a kind of fuzzy sets. In that paper, the notion of rank is used to sort truth degrees represented by a fuzzy logic [Belohlavek et al., 2011; Belohlavek and Vychodil, 2009, 2010].

In the case of the exact approach, Atnafu et al. [2001] defined a well-formalized multimedia content-based algebra, useful for modeling, optimizing and processing of multimedia queries. The authors also introduced a similarity-based algebra that formalizes the search operations over images stored in multimedia database systems, defining new operators, such as the "multimedia content-based join", which can be used to perform operations either isolated or together with other relational operators.

Silva et al. [2009] and Silva et al. [2010b] presented, respectively, multiple equivalence rules just for both similarity aggregations and similarity join operators.

Traina Jr. et al. [2006] introduced a relational algebra extension considering complex similarity queries composed of two or more similarity predicates combined through Boolean operators. However, the rules derived are able only to handle queries centered at the same query center (a single center), which is restrictive and does not cover all cases occurring in RDBMSs. In this monograph, we take the algebra proposed by Traina Jr. et al. [2006] as a start point and we generalize it to allow handling queries centered either at the same or at distinct query centers. Also, we present the fundamental properties that allow the integration of the unary similarity operators into the Relational Algebra, handling similarity queries either alone or mixed with the traditional operators. These properties are presented in Section 4.5.

Figure 3.2 presents the time line of the similarity algebra literature papers, following the four approaches employed for similarity in MIS.

3.5 Query Optimization

Similarity query optimization began to receive more attention since the paper of Adali et al. [1998] on the Multi-Similarity Algebra, which developed query optimization techniques to reduce the cost of query processing, by pushing selections down and reordering costly joins, following the rank approach. As the similarity operators are among the most costly retrieval operations, pushing down the similarity selections allows employing access methods designed to handle then independently of taking into consideration the other operators, which also permits to reduce the search space at a great extent. The authors provided a set of equivalence rules between expressions in MSA, allowing rewrite queries represented using its operators.

The paper of Chang and Hwang [2002] treated the query optimization based on rank using expensive predicates. Also, Chaudhuri et al. [2004] investigated how to optimize the processing of top-k selections queries over multimedia repositories using a cost-based approach. In its turn, Li et al. [2005] extended bottom-up query optimizers, such as the System-R optimizer, incorporating ranking. The authors used algebraic laws based on a rank-relational algebra to define equivalent plans in the search space handled by the query optimizers. Finally, Schnaitter et al. [2009] introduced the Deep estimation framework, which enables a systematic estimation methodology that takes directly into account the distribution of scores and values in the underlying data, to approximate the number of input tuples that an operator must access following the physical plan using rank join operators.

For the fuzzy approach, Montesi et al. [2003] presented a query optimizer that uses the standard heuristic that guides RDBMSs to find equivalent queries that minimize the cardinality of intermediate results, minimizing the number of I/O operations. The authors used equivalence and containment rules to choose, among the equivalent query plans of a

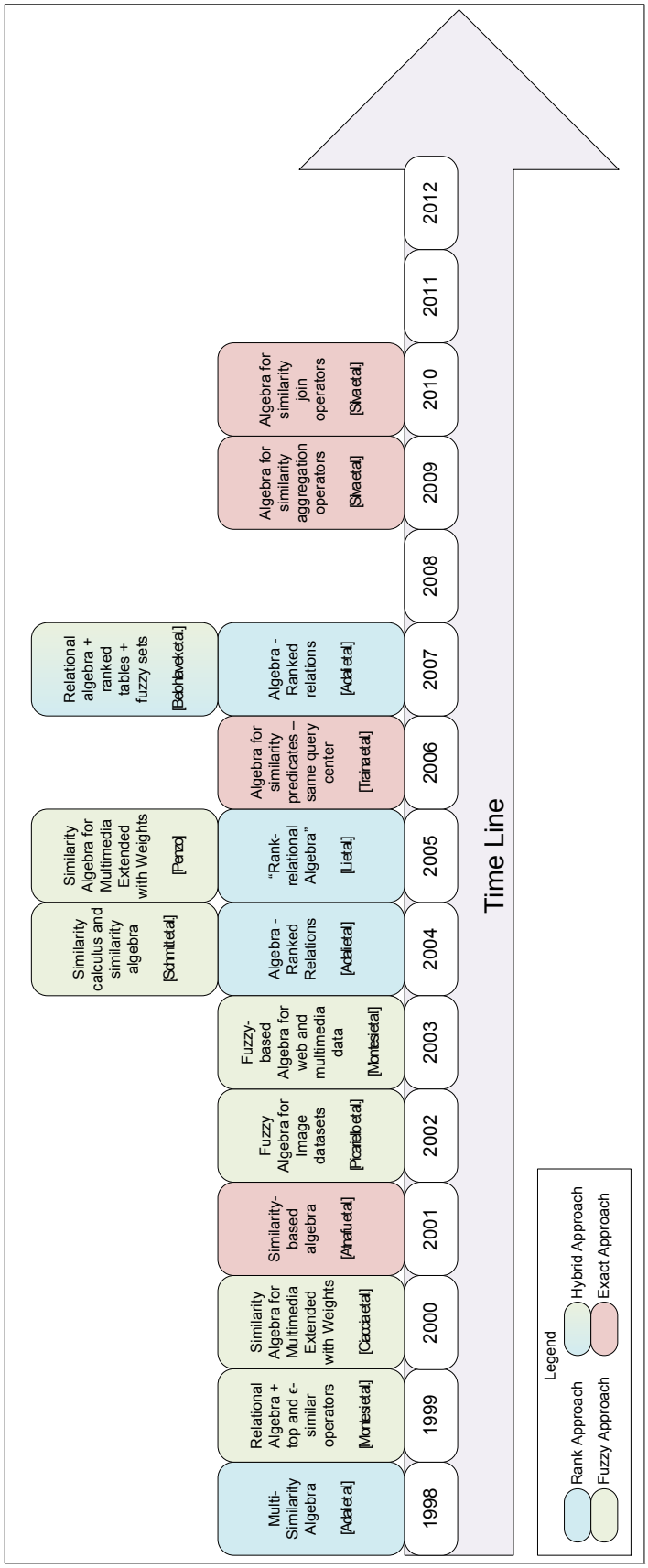


Figure 3.2: Time line for the existing similarity algebra works, following the four approaches used in MIS.

SAME^W query, the one that minimizes the size of the intermediate results [Montesi et al., 2003]. Another heuristic used by the same authors is to evaluate similarity predicates over as few tuples as possible, because they consider those predicates as being very costly [Ciaccia et al., 2001]. Herstel and Schmitt [2005] used the optimization technique called "relation-collapse" to drastically reduce the number of scans required over the base-relations, aiming at a more efficient query evaluation. The authors applied the SA algebra [Schmitt and Schulz, 2004] to map a similarity calculus expression onto a similarity algebra one, and used the semantic equivalence to transform a given similarity algebra expression into another equivalent, aimed at reducing the computational effort. Semantic equivalence in SA means achieving the same similarity values calculated.

Pursuing the exact approach, Ferreira et al. [2007] developed the query optimizer based on the query rewriting technique to interpret, translate, select the best plan and execute similarity queries over complex data indexed by a MAM. The authors proposed two data structures – the parse tree and the attribute-conditions table structures – to help processing similarity queries expressed in the similarity algebra. Both structures are used as input to the query optimizer. In this monograph, we continue to develop the query optimizer now with the objective to handle similarity queries following the technique shown in Section 4.4. That is, our work advances the state of the art proposing the syntax- and semantic-based query optimization process for similarity queries. Silva et al. [2010a] presented a similarity-aware database system (SimDB), which supports similarity operations as first-class physical database operators. The authors extended the cost-based query optimization to handle also similarity operations, adding equivalence rules for similarity group-by and similarity join operators.

Figure 3.3 presents the time line of the query optimization literature papers, following the four approaches employed for similarity in MIS.

3.6 Query Rewriting

Characterizing similarity in relational algebra, involving one or more predicates, has been evaluated by several authors. Most of the query optimization techniques involve treating complex Boolean expression in RDBMSs. In general, the goal is to rewrite a query generating an algebraically equivalent expression but that can be executed faster.

Similarity query rewriting techniques began to receive attention in Adali et al. [1998]. In that paper, the authors proved the equivalence and the containment relationships between MSA expressions, developing query rewriting methods based on these results. Chang and Hwang [2002] treated the query rewriting based on ranking using expensive predicates. Li et al. [2005] defined a set of algebraic laws that allow rewriting top-k queries to treat ranking as first-class operations. The authors argue that the algebraic equivalences should produce not only a result that have the same elements, but also that

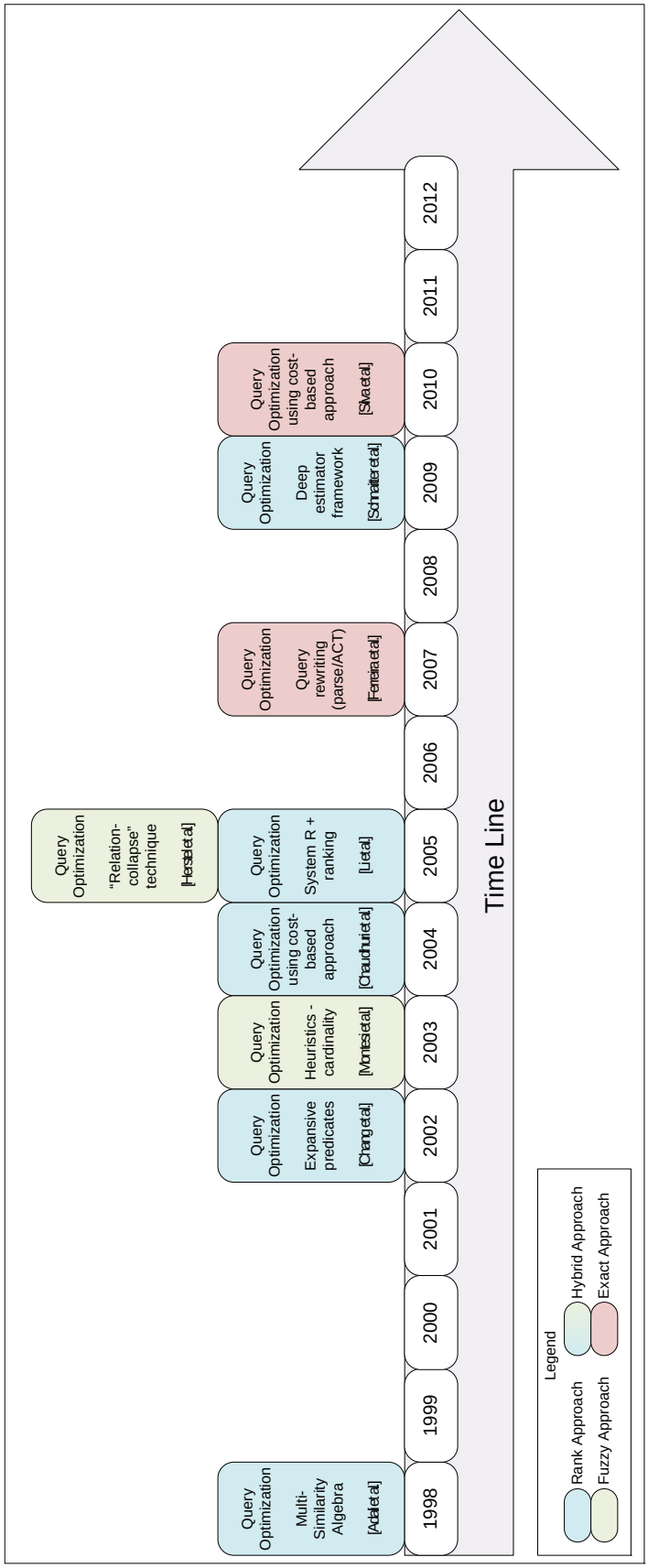


Figure 3.3: Time line for the existing query optimization works, following the four approaches used in MIS.

the elements must be serialized in same order, stating in a new freedom of commands to specify results splitting and interleaving Rank splitting allows to break a scoring function with several predicates into a series of rank operations, useful for processing the predicates individually. Interleaving asserts that rank operations can swap its execution sequence with other operators.

Ciaccia et al. [2000, 2001] presented equivalence and containment rules focusing on both *cut* and *top* operators, and the effect of weighting on fuzzy approach, such as: (i) the predicate with the highest weight could be "pushed down" and a *cut* operator could be added to discard tuples; (ii) the distribution of predicates of a conjunctive formula over the corresponding join operands; and (iii) the introduction of new *cut* operators over the operands of a weighted join, with threshold values determined by the set of weights. Montesi et al. [2003] presented the query rewriting process driven by Ciaccia et al. [2000] rules holding for the SAME^W. Herstel and Schmitt [2005] adapted some optimization rules known from the traditional database theory to rewrite the similarity algebra: (i) the order of subsequent selection may be arbitrarily changed; (ii) the Cartesian product and selection are commutative; and (iii) a selection with equality condition between two attributes of different relations and a Cartesian product can be substituted by a join.

For exact approach, Traina Jr. et al. [2006] proposed a formalism to express similarity queries in multimedia databases, promoting the support to rewrite these queries and defining a set of algorithms able to answer them described by the combination of conjunctions, disjunctions and negations of basic similarity predicates over the same query center. Chalhoub et al. [2006] presented a visual shape-based query rewriting approach, used to increase the relevance of the results. Silva et al. [2010a] presented a set of transformation rules for similarity group-by and similarity join operators. The authors showed that these transformation rules exploit: (i) specific properties of these operators; (ii) equivalence rules between multiples similarity join operators and between similarity join and similarity group-by operators; and (iii) Eager and Lazy aggregation transformations. Our work complements the exact approach literature providing equivalence- and inclusion-based rewriting properties and rules for unary similarity operators alone or mixed with other similarity and non-similarity based operators. These rewriting properties and rules allow handling queries centered either at the same or at distinct query centers.

Figure 3.4 presents the time line of the query rewriting literature papers, following the four approaches employed for similarity in MIS.

3.7 Cost and Condition Selectivity Model

In the last years, few researches have explored query cost and conditions selectivity estimation models for MAMs, regardless of the important role these models plays in the query optimization processes.

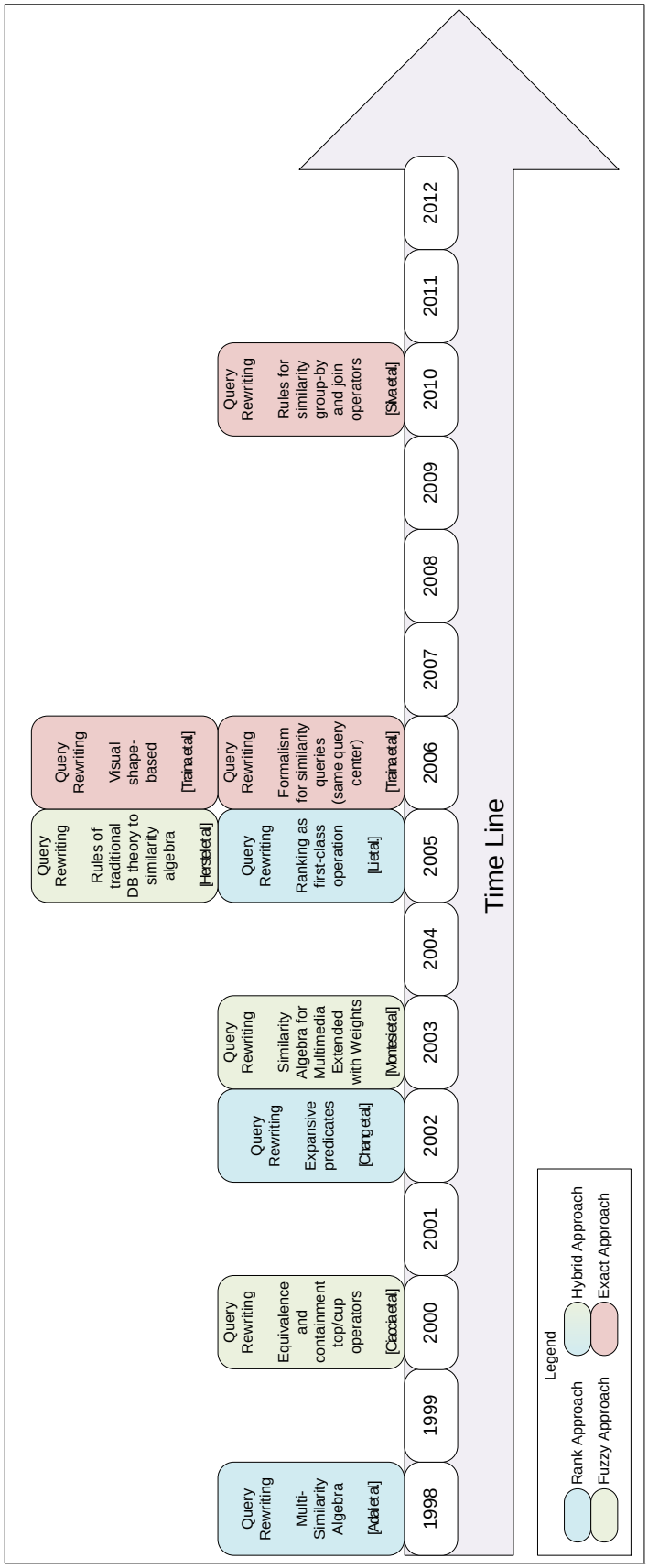


Figure 3.4: Time line for the existing query rewriting works, following the four approaches used in MIS.

Considering selectivity estimation models, Belussi and Faloutsos [1995] analyzed spatial datasets using concepts from the fractal theory. Traina Jr. et al. [2000a] presented the first selectivity condition estimation model for similarity queries in metric spaces, using the Slim-tree. The main contribution of this work was the Distance Law, an empirical power law that expresses the distance distribution function for real datasets. The exponent in this law, called the distance exponent, is the key to solve the problem of selectivity estimation on metric spaces.

The multi-dimensional selectivity estimation method for similarity queries in multimedia databases using the fuzzy approach was presented in Lee et al. [2003]. The discrete cosine transform estimates the selectivity of a similarity query, whose shape is a hyper sphere, by constructing a set of hyper rectangles that are compactly contained in the hypersphere. A histogram technique to approximate the density of multi-dimensional datasets with real attributes using the local density of the data was presented in Gunopulos et al. [2005]. A sampling-based cardinality estimation method for rank-aware operators employed to estimate the output cardinality of the query plan was presented in Li et al. [2005]. An approach based on clustering techniques was employed to approximate the selectivity of multimedia range queries in Döller and Kosch [2005]. Data density functions of relations approximated by cosine series and the usage of these approximations to estimate selectivities of range queries were proposed in Yan et al. [2007]. Sampling techniques for selectivity estimation of set similarity queries using traditional weighted similarity measures and the design of selectivity estimators based on a priori constructed samples were explored in Hadjieleftheriou et al. [2008].

The first cost model for metric trees was proposed by Ciaccia et al. [1997]. This model considers the distance distribution between pair of elements as uniform and estimates the number of disk accesses on the M-tree leaf nodes. A generic cost model to evaluate the cost of query plans in MSA was provided in Adali et al. [1998]. A methodology to model the cost to access the index structure for multi-dimensional data was presented in Böhm [2000]. The cost estimation equations considering the fractal distribution of datasets, which better approximates the behavior of real datasets was modeled in Traina Jr. et al. [2000a]. Regarding the RankSQL framework, Li et al. [2005] indicated that rank-aware operators are context-sensitive selective, and they reduce the cardinality of intermediate results because the framework does not output all the tuples processed. All of these operators depend on k, so they cannot be assumed to be independent from their locations in the whole plan, as it is commonly assumed for selection and join selectivities. Thus, the selectivity of rank-aware operators enables to reduce both the evaluation of predicates that have various estimated costs and the cost of join operations. Also, ranking query plans do not need to materialize a query, making the query plan ranking much more efficient than the traditional ones, which can be prohibitively expensive. The work of Baioco et al. [2007] presented a selectivity and cost model for similarity queries in the

Slim-tree. A cost model to integrate multiple similarity-based image joins in a multimedia database using the R-tree index family was presented in Kosch [2010].

Figure 3.5 presents the time line of the cost and condition selectivity estimation literature papers, following the four approaches employed for similarity in MIS.

3.8 Users' Preferences

The results of a query that do not match the user expectations are common situations in multimedia systems, which frequently lead to a "closed-loop" interactive process, where the user evaluation of an initial query result is fed back to the query engine and then taken into account to compute a further (possibly) "better" result, and so on [Ciaccia et al., 2000]. Users' preferences can be represented in the query engine following either the qualitative or the quantitative approaches [Stefanidis et al., 2011]. Preference query processing models are exploited through (i) expanding queries, and rewriting them to incorporate preferences in a process called query personalization; or (ii) employing preference operators to explicitly express them within queries [Stefanidis et al., 2011].

In the quantitative formulation, the amount of interest is quantified to specify the user preferences [Stefanidis et al., 2011]. This representation is useful to give the user the possibility to assign different relevances to the results obtained by his/her queries. Such "user preferences" can be expressed by weights, which adequate each answer to the user's expectations. Most of the studies in the literature considering users' preferences in similarity queries are based on fuzzy approaches, and weights are introduced to provide additional flexibility to express the user requirement [Ciaccia et al., 2001]. Many of these studies are based on the paper of Fagin and Wimmers [1997], and follow the quantitative approach for preference formulation [Stefanidis et al., 2011]. For instance, the SAME^W formulation considers the presence of weights in the similarity queries aiming at expressing the user preferences [Montesi and Penzo, 2000]. Beecks et al. [2011] presented the "unknown preference retrieval model", which is a content-based multimedia retrieval that is based on weighting the similarity measurement to retrieve all preferable element with respect to any preference setting.

When pursuing the qualitative approach, the preferences between pairs of tuples are directly expressed [Stefanidis et al., 2011]. This approach uses partial order subsets to model the preference queries, exploiting the 'ceteris paribus' (all others being equal) semantics to retrieve tuples. Following this approach, a set of conditional preference rules (cp-rules) that retrieves tuples of a relation T is expressed as [Wilson, 2004]:

$$r : A_h = a_h \wedge \cdots \wedge A_m = a_m \rightarrow (A = a_1) > (A = a_2) \ , \tag{3.1}$$

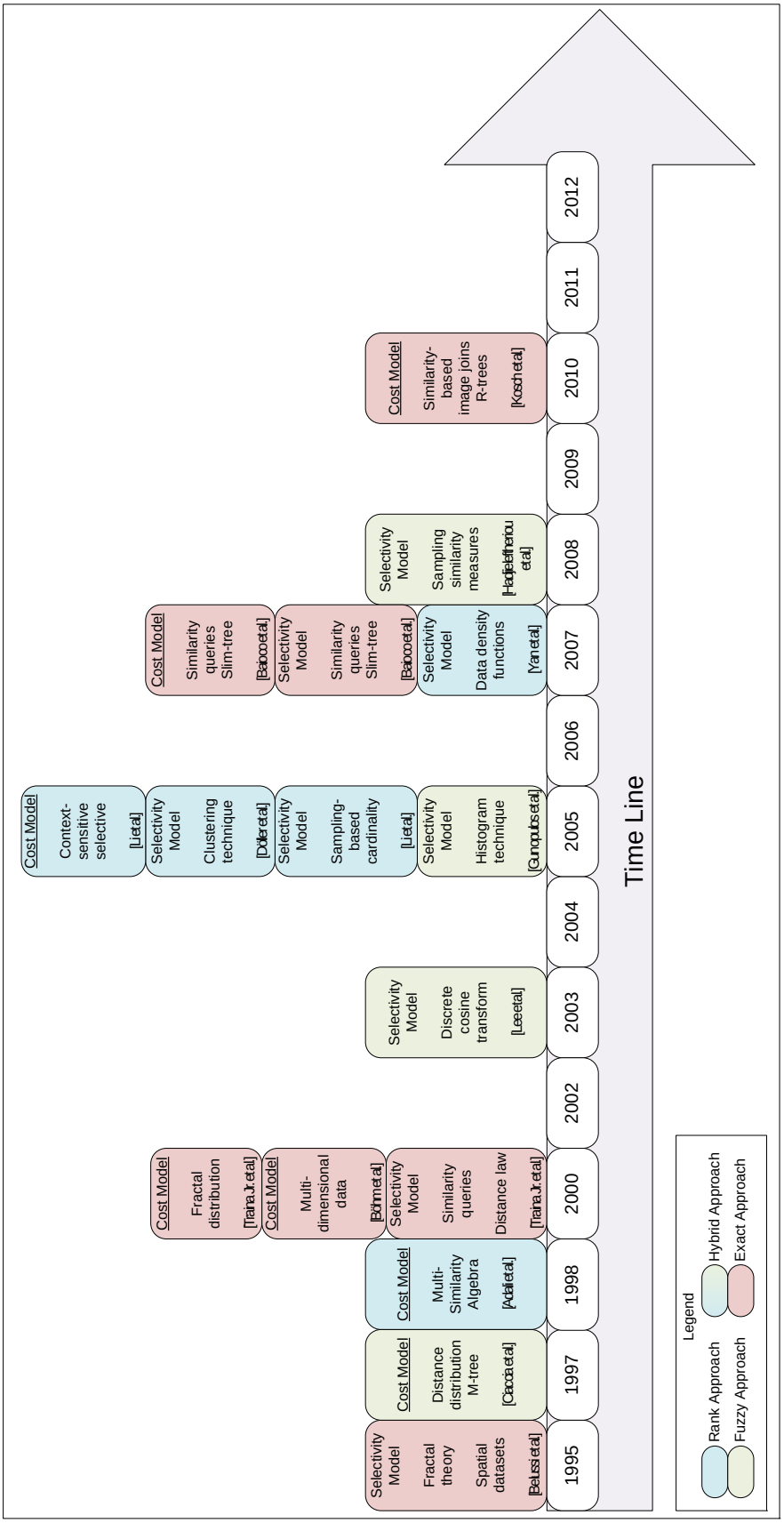


Figure 3.5: Time line for the existing cost and condition selectivity estimation works, following the four approaches used in MIS.

where $\{A_h, \ldots, A_m, A\}$ is a set of attributes of the relation T, and $a_i \in dom(A)$. The left side of rule r is called the antecedent and the right side is called the consequent of r. A set of cp-rules determines the preference partial order set of all the tuples in relation T. The semantic of a cp-rule is described as: Let t_i and t_l be two tuples from relation T. Then t_i is preferred to t_l according to the cp-rule r if $t_i[A_h] = t_l[A_h] = a_h$, for $h \in \{1, \ldots, m\}$, $t_i[A] = a_1$ and $t_l = [A] = a_2$. Tuples can be compared using the transitivity property existing over the partial order set of cp-rules [de Amo and Ribeiro, 2009].

Analyzing its expressiveness power, the qualitative formulation of preferences is more general than the quantitative one, since not all preference relations can be expressed through degrees of interest in conditions [Stefanidis et al., 2011].

3.9 Data Mining

Data mining is the main task of a Knowledge Discovery in Database (KDD) process, responsible for searching and extracting the knowledge in a large volume of data. In this step, data mining tasks and algorithms are employed over data to extract hidden patterns. Data mining tasks can be classified into two categories [Han and Kamber, 2006]: (i) Predictive, which refers to those that perform inference on the current data in order to make predictions; and (ii) Descriptive, which refers to those that characterize the general properties of the data. The main data mining tasks are: classification, clustering, association rules discovery, and summarization. This monograph focuses on the data mining association rules discovery task to support similarity retrieval over RDBMSs.

The association rules discovery task is the discovery of association rules that relate values of attributes occurring with regularities at distinct regions of the data (such as at distinct attributes of the relation or at distinct subsequences in ranked data) that occur frequently in unexpected patterns in a set of data [Han and Kamber, 2006]. This task has been extensively studied and applied to market basket analysis since its introduction by Agrawal et al. [1993], a seminal work that introduced the discovery of association relationship among sets of data items in tuples. It can be described as follows. Let $I = \{i_1, \ldots . i_n\}$ be a set of data items (stored as attribute values). A set $X \in I$ is called an itemset. Let T be a relation with tuples t involving elements that are subsets of I. An association rule is an expression of the form $X \rightarrow Y$, where X and Y are itemsets. The variable X is called the body or antecedent of the rule. The variable Y is called the head or consequent of the rule. The rule 'support' is the ratio between the number of tuples of T containing itemset $X \cup Y$ and the total number of tuples of T. The rule 'confidence' is the percentile of the number of tuples containing X that also contain Y. The problem of mining association rules, as it was firstly stated, consists of finding frequent occurrences

of tuples that satisfy the restrictions of minimum support sup_{min} and confidence $conf_{min}$ specified by the user.

Mining association rules from complex data is more complex than from the scalar one. Although this monograph focuses on association rule mining algorithms that mine complex data as image datasets, the same concepts can be employed to extract rules from other complex data types [Jiang et al., 2008], whenever adequate feature extractors are employed. This concept is exploited in Section 5.4 to integrate data mining algorithm in RDBMSs.

Image mining algorithms extracts relevant features from the images, organizing them into feature vectors [Ribeiro et al., 2008]. These vectors are employed in place of the images to represent them as in the comparison operations, and therefore they are the key information about the images that are handled in the association rule mining process. The original Apriori [Agrawal and Srikant, 1994] algorithm is modified to allow mining rules over the pre-processed data, restricting the body of a rule to be composed of feature indexes and the corresponding intervals, and the head of the rule to be composed only of an image class. Therefore, the format of an association rule that is the objective of the image mining task is:

$$f_{r_1}[l_{1_0} - l_{1_1}], \ldots, f_{r_n}[l_{n_0} - l_{n_1}] \rightarrow Class_{R_1}, \ldots, Class_{R_m}\ (sup, conf)\ . \tag{3.2}$$

The meaning of this rule (Equation 3.2) is: the images having the features $f_{r_1}, \ldots, f_{r_n}$, respectively in the closed intervals $[l_{1_0} - l_{1_1}], \ldots, [l_{n_0} - l_{n_1}]$ tend to be in classes $Class_{R_1}, \ldots, Class_{R_m}$, with support sup and confidence $conf$. The maximum number of features $max_{feature}$ in the body is the largest amount of features that can be extracted by the corresponding extractor, such that $1 \leq n \leq max_{feature}$, and the maximum number of classes max_{class} in the head is the number of classes found in the relation such that $1 \leq m \leq max_{class}$.

3.10 Final Comments

With the advent of multimedia applications, similarity operators have evoked a large attention, mainly to handle content-based retrieval of complex data. Multimedia Information Systems usually treat similarity using four different approaches: rank, fuzzy, hybrid and exact. In this chapter we discussed similarity queries, which have the objective of searching the most similar elements to a given query center, and that follow a similarity criterion. Among the several types of similarity queries, the most used are the range and the k-nearest neighbor queries. We also presented an overview of similarity algebras present in the literature, of the main query optimization techniques, query rewriting techniques, and cost and condition selectivity estimation model that explore those four

approaches. The techniques employed to explore users' preferences and data mining tasks over complex data are also presented.

In this monograph, we adopt the exact approach, and the techniques that we developed are based on the definitions and concepts of similarity queries presented in this chapter, as we will present in Chapters 4 and 5.

Chapter

4

A novel approach for Similarity Query Optimization Process in DBMSs

4.1 Introduction

As the performance of similarity queries tends to be significantly more expensive than the identity- and TOR-based queries over scalar data, improving their executions whenever possible is always worth to exploit. To this intent, three mechanisms have been applied: (i) Metric access methods based on index structures; (ii) Query optimization; and (iii) Semantic restrictions over the search space based on local or transient conditions, such as users' interests at that query time . As mentioned earlier, a metric access method organizes the set of stored elements in order to speed up similarity-based retrieval. Query optimization finds the execution plan, among all the equivalent possibilities, that bears the minimum cost. Semantic restrictions are used as conditions system-inserted in the query aiming to filter out whole subspaces of the data distribution assuredly excluded by the query semantic, thus speeding up the data retrieval. Semantic restrictions can also help to improve the efficacy of a query, as it enables retrieving answers that more closely follow the user's expectation. Employing semantic restrictions enable developing a "query refinement technique" that retrieves elements closer to the user's expectation and excludes elements that assuredly the user is not interested in. This chapter presents techniques developed in this doctorate to integrate similarity queries into relational database management systems (RDBMSs) employing these three mechanisms to improve both efficiency and efficacy of similarity queries, helping to take into account the user's expectations and enabling the use of query refinement techniques.

The structure of this chapter is as follows. Section 4.2 shows our proposal to include a set of new algebraic operators to the relational model, to allow handling simple and complex attributes and similarity queries in the same conceptual model. Section 4.3 presents an algorithm to generate the canonical plan that takes into account the new algebraic operators to process similarity queries. Section 4.4 shows techniques developed in this doctorate to integrate similarity queries together with the traditional query processing, in a way that allows performing query optimization through query rewriting. Section 4.5 presents the proposed Similarity Algebra. Section 4.6 shows how semantic restrictions can be effectively used as filter in query refinement and Section 4.7 shows the concluding remarks of this chapter.

4.2 Including similarity-based operators into the Relational Model

In order to allow managing complex data integrated with the scalar ones, we propose including into the relational model new algebraic operators to perform similarity-based operations in a way that complex and simple attributes in the tuples of a relation can be queried by similarity, identity and by relational comparisons. Notice that the new algebraic operators can be expressed in terms of the existing operators, but it is far more convenient employing them than the basic algebraic operators. This is an important consideration, as it allows that all the properties that meet by the existing operators remain valid when the new operators are included, and we need just to define the properties that involve the new ones. In this way, the relational algebra that governs the set of operators extended by the new similarity-based algebraic operators is the same original algebra (there is no extension to the algebra itself). However, to simplify referring to the "relational algebra applied to the set of operators extended by the new similarity-based algebraic operators" in this monograph, we call it simply the "extended algebra". In the same way, we call relations that include complex attributes as extended relations, even though they follows the same properties and definitions of the traditional ones. Here, the term 'simple attribute' refers to an attribute of a scalar data type that is compared by the traditional relational or identity comparison operators, while the term 'complex attribute' refers to an attribute that can be compared by similarity, i.e., it is drawn from a metric domain where a distance function was defined.

Let $A_h \subset \mathbb{A}_h$ be a simple attribute in a domain $\mathbb{A}_h$ that allows comparisons using traditional operators; $S_j \subset \mathbb{S}_j$ be a complex attribute in a domain $\mathbb{S}_j$ in a metric space that allows comparisons using complex operators; and T be a relation with any number of both simple and complex attributes. Thus, $\mathbb{T} = \{\mathbb{A}_1, \ldots, \mathbb{A}_m, \mathbb{S}_1, \ldots, \mathbb{S}_p\}$ is a relation schema and a relation T whose schema $\mathbb{T}$ is a set of attribute roles $T = \{A_1, \ldots, A_m, S_1, \ldots, S_p\}$,

that is, the domain $dom(A_h) = \mathbb{A}_h$ and the domain $dom(S_j) = \mathbb{S}_j$. Considering that T is also a set of tuples, each tuple $t = \langle a_1, \ldots, a_m, s_1, \ldots, s_p \rangle \in T$ has each value a_h $(1 \leq h \leq m)$ obtained in domain $\mathbb{A}_h$ and each value s_j $(1 \leq j \leq p)$ obtained in the domain $\mathbb{S}_j$. Notice that as attributes A_h and S_j are roles, it is possible that more than one attribute obtain values in the same domain, that is, $\mathbb{A}_h = \mathbb{A}_{h'}$ and $\mathbb{S}_j = \mathbb{S}_{j'}$. In this way, $t_i(S_j)$ $(1 \leq i \leq n)$ is the value of the i^{th} tuple on complex attribute S_j, and correspondingly $t_i(A_h)$ is the A_h-value of t_i. Figure 4.1 illustrates a relation composed of both simple and complex attributes.

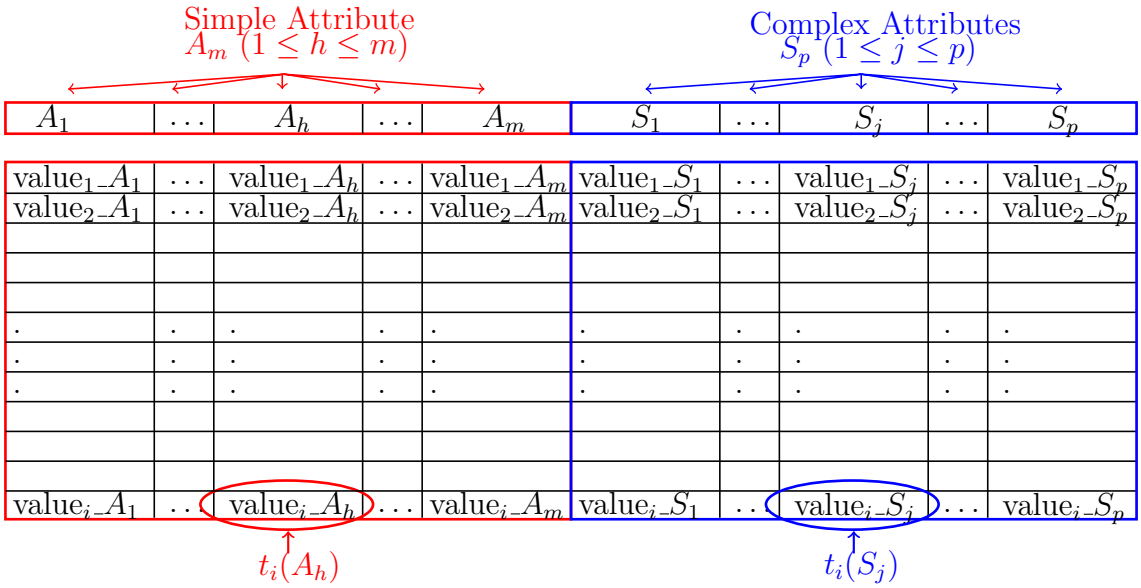


Figure 4.1: A relation composed of both simple and complex attributes.

To alleviate the notation of handling several attributes in a relation, whenever the focus of the text is over only one attribute, this thesis uses just S and $\mathbb{S}$ to refer to a complex attribute S_j and its respective domain $\mathbb{S}_j$, and A and $\mathbb{A}$ to refer to a simple attribute A_h and its respective domain $\mathbb{A}_h$.

Once the relational model had been settled, an extension of an SQL-like query language can be used to write queries that mix traditional and similarity-based predicates. In this thesis we use the extension already existing for the SIREN prototype [Barioni et al., 2009] for a SQL extension able to represent similarity queries, which we present some details in the next chapter.

4.3 Canonical Plan Algorithm

The syntax of the SQL-like query language to describe a similarity selection (an unary operator), which can be combined or not with traditional predicates, follows the same syntax as standard SQL. Queries are expressed with the "`SELECT-FROM-WHERE`" statement that has the general form presented in Figure 4.2. The strings `<table_references>`,

<attribute_list$_1$>, <attribute_list$_2$> and <attribute_list$_3$> are defined in the same way as in standard SQL, and <where_conditions> and <having_conditions> can be a Boolean combination of either traditional or similarity predicates.

```
SELECT <attribute_list1>
  FROM <table_references>
 WHERE <where_conditions>
 GROUP BY <attribute_list2>
HAVING <having_conditions>
 ORDER BY <attribute_list3>
```

Figure 4.2: General form of an SQL-like query.

When similarity queries are received by a RDBMS, the query is compiled, optimized and then executed. The query compiler makes the lexical, syntactic and semantic analysis. Besides handling the special constructs involved in the similarity-related syntax, the compilation process is virtually the same as before, since it is specific for the query language. Because this monograph does not propose a specific extension to SQL-like query language, we do not focus the description of the properties shown here on the query compiler analysis. In fact, although those properties aim at being used to improve query rewriting, they are generic to any query language supporting the involved algebraic operators.

After the compilation process has been successfully executed, the canonical tree is generated and sent as input to the query optimizer. The basic steps to translate a query involving similarity-based constructs into an algebraic expression (that is, ignoring special constructs like set-theoretical operators), and thus generating the canonical tree that includes complex attributes and similarity-based operators, are presented in Algorithm 4.1.

For illustration purposes, let us use again the CoPhIR[1] database presented in Chapter 2 and Appendix A. This database has a relation whose schema is the following:

$$\texttt{CoPhIRdb} = \{\underline{\texttt{UserId}}, \underline{\texttt{PhotoId}}, \texttt{Title}, \texttt{Description}, \texttt{Tags}, \texttt{Lat}, \texttt{Long}, \texttt{Country}, \texttt{Image}, \texttt{Coordinate}\},$$

where UserId, PhotoId, Title, Description, Tags, Lat, Long and Country are traditional attributes (colored in red), Image and Coordinate are complex attributes (colored in blue) and {UserId, PhotoId} are the attributes that compose the primary key. The Manhattan (L_1) distance function is employed to calculate the similarity between elements of the complex attribute Image, and the Euclidean (L_2) distance function is employed to calculate the similarity between elements of the complex attribute Coordinate.

[1]CoPhIR website. Available at: http://cophir.isti.cnr.it/. Accessed in: July 02, 2012.

Algorithm 4.1 Generation of canonical plan.

Input: Compiled SQL-like query.

Output: Canonical plan.

1: Read <table_references>
   1.1: if <table_references> = <table_name>, then convert $\boxed{\text{Read}_{\texttt{(relation)}}}$
   1.2: if <table_references> = <subquery>, the recall $\boxed{\text{Algorithm 4.1}}$
2: Read <where_conditions>
   2.1: if $\texttt{<}t_i(A_1)\,\theta\,\texttt{constant>}$, then convert $\boxed{\sigma_{\texttt{(condition)}}}$
   2.2: if $\texttt{<}t_i(A_1)\,\theta\,t_j(A_1)\texttt{>}$ and $i = j$, then convert $\boxed{\sigma_{\texttt{(condition)}}}$
   2.3: if $\texttt{<}t_i(A_1)\,\theta\,t_j(A_1)\texttt{>}$ and $i \neq j$, then convert $\boxed{t_i \overset{\texttt{(condition)}}{\bowtie} t_j}$
   2.4: if $\texttt{<}t_i(S_1)\,\theta_c\,\texttt{constant>}$, then convert $\boxed{\sigma_{c\,\texttt{(condition)}}}$
   2.5: if $\texttt{<}t_i(S_1)\,\theta_c\,t_j(S_1)\texttt{>}$ and $i = j$, then convert $\boxed{\sigma_{c\,\texttt{(condition)}}}$
   2.6: if $\texttt{<}t_i(S_1)\,\theta_c\,t_j(S_1)\texttt{>}$ and $i \neq j$, then convert $\boxed{t_i \overset{\texttt{(condition)}}{\bowtie_c} t_j}$
3: Where there are more than one table, then convert $\boxed{t_i \times t_j}$
4: Read $\texttt{<attribute\_list}_2\texttt{>}$, convert $\boxed{\mathscr{F}() \rightarrow \Pi_{\{\texttt{<attlist}_1\texttt{>}\cup\texttt{<attlist}_2\texttt{>}\cup\texttt{<atr.cond2>}\}}}$
5: Read <having_conditions>
   5.1: if $\texttt{<}t_i(A_1)\,\theta\,\texttt{constant>}$, then convert $\boxed{\sigma_{\texttt{(condition)}}}$
   5.2: if $\texttt{<}t_i(S_1)\,\theta_c\,\texttt{constant>}$, then convert $\boxed{\sigma_{c\,\texttt{(condition)}}}$
6: Read $\texttt{<attribute\_list}_3\texttt{>}$, convert $\boxed{Op_{\texttt{(<attlist>)}}}$
7: Read $\texttt{<attribute\_list}_1\texttt{>}$, convert $\boxed{\pi_{\texttt{\{<attlist>\}}}}$

Suppose that a user wants to retrieve three photos of beaches that are the more similar to the given photo and such that the retrieved photos were taken from tropical climate beaches. That is:

Example 4.1:

Q2: "Select the 3 beach photos more similar to the one stored at `'c:\MyBeachPhoto.jpg'`, such that the photos were taken from tropical climate beaches".

Figure 4.3(a) expresses Query Q2 in an extension of the SQL query language employed by the SIREN prototype, which is presented in Chapter 5.

The purpose of Algorithm 4.1 is to take the query expressed in SQL (or in our case, in the SQL-like language of SIREN) and convert it to its canonical tree, which is an operation-tree whose leaf nodes are the relation accessed by the query and each interior node is an relational operator. Thus, Algorithm 4.1 reads the SQL command for Query Q2 to first find <table_references> (Step 1), creating the leaf nodes. Since <table_references> is a relation name, the FROM clause is converted in just the leaf node $\boxed{\text{Read}_{\texttt{(CoPhIRdb)}}}$ (Step 1.1). Following, the algorithm searches for the WHERE clauses (Step 2). Considering the <where_conditions>, the algorithm proceeds looking for traditional predicates (Steps 2.1 to 2.3), and then for similarity-based

predicates (Steps 2.4 to 2.6). Then, the traditional condition is converted in one internal node $\sigma_{\texttt{(Lat BETWEEN -23.43 AND +23.43)}}$, and the similarity condition in another internal node $\ddot{\sigma}_{\texttt{(Image } \ddot{\theta}\texttt{(L1, 3) 'c:\MyBeachPhoto.jpg')}}$ (Steps 2.1 and 2.4, respectively) that will be applied over the result of the traditional one. Finally, the algorithm projects the list of attributes in the SELECT clause (Step 7), generating $\pi_{\texttt{\{PhotoId\}}}$. Figure 4.3(b) presents the canonical plan obtained for the SQL command that represents Query Q2.

(a) SIREN SQL-like query language

```
SELECT PhotoId
  FROM CoPhIRdb
 WHERE Image NEAR 'c:\MyBeachPhoto.jpg'
                       STOP AFTER 3
   AND Lat BETWEEN -23.43 AND +23.43
```

(b) Canonical Plan

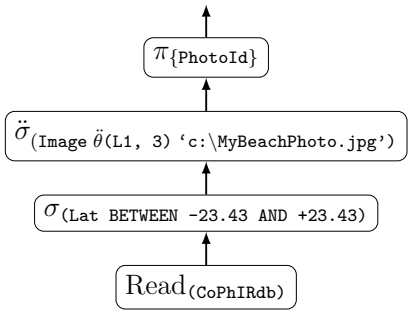


Figure 4.3: (a) A query expressed in the SIREN extension of SQL to support similarity, and (b) the canonical query plan, represented as tree, for Query **Q**2.

To employ an index, a query plan must execute the selection predicate associated to the physical operation that reads the table. When a similarity predicate is executed over a temporary, intermediate result of a previous operation, the index structures cannot be used. As the vast majority of published works on similarity queries focuses on index structures, there is a well-accepted fact (as mentioned in Section 3.5) that the kNN predicates should always be the first to be executed, preceding any other. The reasoning is that similarity-based operators are the most time-consuming ones, so it makes sense to improve its execution rather than improving the already faster identity and TOR-based ones. However, due to the lack of the commutativity property among the kNN and the other operators, as we will see following, if a k-nearest neighbor predicate is executed first, there is a great probability that the answer comes with less than k (or even with no) elements. In fact, there is no established standard for the order which similarity selections should be executed regarding the other selections. Thus, in this thesis, we prefer to execute first the selections based on identity or TOR, aiming at returning k elements whenever possible, which seems to us to be usually closer to the user's expectation. Thus, Algorithm 4.1 always generates the canonical plan executing first the traditional predicates and then the similarity ones.

4.4 Query Optimization

The query optimization process is well understood for scalar data. However, to support complex data in RDBMSs, the query optimizer must be able to rewrite the similarity queries and to estimate their costs. Therefore, some modules of the traditional query optimizer must be extended to allow handling the similarity operators. There are two distinct kinds of extensions that should be made: syntax-based and semantic-based. The syntax-based extension corresponds to identify the algebraic properties of the similarity-based operators and its interaction with the other existing operators, as well as the estimation of the corresponding execution costs.

The semantic-based extension corresponds to identify the "external" knowledge about the stored data or about the user's expectation regarding the data or about the expected query answer, generally known as semantic restrictions, that can be used to improve the execution performance and effectiveness of queries posed over complex data. Notice that handling semantic-based knowledge can also be applied over scalar data, but the resulting additional overhead has precluded its widespread use. As the processing of similarity queries is generally more time-consuming than the processing of traditional ones, the overhead of handling semantic knowledge is relatively reduced. Moreover, the semantic associated to similarity has, in general, more impact over both the query processing time and the query answer quality. Therefore, it turns out that taking into consideration the semantic-based knowledge about the stored data and about the user's expectation when a query is posed is an important asset that can be used to improve the similarity query optimization process. To handle the information about the users' interests, a third module, called the Semantic Restriction Plan Generator must be included together with the other two existing modules of the traditional query optimizer. It interacts with the other two modules, including/changing predicates that improve either the filtering predicates of the query or the screening options available to access the required data.

Figure 4.4 presents the similarity query optimizer architecture to support similarity predicates over complex data either alone or combined with traditional predicates, taking into account the semantic restrictions. The blue rectangles in this figure highlight the modules that were explored in this research. As mentioned in Chapter 2, the Logical Query Plan Generator is based on equivalence properties to apply transformations to a logical query plan and to produce other equivalent query plans that can, ideally, be executed faster. The algebraic laws and heuristics that drive the Algebraic Space Enumerator module are used to specify alternative query trees, trying to reduce the size of the space to explore. Thus, to support the similarity algebra and integrate similarity operators with the existing relational algebra, the algebraic laws and corresponding heuristics that apply both to the similarity operators and to their integration with the traditional ones must be included in the Algebraic Space Enumerator module. As the commutativity property

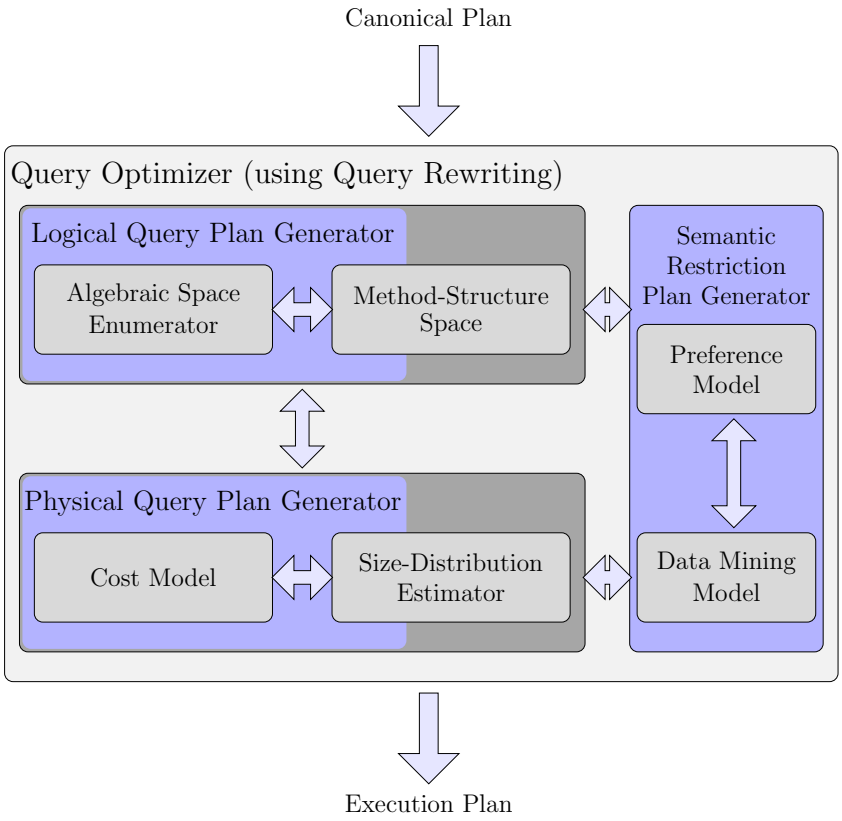


Figure 4.4: Similarity query optimizer architecture.

does not apply to the kNN operator, only few properties based on query equivalence can help in the query optimization process. Therefore, we identified a new set of properties, based on query results continence, that can be employed to generate alternative plans, with an additional bonus that they make it easier to explore semantic optimization too. The similarity algebra proposed in this doctorate is presented in Section 4.5.

To convert the logical query plan into a physical plan, the Method-Structure Space module must define the strategy that best executes both each operator and their combination, taking into account the existing indexes, either based on the metric spaces or on the traditional ones. The ordering of traditional and similarity operators specified by the Algebraic Space Enumerator module determines whether each index can improve a physical plan. Our work with real databases and applications using similarity queries has shown that often it is required to execute first the traditional and then the similarity-based operators. When this occurs, the similarity operators need to be executed without relying on an index structure; i.e., the sequential scan method must be applied over the result of the previous operators to perform the (usually costly) similarity-based operators. This is a situation that should be prevented, as it means that a MAM will be used almost exclusively when the similarity query is not combined to traditional predicates.

The usage of semantic restrictions to improve the execution performance of queries posed over complex data becomes attractive, since similarity query execution costs tend to be more expensive than traditional ones. Semantic restrictions are properties known

to exist among each particular subset of the data domain that either is effectively stored in the database, or meets users' interests. They can be employed as conditions to filter out whole subspaces of the data distribution and thus speed up data retrieval. To handle semantic restrictions in DBMSs, the query optimizer architecture must be complemented by the Semantic Restriction Plan Generator, which has two modules: (i) Preference Model and (ii) Data Mining Model. The Preference Model module evaluates criteria that express the knowledge over users' interests to identify the regions where the answer should be searched, pruning those answers that can not be found. Thus, this module is responsible for the inclusion of users' preferences in the query. The Data Mining Model module mines knowledge from the complex data stored and retrieved by previous queries, using the patterns found to detect correlations, clusters, etc. and to exploit them improving the query plan. Thus, this module is responsible to include knowledge about the stored data into the query. Techniques proposed in this doctorate to be employed in the Preference Model and the Data Mining Model modules are described in Subsections 4.6.1 and 4.6.2, respectively.

4.5 Similarity Algebra for metric spaces

The Similarity Algebra is an extension that we developed for the relational algebra to couple similarity-based algebraic operators to the already existing identity- and TOR-based operators. The fundamental properties defined by the Similarity Algebra aim at integrating both unary similarity operators, range and k-nearest neighbor, into the relational algebra. These properties allow handling queries including any number of query centers, and are suitable to support both similarity-based and traditional operators in the same query. The properties that we stated are the most flexible possible, as they allow that both the query centers, the distance functions employed and the querying attributes can be different at each predicate. Subsection 4.5.1 defines the similarity operations. Subsection 4.5.2 presents the properties of the similarity range operator and Subsection 4.5.3 defines the properties of the k-nearest neighbor operator.

4.5.1 Similarity Operations - Definitions

To begin our search for the properties involving the similarity selection, we indicate any similarity selection as a new operator in the relational algebra using the symbol σ_c in place of the traditional σ. As we will see later, the similarity range selection shares the same properties of the traditional selection, thus only the k-nearest neighbor selection effectively requires a distinct symbol. However, we start using σ_c for both similarity selections. Similarity selections follow the same syntax of the traditional ones: $\boxed{\sigma_{c\,(S\ \theta_c\ s_q)}\ T}$, where '$\sigma_c$' represents a similarity selection, S is the selection attribute chosen from the complex

attributes of the relation T, 'θ_c' is a similarity comparison operator valid in the domain $\mathbb{S}$ of the attribute S, and 's_q' is the query center and is either a constant (or an expression that returns a constant) taken in the domain $\mathbb{S}$ or the value of another attribute of T having the same domain $\mathbb{S}$ occurring in the same tuple.

The similarity comparison operator θ_c can be either the range or the k-nearest neighbor operators (and their variations), as described following. Each similarity comparison operator must define the distance function employed to measure the similarity among any pair of elements of the respective domain and the similarity threshold employed to decide whether each element meets the corresponding predicate.

The range predicate is represented as $\hat{\theta}\,(d,\ \xi)$, where d is the distance function and ξ is the similarity threshold. A similarity range selection returns every tuple where the value $t(S)$ of the attribute S differs from the query center in at most the similarity threshold ξ measured by the distance function d. Definition 4.1 shows the range selection definition. The complementary operation of range query is called Reversed Range query (R_q^{-1}).

Definition 4.1. Range query - R_q: Let S be a complex attribute taking values in the domain $\mathbb{S}$ over which the similarity condition is expressed, d be a distance function, ξ be the similarity threshold and $s_q \in \mathbb{S}$ be the query center. Then the query $\boxed{\hat{\sigma}_{\left(S\ \hat{\theta}(d,\,\xi)\ s_q\right)}\ T}$ returns every tuple $t_i \in T$ such that $d\,(t_i(S), s_q) \leq \xi$. That is,

$$\hat{\sigma}_{\left(S\ \hat{\theta}(d,\,\xi)\ s_q\right)}\ T = \{t_i \in T \mid d\,(t_i(S), s_q) \leq \xi\}\ . \tag{4.1}$$

The k-Nearest Neighbor predicate is represented as $\ddot{\theta}\,(d,\ k)$, where d is the distance function and $k \in \mathbb{N}^*$ is the similarity threshold. A k-Nearest Neighbor selection returns the tuples where the value $t(S)$ of the attribute S is one of the k elements most similar to the query center based on the distance function d. Definition 4.2 presents the k-Nearest Neighbor selection definition. The complementary operation of $k\mathrm{NN}_q$ is the k-Farthest Neighbor query ($k\mathrm{FN}_q$).

Definition 4.2. k-Nearest Neighbor query - $k\mathbf{NN}_q$: Let S be a complex attribute taking values in the domain $\mathbb{S}$ over which the similarity condition is expressed, d be a distance function, $k \in \mathbb{N}^*$ be the similarity threshold and $s_q \in \mathbb{S}$ be the query center. The query $\ddot{\sigma}_{\left(S\ \ddot{\theta}(d,\,k)\ s_q\right)}\ T$ returns the tuples $\{t_1, \ldots, t_k\} \subset T$ such that, for each $i = 1, \ldots, k$ the value of the attribute S in the tuple t_i – $t_i(S)$ – is one of the k elements in S nearest to the query center s_q based on the distance function d. That is,

$$\ddot{\sigma}_{\left(S\ \ddot{\theta}(d,\,k)\ s_q\right)}\ T = \{t_i \in T \mid \forall\, t \in T - T',\ d\,(t_i(S), s_q) \leq d\,(t(S), s_q)\}\ , \tag{4.2}$$

where $T' = \emptyset$, if $i = 1$ and $T' = \{t_1, \ldots, t_{i-1}\}$, if $1 < i \leq k$.

A more formal definition of $k\text{NN}_q$ is:

$$\ddot{\sigma}_{\left(S\ \ddot{\theta}(d,\,k)\ s_q\right)}\, T = \{t_1, \ldots, t_k\}\ , \tag{4.3}$$

where

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

Equation 4.2 is commonly used in the database literature to explain kNN operator. However, to prove the inclusion-based properties, it is convenient to express the $k\text{NN}_q$ following algebraic rules, where the concept being defined cannot itself be employed in its definitions. For this reason, Equation 4.3 presents the formal definition of $k\text{NN}_q$. Table 4.1 summarizes the unary similarity operators, showing its notation in the similarity algebra and the condition that a tuple should meet to belong to the query result.

Query		Notation	Condition
Range query	R_q	$\hat{\sigma}_{\left(S\ \hat{\theta}(d,\,\xi)\ s_q\right)}\, T$	$d(t_i(S), s_q) \leq \xi$
Point query		$\hat{\sigma}_{\left(S\ \hat{\theta}(d,\,0)\ s_q\right)}\, T$	$d(t_i(S), s_q) = 0$
Reversed Range query	R_q^{-1}	$\hat{\sigma}_{\left(S\ \hat{\theta}^{-1}(d,\,\xi)\ s_q\right)}\, T$	$d(t_i(S), s_q) > \xi$
k-Nearest Neighbor query	$k\text{NN}_q$	$\ddot{\sigma}_{\left(S\ \ddot{\theta}(d,\,k)\ s_q\right)}\, T$	$d(t_i(S), s_q) \leq d(t(S), s_q)$
k-Farthest Neighbor query	$k\text{FN}_q$	$\ddot{\sigma}_{\left(S\ \ddot{\theta}_F(d,\,k)\ s_q\right)}\, T$	$d(t_i(S), s_q) > d(t(S), s_q)$

Table 4.1: Summary of unary similarity operators.

4.5.2 Properties of the Range Selection

This subsection presents algebraic equivalence-based properties and their proofs, which are useful to rewrite expressions involving the range ($\hat{\theta}$) operator. The same properties can be used for the reversed range ($\hat{\theta}^{-1}$) operator.

These properties show that the range selection shares same algebraic equivalences as the traditional selections. Moreover, as we will show (Property R4.5), range and traditional selections are commutative. Therefore, the query optimizer of a DBMS can treat range selection using the same properties of the traditional ones.

Property R4.1 shows that the range selection operator meets the idempotent property.

Property R4.1: Let T be a relation, S in T be a complex attribute taking values in the domain $\mathbb{S}$ over which the similarity condition is expressed, $\hat{\theta}$ be the similarity range operator, d be a distance function, ξ be the similarity threshold and $s_q \in \mathbb{S}$ be the query center. Then, the range idempotent property is expressed as

$$\hat{\sigma}_{\left(S\ \hat{\theta}(d,\,\xi)\ s_q\right)}\left(\hat{\sigma}_{\left(S\ \hat{\theta}(d,\,\xi)\ s_q\right)}\ T\right) = \hat{\sigma}_{\left(S\ \hat{\theta}(d,\,\xi)\ s_q\right)}\ T\ . \tag{4.4}$$

Proof. Its proof follows directly from the Definition 4.1. Thus, Definition 4.1 is used to proof Property 4.1.

$$\begin{aligned}
\hat{\sigma}_{\left(S\ \hat{\theta}(d,\,\xi)\ s_q\right)}\left(\hat{\sigma}_{\left(S\ \hat{\theta}(d,\,\xi)\ s_q\right)}\ T\right) &= \hat{\sigma}_{\left(S\ \hat{\theta}(d,\,\xi)\ s_q\right)}\{t_i \in T \mid d(t_i(S), s_q) \leq \xi\} \\
&= \{t_i \in \{t_i \in T \mid d(t_i(S), s_q) \leq \xi\} \mid d(t_i(S), s_q) \leq \xi\} \\
&= \{t_i \in T \mid d(t_i(S), s_q) \leq \xi\ \wedge\ d(t_i(S), s_q) \leq \xi\} \\
&= \{t_i \in T \mid d(t_i(S), s_q) \leq \xi\} \\
&= \hat{\sigma}_{\left(S\ \hat{\theta}(d,\,\xi)\ s_q\right)}\ T\ .
\end{aligned}$$

□

Properties R4.2 and R4.3 consider conjunctive and disjunctive conditions of range selection operators. For conjunctive conditions, Property R4.2 shows that it can be rewritten into a cascade of individual $\hat{\sigma}$ operators or a sequence of intersection operators.

Property R4.2: Let T be a relation, $S_1, S_2 \in T$ be complex attributes taken in the same domain $\mathbb{S}$ over which the similarity condition is expressed, $\hat{\theta}$ be the similarity range operator, d_1, d_2 be distance functions, ξ_1, ξ_2 be similarity thresholds and $s_{q1}, s_{q2} \in \mathbb{S}$ be query centers. Then,

$$\begin{aligned}
\hat{\sigma}_{\left(S_1\ \hat{\theta}(d_1,\,\xi_1)\ s_{q1}\right)\ \wedge\ \left(S_2\ \hat{\theta}(d_2,\,\xi_2)\ s_{q2}\right)}\ T\ &=\ \hat{\sigma}_{\left(S_1\ \hat{\theta}(d_1,\,\xi_1)\ s_{q1}\right)}\left(\hat{\sigma}_{\left(S_2\ \hat{\theta}(d_2,\,\xi_2)\ s_{q2}\right)}\ T\right) \\
&=\ \left(\hat{\sigma}_{\left(S_1\ \hat{\theta}(d_1,\,\xi_1)\ s_{q1}\right)}\ T\right)\ \cap\ \left(\hat{\sigma}_{\left(S_2\ \hat{\theta}(d_2,\,\xi_2)\ s_{q2}\right)}\ T\right)\ .
\end{aligned} \tag{4.5}$$

The proof of Property R4.2 uses the similarity range selection Definition 4.1.

Proof.

$$
\begin{aligned}
&\hat{\sigma}_{\left(S_1\ \hat{\theta}(d_1,\,\xi_1)\ s_{q1}\right)\,\wedge\,\left(S_2\ \hat{\theta}(d_2,\,\xi_2)\ s_{q2}\right)}\ T \\
&= \{t_i \in T \mid d(t_i(S_1), s_{q1}) \leq \xi_1\ \wedge\ d(t_i(S_2), s_{q2}) \leq \xi_2\} \\
&= \{t_{i1} \in T \mid d(t_i(S_1), s_{q1}) \leq \xi_1\}\ \cap\ \{t_{i2} \in T \mid d(t_i(S_2), s_{q2}) \leq \xi_2\} \\
&= \left(\hat{\sigma}_{(S_1\ \hat{\theta}(d_1,\,\xi_1)\ s_{q1})}\ T\right)\ \cap\ \left(\hat{\sigma}_{(S_2\ \hat{\theta}(d_2,\,\xi_2)\ s_{q2})}\ T\right)\ .
\end{aligned}
$$

On the other hand,

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

□

A special case exists when $s_{q1} = s_{q2}$, which meets the property following.

Property R4.2.1: Special case where $s_{q1} = s_{q2} = s_q$.

$$
\begin{aligned}
&\left(\hat{\sigma}_{\left(S\ \hat{\theta}(d,\,\xi_1)\ s_q\right)}\ T\right)\ \cap\ \left(\hat{\sigma}_{\left(S\ \hat{\theta}(d,\,\xi_2)\ s_q\right)}\ T\right)\ = \\
&\hat{\sigma}_{\left(S\ \hat{\theta}(d,\,\xi_1)\ s_q\right)\,\wedge\,\left(S\ \hat{\theta}(d,\,\xi_2)\ s_q\right)}\ T\ =\ \hat{\sigma}_{\left(S\ \hat{\theta}\left(d,\,\min(\xi_1,\,\xi_2)\right)\ s_q\right)}\ T\ .
\end{aligned}
\tag{4.6}
$$

For disjunctive conditions, Property R4.3 presents that it can be rewritten into a sequence of union operations, as follows.

Property R4.3: Let T be a relation, S_1, S_2 in T be complex attributes taking values in the domain $\mathbb{S}$ over which the similarity condition is expressed, $\hat{\theta}$ be the similarity range operator, $d1, d_2$ be distance functions, ξ_1, ξ_2 be similarity thresholds and $s_{q1}, s_{q2} \in \mathbb{S}$ be query centers. Thus,

$$
\hat{\sigma}_{\left(S_1\ \hat{\theta}(d_1,\,\xi_1)\ s_{q1}\right)\,\vee\,\left(S_2\ \hat{\theta}(d_2,\,\xi_2)\ s_{q2}\right)}\ T = \left(\hat{\sigma}_{\left(S_1\ \hat{\theta}(d_1,\,\xi_1)\ s_{q1}\right)}\ T\right) \cup \left(\hat{\sigma}_{\left(S_2\ \hat{\theta}(d_2,\,\xi_2)\ s_{q2}\right)}\ T\right)\ . \tag{4.7}
$$

Proof. Property R4.3 can be proved using Definition 4.1.

$$
\begin{aligned}
&\hat{\sigma}_{\left(S_1\ \hat{\theta}(d_1,\,\xi_1)\ s_{q1}\right)\,\vee\,\left(S_2\ \hat{\theta}(d_2,\,\xi_2)\ s_{q2}\right)}\ T \\
&= \{t_i \in T \mid d(t_i(S_1), s_{q1}) \leq \xi_1\ \vee\ d(t_i(S_2), s_{q2}) \leq \xi_2\} \\
&= \{t_{i1} \in T \mid d(t_i(S_1), s_{q1}) \leq \xi_1\}\ \cup\ \{t_{i2} \in T \mid d(t_i(S_2), x_{q2}) \leq \xi_2\} \\
&= \left(\hat{\sigma}_{\left(S_1\ \hat{\theta}(d_1,\,\xi_1)\ s_{q1}\right)}\ T\right) \cup \left(\hat{\sigma}_{\left(S_2\ \hat{\theta}(d_2,\,\xi_2)\ s_{q2}\right)}\ T\right)\ .
\end{aligned}
$$

□

It exists a special case of Property R4.3 when $s_{q1} = s_{q2}$, which is stated as follows.

Property R4.3.1: Special case where $s_{q1} = s_{q2} = s_q$.

$$\left(\hat{\sigma}_{\left(S\ \hat{\theta}(d,\,\xi_1)\ s_q\right)}\ T\right) \cup \left(\hat{\sigma}_{\left(S\ \hat{\theta}(d,\,\xi_2)\ s_q\right)}\ T\right) =$$
$$\hat{\sigma}_{\left(S\ \hat{\theta}(d,\,\xi_1)\ s_q\right)\ \vee\ \left(S\ \hat{\theta}(d,\,\xi_2)\ s_q\right)}\ T\ =\ \hat{\sigma}_{\left(S\ \hat{\theta}\left(d,\,\max(\xi_1,\,\xi_2)\right)\ s_q\right)}\ T\ . \tag{4.8}$$

Properties R4.4 and R4.5 explore the commutativity of the $\hat{\sigma}$ operator both with other $\hat{\sigma}$ operators and with the traditional operators. Property R4.4 shows that the R_q selection operator commutes with other $\hat{\sigma}$ operators.

Property R4.4: Let T be a relation, $S_1, S_2 \in T$ be complex attributes taking values in the domain $\mathbb{S}$ over which the similarity condition is expressed, $\hat{\theta}$ be the similarity range operator, d_1, d_2 be distance functions, ξ_1, ξ_2 be similarity thresholds and $s_{q1}, s_{q2} \in \mathbb{S}$ be query centers. Then,

$$\hat{\sigma}_{\left(S_1\ \hat{\theta}(d_1,\,\xi_1)\ s_{q1}\right)}\left(\hat{\sigma}_{\left(S_2\ \hat{\theta}(d_2,\,\xi_2)\ s_{q2}\right)}\ T\right) =\ \hat{\sigma}_{\left(S_2\ \hat{\theta}(d_2,\,\xi_2)\ s_{q2}\right)}\left(\hat{\sigma}_{\left(S_1\ \hat{\theta}(d_1,\,\xi_1)\ s_{q1}\right)}\ T\right)\ . \tag{4.9}$$

Proof. This proof is obtained directly using Property R4.2 and Definition 4.1. □

Property R4.5 states that the range selection operation and traditional selection operation commutes.

Property R4.5: Let T be a relation, $S \in T$ be a complex attribute taking values in the domain $\mathbb{S}$ over which the similarity condition is expressed, $\hat{\theta}$ be the similarity range operator, d be a distance function, ξ be the similarity threshold and $s_q \in \mathbb{S}$ be the query center. Let also $A \in T$ be a traditional attribute taking values in the domain $\mathbb{A}$ over which the traditional condition is expressed, θ be either a exact match or a relational comparison operator, and a be either a constant (or an expression that returns a constant) taken in a domain of A or the value of another attribute from the same domain of A in the same tuple. Then,

$$\hat{\sigma}_{\left(S\ \hat{\theta}(d,\,\xi)\ s_q\right)}\left(\sigma_{(A\ \theta\ a)}\ T\right) =\ \sigma_{(A\ \theta\ a)}\left(\hat{\sigma}_{\left(S\ \hat{\theta}(d,\,\xi)\ s_q\right)}\ T\right)\ . \tag{4.10}$$

Proof. Definition 4.1 and definition of the traditional selection are employed to prove Property R4.5.

$$\begin{aligned}
\hat{\sigma}_{\left(S\ \hat{\theta}(d,\,\xi)\ s_q\right)}\left(\sigma_{(A\ \theta\ a)}\ T\right) &= \hat{\sigma}_{\left(S\ \hat{\theta}(d,\,\xi)\ s_q\right)}\{t_i \in T \mid t_i(A)\ \theta\ a\} \\
&= \{t_i \in \{t_i \in T \mid t_i(A)\ \theta\ a\} \mid d(t_i(S), s_q) \leq \xi\} \\
&= \{t_i \in T \mid t_i(A)\ \theta\ a\ \wedge\ d(t_i(S), s_q) \leq \xi\} \\
&= \{t_i \in \{t_i \in T \mid d(t_i(S), s_q) \leq \xi\} \mid t_i(A)\ \theta\ a\} \\
&= \sigma_{(A\ \theta\ a)}\left(\hat{\sigma}_{\left(S\ \hat{\theta}(d,\,\xi)\ s_q\right)}\ T\right)\ .
\end{aligned}$$

□

Since $\hat{\sigma}$ is commutative with σ, Properties R4.2 and R4.3 can also be employed to handle these operations. Therefore, Properties R4.2 and R4.3 can be used with expressions involving either the $\hat{\sigma}$ operator only or $\hat{\sigma}$ and σ operators.

The next set of properties involves traditional binary operators. These properties allow pushing range selections through union ($\cup$), intersection ($\cap$), difference ($-$), cross product ($\times$) and join ($\bowtie$) operators.

Property R4.6 shows that $\hat{\sigma}$ is distributive over the set-theoretical binary operators $\cup$, $-$ and $\cap$. As it is required by the relational algebra, relations T_1 and T_2 must be union compatible.

Property R4.6: Let T_1 and T_2 be two relations, $S \in T_1$ and $S \in T_2$ be a complex attribute occurring in both relations and taking values in the domain $\mathbb{S}$ over which the similarity condition is expressed, $\hat{\theta}$ be the similarity range operator, d be a distance function, ξ be the similarity threshold and $s_q \in \mathbb{S}$ be the query center. Then,

Property R4.6.1: For union:

$$\hat{\sigma}_{\left(S\ \hat{\theta}(d,\,\xi)\ s_q\right)}\left(T_1\ \cup\ T_2\right)\ =\ \left(\hat{\sigma}_{\left(S\ \hat{\theta}(d,\,\xi)\ s_q\right)}\ T_1\right)\ \cup\ \left(\hat{\sigma}_{\left(S\ \hat{\theta}(d,\,\xi)\ s_q\right)}\ T_2\right)\ . \tag{4.11}$$

Property R4.6.2: For difference:

$$\begin{aligned}
\hat{\sigma}_{\left(S\ \hat{\theta}(d,\,\xi)\ s_q\right)}\left(T_1\ -\ T_2\right) &= \left(\hat{\sigma}_{\left(S\ \hat{\theta}(d,\,\xi)\ s_q\right)}\ T_1\right)\ -\ \left(\hat{\sigma}_{\left(S\ \hat{\theta}(d,\,\xi)\ s_q\right)}\ T_2\right) \\
&= \left(\hat{\sigma}_{\left(S\ \hat{\theta}(d,\,\xi)\ s_q\right)}\ T_1\right)\ -\ T_2\ .
\end{aligned} \tag{4.12}$$

Property R4.6.3: For intersection:

$$\begin{aligned}\hat{\sigma}_{\left(S\ \hat{\theta}(d,\,\xi)\ s_q\right)}\left(T_1\ \cap\ T_2\right) &= \left(\hat{\sigma}_{\left(S\ \hat{\theta}(d,\,\xi)\ s_q\right)}\ T_1\right)\ \cap\ \left(\hat{\sigma}_{\left(S\ \hat{\theta}(d,\,\xi)\ s_q\right)}\ T_2\right)\\ &= \left(\hat{\sigma}_{\left(S\ \hat{\theta}(d,\,\xi)\ s_q\right)}\ T_1\right)\ \cap\ T_2\\ &= T_1\ \cap\ \left(\hat{\sigma}_{\left(S\ \hat{\theta}(d,\,\xi)\ s_q\right)}\ T_2\right)\ .\end{aligned} \tag{4.13}$$

Proof. The proof of Property R4.6.1 follows from the Definition 4.1. Thus:

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

The proof for Properties R4.6.2 and R4.6.3 is analogous. □

Regarding the binary join ($\bowtie$) and cross product ($\times$) operators, $\hat{\sigma}$ is distributive over the relations that contain all the complex attribute mentioned in the similarity condition. This is stated in Property R4.7.

Property R4.7: Let T_1 and T_2 be two relations, $S \in T_1$ be a complex attribute taking values in the domain $\mathbb{S}$ over which the similarity condition is expressed, $\hat{\theta}$ be the similarity range operator, Θ is either the traditional $\bowtie$ or the traditional $\times$ operators, d be a distance function, ξ be the similarity threshold and $s_q \in \mathbb{S}$ be the query center. Then:

$$\hat{\sigma}_{\left(S\ \hat{\theta}(d,\,\xi)\ s_q\right)}\left(T_1\ \Theta\ T_2\right)\ =\ \left(\hat{\sigma}_{\left(S\ \hat{\theta}(d,\,\xi)\ s_q\right)}\ T_1\right)\ \Theta\ T_2\ . \tag{4.14}$$

Proof. Let Θ be the $\times$ operator, $S \in T_1$ and Definition 4.1, then:

$$\begin{aligned}\hat{\sigma}_{\left(S\ \hat{\theta}(d,\,\xi)\ s_q\right)}\ \left(T_1\ \times\ T_2\right) &= \{(t_i, t_j) \in T_1 \times T_2 \mid d(t_i(S), s_q) \leq \xi\}\\ &= \{t_i \in T_1 \mid d(t_i(S), s_q) \leq \xi\}\ \times\ T_2\\ &= \left(\hat{\sigma}_{\left(S\ \hat{\theta}(d,\,\xi)\ s_q\right)}\ T_1\right)\ \times\ T_2\ .\end{aligned}$$

Notice that the same proof can be applied if $S \in T_2$. The proof is analogous when Θ is the $\bowtie$ operator. □

When the range selection operator is employed in a conjunctive expression, such that S_1 is a complex attribute of relation T_1 and S_2 is a complex attribute of relation T_2,

Properties R4.2 and R4.7 can be used to prove that:

$$\hat{\sigma}_{\left(S_1\ \hat{\theta}(d,\,\xi)\ s_q\right) \wedge \left(S_2\ \hat{\theta}(d,\,\xi)\ s_q\right)}\ (T_1\ \Theta\ T_2) = \left(\hat{\sigma}_{\left(S_1\ \hat{\theta}(d,\,\xi)\ s_q\right)}\ T_1\right) \Theta \left(\hat{\sigma}_{\left(S_2\ \hat{\theta}(d,\,\xi)\ s_q\right)}\ T_2\right)\ , \quad (4.15)$$

when Θ is either $\times$ or $\bowtie$. Therefore, the Equivalence 4.15 completes the Property R4.7.

4.5.3 Properties of the k-Nearest Neighbor Selection

This subsection presents algebraic equivalence-based properties and their proofs, which are useful to rewrite expressions involving the k-nearest neighbor ($\ddot{\theta}$) operator. It also presents properties derived from the inclusion of the result set of an expression in the result set of another expression. The same properties can be employed for the k-farthest neighbor ($\ddot{\theta}_F$) operator.

Based on query equivalence, the k-nearest neighbor selection operator has only three algebraic properties, two special cases and the idempotent property. Moreover, a kNN selection does not commute with any other selection operators, neither with other kNN operators. The lack of the commutativity property has strong implications on the optimization process, since this property is one of the most employed by the query optimizer to reorder operators in query plans, and even for the definition of the SQL syntax. Thus, to provide a robust set of properties to help the optimization of expressions involving the kNN selection, we enriched the set of algebraic properties regarding the kNN operator with the set of inclusion-based properties.

Equivalence-based Properties

Distinctly from the range selection operator, the k-nearest neighbor selector has only three properties, two special cases and the idempotent properties based on expression equivalence.

Property k4.1 shows that the kNN selection operation meets the idempotent property.

Property k4.1: Let T be a relation, $S \in T$ be a complex attribute taking values in the domain $\mathbb{S}$ over which the similarity condition is expressed, $\ddot{\theta}$ be the similarity kNN operator, d be a distance function, $k \in \mathbb{N}^*$ be the similarity threshold and $s_q \in \mathbb{S}$ be the query center. Then, the kNN idempotent property is expressed as

$$\ddot{\sigma}_{\left(S\ \ddot{\theta}(d,\,k)\ s_q\right)} \left(\ddot{\sigma}_{\left(S\ \ddot{\theta}(d,\,k)\ s_q\right)}\ T\right) = \ddot{\sigma}_{\left(S\ \ddot{\theta}(d,\,k)\ s_q\right)}\ T\ . \quad (4.16)$$

Proof. Property k4.1 follows directly from Definition 4.2.

$$
\begin{aligned}
&\ddot{\sigma}_{\left(S\ \ddot{\theta}(d,\,k)\ s_q\right)}\left(\ddot{\sigma}_{\left(S\ \ddot{\theta}(d,\,k)\ s_q\right)}\ T\right)\\
&\qquad=\ddot{\sigma}_{\left(S\ \ddot{\theta}(d,\,k)\ s_q\right)}\{t_i\in T\mid\forall\, t\in T-T',d\left(t_i(S),s_q\right)\le d\left(t(S),s_q\right)\}\\
&\qquad=\{t_i\in\{t_i\in T\mid\forall\, t\in T-T',d\left(t_i(S),s_q\right)\le d\left(t(S),s_q\right)\}\mid\\
&\qquad\qquad\qquad\forall\, t\in T-T',d\left(t_i(S),s_q\right)\le d\left(t(S),s_q\right)\}\\
&\qquad=\{t_i\in T\mid\forall\, t\in T-T',d\left(t_i(S),s_q\right)\le d\left(t(S),s_q\right)\ \wedge\\
&\qquad\qquad\qquad\forall\, t\in T-T',d\left(t_i(S),s_q\right)\le d\left(t(S),s_q\right)\}\\
&\qquad=\{t_i\in T\mid\forall\, t\in T-T',d\left(t_i(S),s_q\right)\le d\left(t(S),s_q\right)\}\\
&\qquad=\ddot{\sigma}_{\left(S\ \ddot{\theta}(d,\,k)\ s_q\right)}\ T\ \ ,
\end{aligned}
$$

where $T'=0$ for $i=1$ and $T'=\{t_1,\ldots,t_{i-1}\}$ for $1<i\le k$. □

Property k4.2 shows that conjunctions of $\ddot{\theta}$ operators can be rewritten into a sequence of intersection operations.

Property k4.2: Let T be a relation, $S_1,S_2\in T$ be complex attributes taking values in the domain $\mathbb{S}$ over which the similarity condition is expressed, $\ddot{\theta}$ be the similarity kNN operator, d_1,d_2 be distance functions, $k_1,k_2\in\mathbb{N}^*$ be similarity thresholds and $s_{q1},s_{q2}\in\mathbb{S}$ be query centers. Then,

$$
\ddot{\sigma}_{\left(\left(S_1\ \ddot{\theta}(d_1,\,k_1)\ s_{q1}\right)\wedge\left(S_2\ \ddot{\theta}(d_2,\,k_2)\ s_{q2}\right)\right)}\ T=\left(\ddot{\sigma}_{\left(S_1\ \ddot{\theta}(d_1,\,k_1)\ s_{q1}\right)}\ T\right)\cap\left(\ddot{\sigma}_{\left(S_2\ \ddot{\theta}(d_2,\,k_2)\ s_{q2}\right)}\ T\right)\ \ .\quad(4.17)
$$

Proof. Definition 4.2 is used to prove Property k4.2.

$$
\begin{aligned}
&\ddot{\sigma}_{\left(\left(S_1\ \ddot{\theta}(d_1,\,k_1)\ s_{q1}\right)\wedge\left(S_2\ \ddot{\theta}(d_2,\,k_2)\ s_{q2}\right)\right)}\ T\\
=&\{t_i\in T\mid\forall\, t\in T-T_1',\ d_1(t_i(S_1),s_{q1})\le d_1(t(S_1),s_{q1})\ \wedge\\
&\qquad\forall\, t\in T-T_2',\ d_2(t_i(S_2),s_{q2})\le d_2(t(S_2),s_{q2})\}\\
=&\{t_i\in T\mid\forall\, t\in T-T_1',\ d_1(t_i(S_1),s_{q1})\le d_1(t(S_1),s_{q1})\}\cap\\
&\{t_i\in T\mid\forall\, t\in T-T_2',\ d_2(t_i(S_2),s_{q2})\le d_2(t(S_2),s_{q2})\}\\
=&\left(\ddot{\sigma}_{\left(S_1\ \ddot{\theta}(d_1,\,k_1)\ s_{q1}\right)}\ T\right)\cap\left(\ddot{\sigma}_{\left(S_2\ \ddot{\theta}(d_2,\,k_2)\ s_{q2}\right)}\ T\right)\ \ ,
\end{aligned}
$$

where $T_1'=\emptyset$ for $i=1$ and $T_1'=\{t_1,\ldots,t_{i-1}\}$ for $1<i\le k_1$, whereas $T_2'=\emptyset$ for $i=1$ or $T_2'=\{t_1,\ldots,t_{i-1}\}$ for $1<i\le k_2$. □

A special case exists when $s_{q1}=s_{q2}$, which is expressed as follows.

Property k4.2.1: Special case where $s_{q1} = s_{q2} = s_q$

$$\Big(\ddot{\sigma}_{\left(S\ \ddot{\theta}(d,\,k_1)\ s_q\right)}\ T\Big) \cap \Big(\ddot{\sigma}_{\left(S\ \ddot{\theta}(d,\,k_2)\ s_q\right)}\ T\Big) = \ddot{\sigma}_{\Big(\left(S\ \ddot{\theta}(d,\,k_1)\ s_q\right) \wedge \left(S\ \ddot{\theta}(d,\,k_2)\ s_q\right)\Big)}\ T\ =\ \ddot{\sigma}_{\Big(S\ \ddot{\theta}\left(d,\,\min(k_1,\,k_2)\right)\ s_q\Big)}\ T\ . \tag{4.18}$$

Property k4.3 expresses that a disjunction of $\ddot{\theta}$ operators can be rewritten into a sequence of union operators.

Property k4.3: Let T be a relation, $S_1, S_2 \in T$ be two complex attributes taking values in the domain $\mathbb{S}$ over which the similarity condition is expressed, $\ddot{\theta}$ be the similarity kNN operator, d_1, d_2 be distance functions over $\mathbb{S}$, $k_1, k_2 \in \mathbb{N}^*$ be similarity thresholds and $s_{q1}, s_{q2} \in \mathbb{S}$ be query centers. Then,

$$\ddot{\sigma}_{\Big(\left(S_1\ \ddot{\theta}(d_1,\,k_1)\ s_{q1}\right) \vee \left(S_2\ \ddot{\theta}(d_2,\,k_2)\ s_{q2}\right)\Big)}\ T = \Big(\ddot{\sigma}_{\left(S_1\ \ddot{\theta}(d_1,\,k_1)\ s_{q1}\right)}\ T\Big) \cup \Big(\ddot{\sigma}_{\left(S_2\ \ddot{\theta}(d_2,\,k_2)\ s_{q2}\right)}\ T\Big)\ . \tag{4.19}$$

Proof. By Definition 4.2, the Property k4.3 is proved.

$$\begin{aligned}
&\ddot{\sigma}_{\Big(\left(S_1\ \ddot{\theta}(d_1,\,k_1)\ s_{q1}\right) \vee \left(S_2\ \ddot{\theta}(d_2,\,k_2)\ s_{q2}\right)\Big)}\ T \\
=&\{t_i \in T \mid \forall\, t \in T - T_1',\ d_1(t_i(S_1), s_{q1}) \leq d_1(t(S_1), s_{q1}) \vee \\
&\qquad \forall\, t \in T - T_2',\ d_2(t_i(S_2), s_{q2}) \leq d_2(t(S_2), s_{q2})\} \\
=&\{t_i \in T \mid \forall\, t \in T - T_1',\ d_1(t_i(S_1), s_{q1}) \leq d_1(t(S_1), s_{q1})\} \cup \\
&\{t_i \in T \mid \forall\, t \in T - T_2',\ d_2(t_i(S_2), s_{q2}) \leq d_2(t(S_2), s_{q2})\} \\
=&\Big(\ddot{\sigma}_{\left(S_1\ \ddot{\theta}(d_1,\,k_1)\ s_{q1}\right)}\ T\Big) \cup \Big(\ddot{\sigma}_{\left(S_2\ \ddot{\theta}(d_2,\,k_2)\ s_{q2}\right)}\ T\Big)\ ,
\end{aligned}$$

where $T_1' = \emptyset$ for $i = 1$ and $T_1' = \{t_1, \ldots, t_{i-1}\}$ for $1 < i \leq k_1$, whereas $T_2' = \emptyset$ for $i = 1$ or $T_2' = \{t_1, \ldots, t_{i-1}\}$ for $1 < i \leq k_2$. □

A special case of Property k4.3 exists when $s_{q1} = s_{q2}$, which is expressed as follows.

Property k4.3.1: Special case where $s_{q1} = s_{q2} = s_q$.

$$\Big(\ddot{\sigma}_{\left(S\ \ddot{\theta}(d,\,k_1)\ s_q\right)}\ T\Big) \cup \Big(\ddot{\sigma}_{\left(S\ \ddot{\theta}(d,\,k_2)\ s_q\right)}\ T\Big) = \ddot{\sigma}_{\Big(\left(S\ \ddot{\theta}(d,\,k_1)\ s_q\right) \vee \left(S\ \ddot{\theta}(d,\,k_2)\ s_q\right)\Big)}\ T\ =\ \ddot{\sigma}_{\Big(S\ \ddot{\theta}\left(d,\,\max(k_1,\,k_2)\right)\ s_q\Big)}\ T\ . \tag{4.20}$$

The lack of the commutativity property of the kNN operator implies that each selection must be executed separately and the intersection (in conjunctive conditions) or the union (in disjunctive conditions) of their results must be employed to compute the final answer.

As the intersection of two sets is commutative, the intersection of the results of two kNN selections is commutative, even that the kNN predicates do not commute neither in conjunctive nor in disjunctive conditions, as is shown in Property k4.4. That is, the operator $\ddot{\sigma}$ is not commutative neither with other selection operators nor with itself. For completeness reason, we express this fact in Property k4.4, as follows.

Property k4.4: Let T be a relation, $S_1, S_2 \in T$ be two complex attributes of taking values in the domain $\mathbb{S}$ over which the similarity condition is expressed, $\ddot{\theta}$ be the similarity kNN operator, d_1, d_2 be distance functions, $k_1, k_2 \in \mathbb{N}^*$ be similarity thresholds and $s_{q1}, s_{q2} \in \mathbb{S}$ be query centers. Then, for conjunctive conditions

$$\left(\ddot{\sigma}_{\left(S_1\ \ddot{\theta}(d_1,\,k_1)\ s_{q1}\right)}\ T\right) \cap \left(\ddot{\sigma}_{\left(S_2\ \ddot{\theta}(d_2,\,k_2)\ s_{q2}\right)}\ T\right) = \left(\ddot{\sigma}_{\left(S_2\ \ddot{\theta}(d_2,\,k_2)\ s_{q2}\right)}\ T\right) \cap \left(\ddot{\sigma}_{\left(S_1\ \ddot{\theta}(d_1,\,k_1)\ s_{q1}\right)}\ T\right)\ ; \tag{4.21}$$

and for disjunctive conditions

$$\left(\ddot{\sigma}_{\left(S_1\ \ddot{\theta}(d_1,\,k_1)\ s_{q1}\right)}\ T\right) \cup \left(\ddot{\sigma}_{\left(S_2\ \ddot{\theta}(d_2,\,k_2)\ s_{q2}\right)}\ T\right) = \left(\ddot{\sigma}_{\left(S_2\ \ddot{\theta}(d_2,\,k_2)\ s_{q2}\right)}\ T\right) \cup \left(\ddot{\sigma}_{\left(S_1\ \ddot{\theta}(d_1,\,k_1)\ s_{q1}\right)}\ T\right)\ . \tag{4.22}$$

The same property holds when the kNN operator $\ddot{\theta}$ is combined (conjunctively or disjunctively) either with the range operator "$\ddot{\sigma} \cap / \cup \hat{\sigma}$" or with identity and relational operators "$\ddot{\sigma} \cap / \cup \sigma$".

Proof. Property k4.4 can be proved using Definition 4.2.

$$\begin{aligned}
&\left(\ddot{\sigma}_{\left(S_1\ \ddot{\theta}(d_1,\,k_1)\ s_{q1}\right)}\ T\right) \cap \left(\ddot{\sigma}_{\left(S_2\ \ddot{\theta}(d_2,\,k_2)\ s_{q2}\right)}\ T\right) \\
=&\{t_i \in T \mid \forall\, t \in T - T'_1,\ d_1(t_i(S_1), s_{q1}) \leq d_1(t(S_1), s_{q1})\} \cap \\
&\{t_i \in T \mid \forall\, t \in T - T'_2,\ d_2(t_i(S_2), s_{q2}) \leq d_2(t(S_2), s_{q2})\} \\
=&\{t_i \in T \mid \forall\, t \in T - T'_1,\ d_1\left(t_i\left(S_1\right), s_{q1}\right) \leq d_1\left(t\left(S_1\right), s_{q1}\right)\ \wedge \\
&\qquad \forall\, t \in T - T'_2,\ d_2\left(t_i\left(S_2\right), s_{q2}\right) \leq d_2\left(t\left(S_2\right), s_{q2}\right)\} \\
=&\{t_i \in T \mid \forall\, t \in T - T'_2,\ d_2\left(t_i\left(S_2\right), s_{q2}\right) \leq d_2\left(t\left(S_2\right), s_{q2}\right)\ \wedge \\
&\qquad \forall\, t \in T - T'_1,\ d_1\left(t_i\left(S_1\right), s_{q1}\right) \leq d_1\left(t\left(S_1\right), s_{q1}\right)\} \\
=&\{t_i \in T \mid \forall\, t \in T - T'_2,\ d_2(t_i(S_2), s_{q2}) \leq d_2(t(S_2), s_{q2})\} \cap \\
&\{t_i \in T \mid \forall\, t \in T - T'_1,\ d_1(t_i(S_1), s_{q1}) \leq d_1(t(S_1), s_{q1})\} \\
&\left(\ddot{\sigma}_{\left(S_2\ \ddot{\theta}(d_2,\,k_2)\ s_{q2}\right)}\ T\right) \cap \left(\ddot{\sigma}_{\left(S_1\ \ddot{\theta}(d_1,\,k_1)\ s_{q1}\right)}\ T\right)\ ,
\end{aligned}$$

where $T'_1 = \emptyset$ for $i = 1$ and $T'_1 = \{t_1, \ldots, t_{i-1}\}$ for $1 < i \leq k_1$, whereas $T'_2 = \emptyset$ for $i = 1$ or $T'_2 = \{t_1, \ldots, t_{i-1}\}$ for $1 < i \leq k_2$. □

The proof of the conjunctive conditions over the σ and $\hat{\sigma}$ operators is analogous. The proof of the disjunctive conditions over the σ, $\hat{\sigma}$ and $\ddot{\sigma}$ operators is also realized in analogous ways.

Inclusion-based Properties

Now we present properties based on the set-inclusion of the results of similarity selections when executing a kNN operation in sequence either with other kNN operation or with traditional and similarity operations. Although they do not preserve equivalence among expressions, they can be employed by the optimizer to generate an alternative expression that surely includes every element of the original one, thus guaranteeing no false dismissals. If the alternative expression can be evaluated faster than the original one and it significantly reduces the cardinality of the working set of results, the original expression can thereafter be executed over that working set to filter out the false positives, in a way that the overall processing can possibly be faster than processing the original expression directly over the full dataset yet producing equivalent results.

Property k4.5 explains the relationship of conjunctions of $\ddot{\theta}$ operators and the composition of kNN selection operations. It states that the result of a kNN selection executed over a conjunction of $\ddot{\theta}$ operators is always included in the result of the conjunction of kNN selection operations.

Property k4.5: Let T be a relation, $S_1, S_2 \in T$ be two complex attributes taking values in the domain $\mathbb{S}$ over which the similarity condition is expressed, $\ddot{\theta}$ be the similarity kNN operator, d_1, d_2 be distance functions, $k_1, k_2 \in \mathbb{N}^*$ be similarity thresholds and $s_{q1}, s_{q2} \in \mathbb{S}$ be query centers. Then,

$$\underbrace{\ddot{\sigma}_{\left(\left(S_1\ \ddot{\theta}(d_1,\,k_1)\ s_{q_1}\right)\,\wedge\,\left(S_2\ \ddot{\theta}(d_2,\,k_2)\ s_{q_2}\right)\right)}\,T}_{(i)} \;\subseteq\; \underbrace{\ddot{\sigma}_{\left(S_1\ \ddot{\theta}(d_1,\,k_1)\ s_{q_1}\right)}\left(\ddot{\sigma}_{\left(S_2\ \ddot{\theta}(d_2,\,k_2)\ s_{q_2}\right)}\,T\right)}_{(ii)} \;. \qquad (4.23)$$

Proof. Property k4.5 follows directly from Definition 4.2.

$$\ddot{\sigma}_{\left(S_2\ \ddot{\theta}(d_2,\,k_2)\ s_{q_2}\right)}\,T = \{t_i \in T \mid \forall\, t \in T - T'_2, d_2\left(t_i(S_2), s_{q_2}\right) \leq d_2\left(t(S_2), s_{q_2}\right)\} = D \subseteq T \;\;,$$

where $T'_2 = \emptyset$ for $i = 1$ and $T'_2 = \{t_1, \ldots, t_{i-1}\}$ for $1 < i \leq k_2$.

Replacing in (ii), then:

$$(ii) = \ddot{\sigma}_{\left(S_1\ \ddot{\theta}(d_1,\,k_1)\ s_{q_1}\right)}\,D = \{t_i \in D \mid \forall\, t \in D - T'_1, d_1\left(t_i(S_1), s_{q_1}\right) \leq d_1\left(t(S_1), s_{q_1}\right)\} \;\;,$$

where $T'_1 = \emptyset$ for $i = 1$ and $T'_1 = \{t_1, \ldots, t_{i-1}\}$ for $1 < i \leq k_1$.

As $D \subseteq T$, and considering (ii):

$$\begin{aligned}(ii) &\supset \{t_i \in D \mid \forall\, t \in T - T_1', d_1\left(t_i(S_1), s_{q_1}\right) \leq d_1\left(t(S_1), s_{q_1}\right)\} \\ &= \{t_i \in T \mid \forall\, t \in T - T_2', d_2\left(t_i(S_2), s_{q_2}\right) \leq d_2\left(t(S_2), s_{q_2}\right) \wedge \\ &\qquad \forall\, t \in T - T_1', d_1\left(t_i(S_1), s_{q_1}\right) \leq d_1\left(t(S_1), s_{q_1}\right)\} \\ &= \{t_i \in T \mid \forall\, t \in T - T_1', d_1\left(t_i(S_1), s_{q_1}\right) \leq d_1\left(t(S_1), s_{q_1}\right) \wedge \\ &\qquad \forall\, t \in T - T_2', d_2\left(t_i(S_2), s_{q_2}\right) \leq d_2\left(t(S_2), s_{q_2}\right)\} \\ &= \ddot{\sigma}_{\left(\left(S_1\ \ddot{\theta}(d_1,\,k_1)\ s_{q_1}\right) \wedge \left(S_2\ \ddot{\theta}(d_2,\,k_2)\ s_{q_2}\right)\right)} T = (i)\ ,\end{aligned}$$

where T_1' and T_2' are defined as above. Thus, $(i)\ \subseteq\ (ii)$, as required. □

Property k4.6 describes the relationship of compositions of kNN and traditional selection operations. It assures that the result of a traditional selection performed over a kNN selection is always included in the result of a kNN selection performed over a traditional selection.

Property k4.6: Let T be a relation, $S \in T$ be a complex attribute taking values in the domain $\mathbb{S}$ over which the similarity condition is expressed, $\ddot{\theta}$ be the similarity kNN operator, d be a distance function, $k \in \mathbb{N}^*$ be the similarity threshold and $s_q \in \mathbb{S}$ be the query center. Let also $A \in T$ be a traditional attribute taking values in the domain $\mathbb{A}$ over which the traditional condition is expressed, θ be either a exact match or a relational comparison operator, and a be either a constant (or an expression that returns a constant) taken in a domain of A or be the value of another attribute from the same domain $\mathbb{A}$ in the same tuple. Then,

$$\underbrace{\sigma_{(A\ \theta\ a)}\left(\ddot{\sigma}_{\left(S\ \ddot{\theta}(d,\,k)\ s_q\right)} T\right)}_{(i)} \subseteq \underbrace{\ddot{\sigma}_{\left(S\ \ddot{\theta}(d,\,k)\ s_q\right)}\left(\sigma_{(A\ \theta\ a)}\ T\right)}_{(ii)}\ . \tag{4.24}$$

Proof. Property k4.6 is proved using the definitions of the traditional selection and the similarity kNN operator, as follows.

$$\sigma_{(A\ \theta\ a)}\ T\ =\ \{t_i \in T \mid t_i(A)\ \theta\ a\}\ = D \subseteq T\ ,$$

which replacing in (ii):

$$(ii)\ =\ \ddot{\sigma}_{\left(S\ \ddot{\theta}(d,\,k)\ s_q\right)}\ D = \{t_i \in D \mid \forall\, t \in D - T', d\left(t_i(S), s_q\right) \leq d\left(t(S), s_q\right)\}\ ,$$

where $T' = \emptyset$ for $i = 1$, and $T' = \{t_1, \ldots, t_{i-1}\}$ for $1 < i \leq k$.

As $D \subseteq T$, then

$$\begin{aligned}
(ii) &\supseteq \{t_i \in D \mid \forall\, t \in T - T', d\,(t_i(S), s_q) \leq d\,(t(S), s_q)\} \\
&= \{t_i \in T \mid t_i(A)\,\theta\,a \;\wedge\; \forall\, t \in T - T', d\,(t_i(S), s_q) \leq d\,(t(S), s_q)\} \\
&= \{t_i \in T \mid \forall\, t \in T - T', d\,(t_i(S), s_q) \leq d\,(t(S), s_q) \;\wedge\; t_i(A)\,\theta\,a\} \\
&= \left\{t_i \in \; \ddot{\sigma}_{\left(S\;\ddot{\theta}(d,\,k)\;s_q\right)}\, T \mid t_i(A)\,\theta\,a\right\} \\
&= \sigma_{(A\;\theta\;a)}\left(\ddot{\sigma}_{\left(S\;\ddot{\theta}(d,\,k)\;s_q\right)}\, T\right) \;=\; (i)\;\;,
\end{aligned}$$

where T' is defined as above, and $(i) \;\subseteq\; (ii)$, as required. □

Analogously, Property k4.7 correlates range and kNN selection operations. It expresses that the result of the range selection executed over the result of a kNN selection is included in the result of the kNN selection performed over the result of the range selection.

Property k4.7: Let T be a relation, $S_1, S_2 \in T$ be complex attributes taking values in the domain $\mathbb{S}$ over which the similarity condition is expressed, $\ddot{\theta}$ be the similarity kNN operator, $\hat{\theta}$ be the similarity range operator, d_1, d_2 be distance functions, $k \in \mathbb{N}^*$ and ξ be the similarity thresholds of the kNN operator and the similarity range operators respectively and $s_{q1}, s_{q2} \in \mathbb{S}$ be query centers. Then,

$$\underbrace{\hat{\sigma}_{\left(S_2\;\hat{\theta}(d_2,\,\xi)\;s_{q2}\right)}\left(\ddot{\sigma}_{\left(S_1\;\ddot{\theta}(d_1,\,k)\;s_{q1}\right)}\, T\right)}_{(i)} \;\subseteq\; \underbrace{\ddot{\sigma}_{\left(S_1\;\ddot{\theta}(d_1,\,k)\;s_{q1}\right)}\left(\hat{\sigma}_{\left(S_2\;\hat{\theta}(d_2,\,\xi)\;s_{q2}\right)}\, T\right)}_{(ii)} \;. \tag{4.25}$$

Proof. Property k4.7 is proved by Definition 4.1.

$$\hat{\sigma}_{\left(S_2\;\hat{\theta}(d_2,\,\xi)\;s_{q2}\right)}\, T = \;\{t_i \in T \mid d_2\,(t_i(S_2), s_{q2}) \;\leq\; \xi\} \;= D \subseteq T \;\;.$$

Then,

$$(ii) \;=\; \ddot{\sigma}_{\left(S_1\;\ddot{\theta}(d_1,\,k)\;s_{q1}\right)}\, D = \{t_i \in D \mid \forall\, t \in D - T', d_1\,(t_i(S_1), s_{q1}) \leq d_1\,(t(S_1), s_{q1})\}\;\;,$$

where $T' = \emptyset$ for $i = 1$, and $T' = \{t_1, \ldots, t_{i-1}\}$ for $1 < i \leq k$.

As $D \subseteq T$, then

$$
\begin{aligned}
(ii) &\supseteq \{t_i \in D \mid \forall\, t \in T - T', d_1\left(t_i(S_1), s_{q1}\right) \leq d_1\left(t(S_1), s_{q1}\right)\} \\
&= \{t_i \in T \mid d_2\left(t_i(S_2), s_{q2}\right) \leq \xi \wedge \forall\, t \in T - T', d_1\left(t_i(S_1), s_{q1}\right) \leq d_1\left(t(S_1), s_{q1}\right)\} \\
&= \{t_i \in T \mid \forall\, t \in T - T', d_1\left(t_i(S_1), s_{q1}\right) \leq d_1\left(t(S_1), s_{q1}\right) \wedge d_2\left(t_i(S_2), s_{q2}\right) \leq \xi\} \\
&= \left\{t_i \in \ddot{\sigma}_{\left(S_1\ \ddot{\theta}(d_1,\, k)\ s_{q1}\right)}\, T \mid d_2\left(t_i(S_2), s_{q2}\right) \leq \xi\right\} \\
&= \hat{\sigma}_{\left(S_2\ \hat{\theta}(d_2,\, \xi)\ s_{q2}\right)}\left(\ddot{\sigma}_{\left(S_1\ \ddot{\theta}(d_1,\, k)\ s_{q1}\right)}\, T\right) = (i)\ ,
\end{aligned}
$$

where T' is as above. Then, the Equation 4.25 is true. □

The next properties involve the set theoretical binary operators union ($\cup$), difference ($-$), intersection ($\cap$) and cross product ($\times$), and also the relational operator join ($\bowtie$) using the traditional identity and TOR-based predicates.

Property k4.8 describes the effect of applying the kNN selection over the traditional union binary operator. The result of the executing a kNN selection over the union of relations is included in the result of performing the union of the result of kNN selections executed in both relations.

Property k4.8: Let T_1 and T_2 be two relations, $S \in T_1$ and $S \in T_2$ be a complex attribute occurring in both relations and taking values in the domain $\mathbb{S}$ over which the similarity condition is expressed, $\ddot{\theta}$ be the similarity kNN operator, d be a distance function, $k \in \mathbb{N}^*$ be the similarity threshold and $s_q \in \mathbb{S}$ be the query center. Then,

$$
\ddot{\sigma}_{\left(S\ \ddot{\theta}(d,\, k)\ s_q\right)}\left(T_1 \cup T_2\right) \subseteq \left(\ddot{\sigma}_{\left(S\ \ddot{\theta}(d,\, k)\ s_q\right)}\, T_1\right) \cup \left(\ddot{\sigma}_{\left(S\ \ddot{\theta}(d,\, k)\ s_q\right)}\, T_2\right)\ . \tag{4.26}
$$

Proof. Equation 4.3 and the induction over k allow proving Property 4.8. Let us consider

- $\ddot{\sigma}_{\left(S\ \ddot{\theta}(d,\, k)\ s_q\right)}\left(T_1 \cup T_2\right) = \{t'_1, t'_2, \ldots, t'_k\}$,
- $\left(\ddot{\sigma}_{\left(S\ \ddot{\theta}(d,\, k)\ s_q\right)}\, T_1\right) = \{t_{1_1}, t_{1_2}, \ldots, t_{1_k}\}$ and
- $\left(\ddot{\sigma}_{\left(S\ \ddot{\theta}(d,\, k)\ s_q\right)}\, T_2\right) = \{t_{2_1}, t_{2_2}, \ldots, t_{2_k}\}$.

Let us first show that Property 4.8 holds when $k = 1$. In this case, $t'_1 \in T_1 \cup T_2$ is such that

$$
\begin{aligned}
d\left(t'_1(S), s_q\right) &= \min\left\{d\left(t(S), s_q\right), t \in T_1 \cup T_2\right\} \\
&\leq \min\left\{d\left(t(S), s_q\right), t \in T_1\right\}\ , \text{ and} \\
d\left(t'_1(S), s_q\right) &\leq \min\left\{d\left(t(S), s_q\right), t \in T_2\right\}\ .
\end{aligned}
$$

Therefore,

$$\begin{aligned} d\left(t'_1(S), s_q\right) &\leq \min\left\{\min\left\{d\left(t(S), s_q\right), t \in T_1\right\}, \min\left\{d\left(t(S), s_q\right), t \in T_2\right\}\right\} \\ &= \min\left\{d\left(t_{1_1}(S), s_q\right), d\left(t_{2_1}(S), sq\right)\right\} \quad . \end{aligned}$$

As $t'_1 \in \{t_{1_1}, t_{2_1}\}$, we proved that Equation 4.26 holds when $k = 1$.

Now considering $k = 2$, we know that $t'_2 \in T_1 \cup T_2$ is such that

$$\begin{aligned} d\left(t'_2(S), s_q\right) &= \min\left\{d\left(t(S), s_q\right), t \in T_1 \cup T_2 - \{t'_1\}\right\} \\ &\leq \min\left\{d\left(t(S), s_q\right), t \in T_1 - \{t'_1\}\right\} \quad , \text{ and} \\ d\left(t'_2(S), s_q\right) &\leq \min\left\{d\left(t(S), s_q\right), t \in T_2 - \{t'_1\}\right\} \quad . \end{aligned}$$

Therefore,

$$d\left(t'_2(S), s_q\right) \leq \min\left\{\min\left\{d\left(t(S), s_q\right), t \in T_1 - \{t'_1\}\right\}, \min\left\{d\left(t(S), s_q\right), t \in T_2 - \{t'_1\}\right\}\right\} \quad .$$

As $t'_1 \in \{t_{1_1}, t_{2_1}\}$, we have that $d\left(t'_2(S), s_q\right) \leq \min\left\{d\left(t_{1_2}(S), s_q\right), d\left(t_{2_2}(S), s_q\right)\right\}$ when $t'_1 = t_{1_1} = t_{2_1}$. Thus, $t'_2 \in \{t_{1_1}, t_{1_2}\} \cup \{t_{2_1}, t_{2_2}\}$.

When $t'_1 = t_{1_1}$ and $t'_1 \neq t_{2_1}$, then

$$d\left(t'_2(S), s_q\right) \leq \min\left\{d\left(t_{1_2}(S), s_q\right), \min\left\{d\left(t(S), s_q\right); t \in T_2 - \{t'_1\}\right\}\right\} \quad ,$$

with $t'_1 \notin T_2$. Thus $d\left(t'_2(S), s_q\right) \leq \min\left\{d\left(t_{1_2}(S), s_q\right), d\left(t_{2_1}(S), s_q\right)\right\}$, and, consequently, $t'_2 \in \{t_{1_1}, t_{1_2}, t_{2_1}\} \subseteq \{t_{1_1}, t_{1_2}\} \cup \{t_{2_1}, t_{2_2}\}$.

Analogously, when $t_1 = t_{2_1}$ and $t_1 \neq t_{1_1}$, we obtain $t'_2 \in \{t_{1_1}, t_{2_1}, t_{2_2}\} \subseteq \{t_{1_1}, t_{1_2}\} \cup \{t_{2_1}, t_{2_2}\}$. Therefore, Property 4.8 is valid for $k = 2$.

We complete the proof supposing that the result is valid for $k - 1$. In this case, first suppose that $t'_k \in T_1 \cup T_2$. In such case

$$\begin{aligned} d\left(t'_k(S), s_q\right) =& \min\left\{d\left(t(S), s_q\right), t \in T_1 \cup T_2 - \left\{t'_1, t'_2, \ldots, t'_{k-1}\right\}\right\} \\ \leq& \min\{\min\left\{d\left(t(S), s_q\right), t \in T_1 - \left\{t'_1, t'_2, \ldots, t'_{k-1}\right\}\right\} \quad , \\ & \min\left\{d\left(t(S), s_q\right), t \in T_2 - \left\{t'_1, t'_2, \ldots, t'_{k-1}\right\}\right\}\} \quad , \end{aligned}$$

with $\left\{t'_1, t'_2, \ldots, t'_{k-1}\right\} \subseteq \left\{t_{1_1}, t_{1_2}, \ldots, t_{1_{k-1}}\right\} \cup \left\{t_{2_1}, t_{2_2}, \ldots, t_{2_{k-1}}\right\}$. Thus we need to show that $t'_k \in \{t_{1_1}, \ldots, t_{1_k}\} \cup \{t_{2_1}, \ldots, t_{2_k}\}$, what can be done in the way following.

If $\{t_1, t_2, \ldots, t_{k-1}\} = \left\{t_{1_1}, t_{1_2}, \ldots, t_{1_{k-1}}\right\}$ or $\{t_1, t_2, \ldots, t_{k-1}\} = \left\{t_{2_1}, t_{2_2}, \ldots, t_{2_{k-1}}\right\}$ then, analogously to the cases above, we obtain

$$t'_k \in \{t_{1_1}, t_{1_2}, \ldots, t_{1_k}\} \cup \{t_{2_1}, t_{2_2}, \ldots, t_{2_k}\} \quad .$$

Otherwise, using the induction hypothesis, and without loss of generality, we can assume that exists an integer r, with $1 \leq r < k-1$, such that

$$t'_i = t_{1_i} \notin T_2, \text{ for each } i = 1, \ldots, r \text{ and}$$
$$t'_i = t_{2_{(i-r)}} \notin T_1, \text{ for each } i = r+1, \ldots, k-1.$$

Then, $T_1 - \{t'_1, \ldots, t'_{k-1}\} = T_1 - \{t_{1_1}, \ldots, t_{1_r}\}$ and $T_2 - \{t'_1, \ldots, t'_{k-1}\} = T_2 - \{t_{2_1}, \ldots, t_{2_{(k-1)-r}}\}$.

Therefore,

$$\begin{aligned}\min\left\{d\left(t(S), s_q\right); t \in T_1 - \{t'_1, \ldots, t'_{k-1}\}\right\} &= \min\left\{d\left(t(S), s_q\right);\ t \in T_1 - \{t_{1_1}, \ldots, t_{1_r}\}\right\} \\ &= d\left(t_{1_{r+1}}(S), s_q\right)\end{aligned}$$

and

$$\begin{aligned}\min\{d(t(S), s_q);\ t \in T_2 - \{t'_1, \ldots, t'_{k-1}\}\} & \\ &= \min\{d(t(S), s_q);\ t \in T_2 - \{t_{2_1}, \ldots, t_{2_{(k-1)-r}}\}\} \\ &= d\left(t_{2_{k-r}}(S), s_q\right)\ .\end{aligned}$$

In this way, we have that $d\left(t'_k(S), s_q\right) \leq \min\left\{d\left(t_{1_{r+1}}(S), s_q\right), d\left(t_{2_{k-r}}(S), s_q\right)\right\}$, which implies that

$$t'_k \in \{t_{1_1}, \ldots, t_{1_r}\} \cup \{t_{2_1}, \ldots, t_{2_{(k-1)-r}}\} \subseteq \{t_{1_1}, \ldots, t_{1_k}\} \cup \{t_{2_1}, \ldots, t_{2_k}\}\ ,$$

concluding the proof. □

The property of the relationship among kNN selection and traditional set difference binary operator is presented in Property k4.9. It states that the result of the difference between a kNN selection in the left relation and the right relation is included in the result of the kNN executed over the difference of relations. Moreover, the result of the difference between the kNN selection in the first relation and the second relation is included in the result of the difference executed over the kNN selection operations in both relations.

Property k4.9: Let T_1 and T_2 be two relations, $S \in T_1$ and $S \in T_2$ be a complex attribute occurring in both relations and taking values in the domain $\mathbb{S}$ over which the similarity condition is expressed, $\ddot{\theta}$ be the similarity kNN operator, d be a distance function, $k \in \mathbb{N}^*$ be the similarity threshold and $s_q \in \mathbb{S}$ be the query center. Then,

$$\underbrace{\left(\ddot{\sigma}_{\left(S\ \ddot{\theta}(d,\,k)\ s_q\right)} T_1\right) - T_2}_{(i)} \subseteq \underbrace{\ddot{\sigma}_{\left(S\ \ddot{\theta}(d,\,k)\ s_q\right)} (T_1 - T_2)}_{(ii)}\ ; \tag{4.27}$$

and

$$\left(\ddot{\sigma}_{\left(S\ \ddot{\theta}(d,\,k)\ s_q\right)}\ T_1\right)\ -\ T_2\ \subseteq\ \left(\ddot{\sigma}_{\left(S\ \ddot{\theta}(d,\,k)\ s_q\right)}\ T_1\right)\ -\ \left(\ddot{\sigma}_{\left(S\ \ddot{\theta}(d,\,k)\ s_q\right)}\ T_2\right)\ . \tag{4.28}$$

Proof. Definition 4.2 is used to prove Property k4.9 (Equation 4.27):

$$(i) = \left(\ddot{\sigma}_{\left(S\ \ddot{\theta}(d,\,k)\ s_q\right)}\ T_1\right) - T_2 = \{t_i \in T_1 \mid \forall\, t \in T_1 - T', d\,(t_i(S), s_q) \leq d\,(t(S), s_q)\} - T_2\ ,$$

where $T' = \emptyset$ for $i = 1$, and $T' = \{t_1, \ldots, t_{i-1}\}$ for $1 < i \leq k$.

As $T_1 - T_2 \subseteq T_1$, then

$$\begin{aligned}(i) &\subseteq \{t_i \in T_1 \mid \forall\, t \in (T_1 - T_2) - T', d\,(t_i(S), s_q) \leq d\,(t(S), s_q)\} - T_2 \\ &= \{t_i \in (T_1 - T_2) \mid \forall\, t \in (T_1 - T_2) - T', d\,(t_i(S), s_q) \leq d\,(t(S), s_q)\} \\ &= \ddot{\sigma}_{\left(S\ \ddot{\theta}(d,\,k)\ s_q\right)}\,(T_1\ -\ T_2) = (ii)\ ,\end{aligned}$$

where T' is as above, completing the proof of Equation 4.27.

Now, using the Definition 4.2 and the fact that $\left(\ddot{\sigma}_{\left(S\ \ddot{\theta}(d,\,k)\ s_q\right)}\ T_2\right) \subseteq T_2$, the inclusion in Equation 4.28 is directly proved by set theory, as presented by the Venn diagram shown in Figure 4.5. □

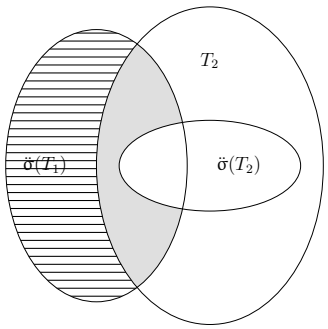


Figure 4.5: Venn diagram representation of inclusion property of Equation 4.28.

Property k4.10 considers the relationship among a kNN selection and the traditional intersection binary operators. It states that the result of the intersection between a kNN selection in the first relation and the second relation is included in the result of the kNN executed over the intersection of both relations. Also, the result of the intersection between a kNN selection on the left relation and the kNN selection on the right relation is properly included in the result of the kNN executed over the intersection of both relations.

Property k4.10: Let T_1 and T_2 be two relations, $S \in T_1$ and $S \in T_2$ be a complex attribute occurring in both relations and taking values in the domain $\mathbb{S}$ over which the

similarity condition is expressed, $\ddot{\theta}$ be the similarity kNN operator, d be a distance function, $k \in \mathbb{N}^*$ be the similarity threshold and $s_q \in \mathbb{S}$ be the query center. Then,

$$\underbrace{\left(\ddot{\sigma}_{\left(S\ \ddot{\theta}(d,\,k)\ s_q\right)}\ T_1\right)\ \cap\ T_2}_{(i)}\ \subseteq\ \underbrace{\ddot{\sigma}_{\left(S\ \ddot{\theta}(d,\,k)\ s_q\right)}\,(T_1\ \cap\ T_2)}_{(ii)}\ ; \tag{4.29}$$

and

$$\left(\ddot{\sigma}_{\left(S\ \ddot{\theta}(d,\,k)\ s_q\right)}\ T_1\right)\ \cap\ \left(\ddot{\sigma}_{\left(S\ \ddot{\theta}(d,\,k)\ s_q\right)}\ T_2\right)\ \subseteq\ \ddot{\sigma}_{\left(S\ \ddot{\theta}(d,\,k)\ s_q\right)}\,(T_1\ \cap\ T_2)\ \ . \tag{4.30}$$

Proof. By Definition 4.2, then

$$(i) = \left(\ddot{\sigma}_{\left(S\ \ddot{\theta}(d,\,k)\ s_q\right)}\ T_1\right) \cap T_2 = \{t_i \in T_1 \mid \forall\, t \in T_1 - T', d\,(t_i(S), s_q) \leq d\,(t(S), s_q)\} \cap T_2\ \ ,$$

where $T' = \emptyset$ for $i = 1$, and $T' = \{t_1, \ldots, t_{i-1}\}$ for $1 < i \leq k$.

As $T_1\ \cap\ T_2 \subseteq T_1$, then

$$\begin{aligned}(i) &\subseteq \{t_i \in T_1 \mid \forall\, t \in T_1 \cap T_2 - T', d\,(t_i(S), s_q) \leq d\,(t(S), s_q)\} \cap T_2\\ &= \{t_i \in T_1 \cap T_2 \mid \forall\, t \in T_1 \cap T_2 - T', d\,(t_i(S), s_q) \leq d\,(t(S), s_q)\}\\ &= \ddot{\sigma}_{\left(S\ \ddot{\theta}(d,\,k)\ s_q\right)}\,(T_1\ \cap\ T_2) = (ii)\ \ ,\end{aligned}$$

where T' is as above, which prove Equation 4.29.

Finally, Equation 4.30 is directly proved from the set theory properties, in the way following. As $\left(\ddot{\sigma}_{\left(S\ \ddot{\theta}(d,\,k)\ s_q\right)}\ T_2\right)\ \subseteq\ T_2$, then $\left(\ddot{\sigma}_{\left(S\ \ddot{\theta}(d,\,k)\ s_q\right)}\ T_1\right) \cap \left(\ddot{\sigma}_{\left(S\ \ddot{\theta}(d,\,k)\ s_q\right)}\ T_2\right)\ \subseteq\ \left(\ddot{\sigma}_{\left(S\ \ddot{\theta}(d,\,k)\ s_q\right)}\ T_1\right) \cap T_2$. Thus, we directly obtain Equation 4.30 from Equation 4.29. □

Property k4.11 states the relationship between a kNN selection and the traditional cross product operator. The result of the kNN executed over the cross product of the relations is included in the result of the cross product between the execution of a kNN selection over one relation and the other relation.

Property k4.11: Let T_1 and T_2 be two relations, $S \in T_1$ be a complex attribute taking values in the domain $\mathbb{S}$ over which the similarity condition is expressed, $\ddot{\theta}$ be the similarity kNN operator, d be a distance function, $k \in \mathbb{N}^*$ be the similarity threshold and $s_q \in \mathbb{S}$ be the query center. Then,

$$\underbrace{\ddot{\sigma}_{\left(S\ \ddot{\theta}(d,\,k)\ s_q\right)}\,(T_1\ \times\ T_2)}_{(i)}\ \subseteq\ \underbrace{\left(\ddot{\sigma}_{\left(S\ \ddot{\theta}(d,\,k)\ s_q\right)}\ T_1\right)\ \times\ T_2}_{(ii)}\ \ . \tag{4.31}$$

Proof. As S is a complex attribute in T_1, we have that

$$\begin{aligned}(i) &= \ddot{\sigma}_{\left(S\ \ddot{\theta}(d,\,k)\ s_q\right)}\left(T_1 \times T_2\right) \\ &= \{(t_i, t_j) \in T_1 \times T_2 \mid \forall\ (t_1, t_2) \in T_1 \times T_2 - T', d\left(t_i(S), s_q\right) \leq d\left(t_1(S), s_q\right)\}\ ,\end{aligned}$$

where $T' = \emptyset$ for either $i = 1$ or $j = 1$, and $T' = \{(t_{1_1}, t_{2_1}), \ldots, (t_{1_{i-1}}, t_{2_{j-1}})\}$ for $1 < i, j \leq k$.

As T_1 is as a subset of $T_1 \times T_2$, we obtain

$$\begin{aligned}(i) &\subseteq \{(t_i, t_j) \in T_1 \times T_2 \mid \forall\, t_1 \in T_1 - T', d\left(t_i(S), s_q\right) \leq d\left(t_1(S), s_q\right)\} \\ &= \{t_i \in T_1 \mid \forall\, t_1 \in T_1 - T', d\left(t_i(S), s_q\right) \leq d\left(t_1(S), s_q\right)\} \times T_2 \\ &= \left(\ddot{\sigma}_{\left(S\ \ddot{\theta}(d,\,k)\ s_q\right)} T_1\right)\ \times\ T_2\ =\ (ii)\ ,\end{aligned}$$

where T' is as above. □

A special case exists regarding the relationship of a kNN selection in conjunctive conditions and the traditional cross product binary operator. In this case, the result of a conjunction of a kNN executed over the cross product of two relations is included in the result of the cross product between the execution of a kNN selection being executed over the left and again over right relation. This special case is presented in Property k4.11.1.

Property k4.11.1: Let T_1 and T_2 be two relations, $S_1 \in T_1$ and S_2 in T_2 be complex attributes taking values in the domain $\mathbb{S}$ over which the similarity condition is expressed, $\ddot{\theta}$ be the similarity kNN operator, d_1, d_2 be distance functions, $k_1, k_2 \in \mathbb{N}^*$ be similarity thresholds and $s_{q1}, s_{q2} \in \mathbb{S}$ be query centers. Then,

$$\ddot{\sigma}_{\left(S_1\ \ddot{\theta}(d_1,\,k_1)\ s_{q_1}\right)\wedge\left(S_2\ \ddot{\theta}(d_2,\,k_2)\ s_{q_2}\right)}\left(T_1 \times T_2\right) \subseteq \left(\ddot{\sigma}_{\left(S_1\ \ddot{\theta}(d_1,\,k_1)\ s_{q_1}\right)}\ T_1\right) \times \left(\ddot{\sigma}_{\left(S_2\ \ddot{\theta}(d_2,\,k_2)\ s_{q_2}\right)}\ T_2\right)\ . \tag{4.32}$$

Proof. By Definition 4.2, we have that

$$\begin{aligned}(i) &= \ddot{\sigma}_{\left(S_1\ \ddot{\theta}(d_1,\,k_1)\ s_{q1}\right)\wedge\left(S_2\ \ddot{\theta}(d_2,\,k_2)\ s_{q2}\right)}\left(T_1 \times T_2\right) \\ &= \{(t_i, t_j) \in T_1 \times T_2 \mid \forall\ (t_1, t_2) \in T_1 \times T_2 - T', \quad d\left(t_i(S), s_{q1}\right) \leq d\left(t_1(S), s_{q1}\right) \wedge \\ &\qquad\qquad d(t_j(S), s_{q2}) \leq d(t_2(S), s_{q2})\}\ ,\end{aligned}$$

where $T' = \emptyset$ for $i = 1$ or $j = 1$, and $T' = \{(t_1, t_1), (t_1, t_2), \ldots, (t_{i-1}, t_{j-1})\}$ for $1 < i \leq k_1$ and $1 < j \leq k_2$.

As T_1 and T_2 can naturally be identified as subsets of $T_1 \times T_2$, we obtain

$$\begin{aligned}(i) \subseteq \{(t_i, t_j) \in T_1 \times T_2 \mid \forall\, t_1 \in T_1 - T', d\,(t_i(S), s_{q1}) \leq d\,(t_1(S), s_{q1}) \wedge \\ \forall\, t_2 \in T_2 - T', d\,(t_j(S), s_{q2}) \leq d\,(t_2(S), s_{q2})\} \\ = \{t_i \in T_1 \mid \forall\, t_1 \in T_1 - T', d\,(t_i(S), s_{q1}) \leq d\,(t_1(S), s_{q1})\} \times \\ \{t_j \in T_2 \mid \forall\, t_2 \in T_2 - T', d\,(t_j(S), s_{q2}) \leq d\,(t_2(S), s_{q2})\} \\ = \left(\ddot{\sigma}_{(S_1\ \ddot{\theta}(d_1,\, k_1)\ s_{q1})} T_1\right) \times \left(\ddot{\sigma}_{(S_2\ \ddot{\theta}(d_2,\, k_2)\ s_{q2})} T_2\right) = (ii)\ ,\end{aligned}$$

where T' is as above, which proves Equation 4.32. □

Property k4.12 shows the relationship between kNN selection and the traditional join operator. Hence, the result of the join executed between the kNN selection executed over the first relation and the second relation is included in the result of the kNN executed over the join of the relations.

Property k4.12: Let T_1 and T_2 be two relations, $S \in T_1$ be a complex attribute taking values in the domain $\mathbb{S}$ over which the similarity condition is expressed, $\ddot{\theta}$ be the similarity kNN operator, d be a distance function, $k \in \mathbb{N}^*$ be the similarity threshold and $s_q \in \mathbb{S}$ be the query center. Thus,

$$\underbrace{\left(\ddot{\sigma}_{(S\ \ddot{\theta}(d,\, k)\ s_q)}\, T_1\right) \bowtie T_2}_{(i)} \subseteq \underbrace{\ddot{\sigma}_{(S\ \ddot{\theta}(d,\, k)\ s_q)}\, (T_1 \bowtie T_2)}_{(ii)}\ . \tag{4.33}$$

Proof. By the definition of the join operator we have that:

$$\begin{aligned}T_1 \overset{A_i \theta A_j}{\bowtie} T_2 &= \left\{(t_i, t_j) \in T_1 \overset{A_i \theta A_j}{\bowtie} T_2 \mid t_i \in T_1 \wedge t_j \in T_2 \wedge t_i(A_i)\, \theta\, t_j(A_j)\right\} \\ &= \{(t_i, t_j) \in T_1 \times T_2 \mid t_i(A_i)\, \theta\, t_j(A_j)\} = D \subseteq T_1 \times T_2\ .\end{aligned}$$

Replacing in (ii) and considering that S is a attribute in T_1:

$$(ii) = \ddot{\sigma}_{(S\ \ddot{\theta}(d,\, k)\ s_q)} D = \{(t_i, t_j) \in D \mid \forall\ (t_1, t_2) \in D - T', d\,(t_i(S), s_q) \leq d\,(t_1(S), s_q)\}\ ,$$

where $T' = \emptyset$ if $i = 1$ or $j = 1$, and $T' = \{(t_{1_1}, t_{2_1}), \ldots, (t_{1_{i-1}}, t_{2_{j-1}})\}$ if $1 < i, j \leq k$.

As $D \subseteq T_1 \times T_2$, then:

$$\begin{aligned}(ii) &\supseteq \{(t_i, t_j) \in D \mid \forall\, (t_1, t_2) \in T_1 \times T_2 - T', d(t_i(S), s_q) \leq d(t_1(S), s_q)\} \\ &= \{(t_i, t_j) \in T_1 \times T_2 \mid \forall\, (t_1, t_2) \in T_1 \times T_2 - T', d(t_i(S), s_q) \leq d(t_1(S), s_q) \wedge \\ &\qquad\qquad t_i(A_i)\, \theta\, t_j(A_j)\} \\ &= \{t_i \in T_1 \mid \forall\, t_1 \in T_1 - T', d(t_i(S), s_q) \leq d(t_1(S), s_q)\} \bowtie T_2 \\ &= \left(\ddot{\sigma}_{\left(S\ \ddot{\theta}(d,\,k)\ s_q\right)} T_1\right) \bowtie T_2 = (i)\ ,\end{aligned}$$

where T' is as above, proving Equation 4.33. □

A special case exists regarding the relationship of a kNN selection in a conjunctive conditions and the traditional join binary operator. The result of performing the join operation between the kNN selection executed over the left relation and the kNN selection executed over the right relation is included in the result of the conjunction of the kNN executed over the join result. This special case is presented in Property k4.12.1.

Property k4.12.1: Let T_1 and T_2 be two relations, $S_1 \in T_1$ and $S_2 \in T_2$ be complex attributes taking values in the domain $\mathbb{S}$ over which the similarity condition is expressed, $\ddot{\theta}$ be the similarity kNN operator, d_1, d_2 be a distance function, $k_1, k_2 \in \mathbb{N}^*$ be the similarity threshold and $s_{q1}, s_{q2} \in \mathbb{S}$ be the query center. Then,

$$\left(\ddot{\sigma}_{\left(S_1\ \ddot{\theta}(d_1,\,k_1)\ s_{q_1}\right)} T_1\right) \bowtie \left(\ddot{\sigma}_{\left(S_2\ \ddot{\theta}(d_2,\,k_2)\ s_{q_2}\right)} T_2\right) \subseteq \ddot{\sigma}_{\left(\left(S_1\ \ddot{\theta}(d_1,\,k_1)\ s_{q_1}\right) \wedge \left(S_2\ \ddot{\theta}(d_2,\,k_2)\ s_{q_2}\right)\right)} (T_1 \bowtie T_2)\,. \tag{4.34}$$

Proof. Using Join definition and replacing $D = T_1 \bowtie T_2$ in (ii), we have:

$$\begin{aligned}(ii) &= \ddot{\sigma}_{\left(\left(S_1\ \ddot{\theta}(d_1,\,k_1)\ s_{q1}\right) \wedge \left(S_2\ \ddot{\theta}(d_2,\,k_2)\ s_{q2}\right)\right)} D \\ &= \{(t_i, t_j) \in D \mid \forall\, (t_1, t_2) \in D - T', d(t_i(S), s_{q1}) \leq d(t_1(S), s_{q1}) \wedge \\ &\qquad\qquad d(t_j(S), s_{q2}) \leq d(t_2(S), s_{q2})\}\ ,\end{aligned}$$

where $T' = \emptyset$ if $i = 1$ or $j = 1$, and $T' = \{(t_1, t_1), (t_1, t_2), \ldots, (t_{i-1}, t_{j-1})\}$ if $1 < i \leq k_1$ and $1 < j \leq k_2$.

As $D \subseteq T_1 \times T_2$, then:

$$
\begin{aligned}
(ii) \supseteq & \{(t_i, t_j) \in D \mid \forall\, (t_1, t_2) \in T_1 \times T_2 - T', d(t_i(S), s_{q1}) \leq d(t_1(S), s_{q1}) \wedge \\
& \qquad d(t_j(S), s_{q2}) \leq d(t_2(S), s_{q2})\} \\
= & \{(t_i, t_j) \in T_1 \times T_2 \mid \forall\, (t_1, t_2) \in T_1 \times T_2 - T', d(t_i(S), s_{q1}) \leq d(t_1(S), s_{q1}) \wedge \\
& \qquad d(t_j(S), s_{q2}) \leq d(t_2(S), s_{q2}) \ \wedge t_i(A_i)\, \theta\, t_j(A_j)\} \\
= & \{t_i \in T_1 \mid \forall\, t_1 \in T_1 - T', d(t_i(S), s_{q1}) \leq d(t_1(S), s_{q1})\} \bowtie \\
& \{t_j \in T_2 \mid \forall\, t_2 \in T_2 - T', d(t_j(S), s_{q2}) \leq d(t_2(S), s_{q2})\} \\
= & \left(\ddot{\sigma}_{\left(S_1\, \ddot{\theta}(d_1, k_1)\, s_{q1}\right)} T_1\right) \bowtie \left(\ddot{\sigma}_{\left(S_2\, \ddot{\theta}(d_2, k_2)\, s_{q2}\right)} T_2\right) = (i)\ ,
\end{aligned}
$$

where T' is as above. Therefore $(i) \subseteq (ii)$, as required. □

Tables 4.2 and 4.3 summarize all the shown properties of the unary similarity operators. Table 4.2 synthesizes the properties of the range similarity operator, while Table 4.3 summarizes the properties of k-nearest neighbor operator.

Properties of Similarity Range Query	
R4.1	$\hat{\sigma}_{\left(S\, \hat{\theta}(d, \xi)\, s_q\right)} \left(\hat{\sigma}_{\left(S\, \hat{\theta}(d, \xi)\, s_q\right)} T\right) = \hat{\sigma}_{\left(S\, \hat{\theta}(d, \xi)\, s_q\right)} T$
R4.2	$\hat{\sigma}_{\left(S_1\, \hat{\theta}(d_1, \xi_1)\, s_{q1}\right) \wedge \left(S_2\, \hat{\theta}(d_2, \xi_2)\, s_{q2}\right)} T = \hat{\sigma}_{\left(S_1\, \hat{\theta}(d_1, \xi_1)\, s_{q1}\right)} \left(\hat{\sigma}_{\left(S_2\, \hat{\theta}(d_2, \xi_2)\, s_{q2}\right)} T\right)$ $= \left(\hat{\sigma}_{\left(S_1\, \hat{\theta}(d_1, \xi_1)\, s_{q1}\right)} T\right) \cap \left(\hat{\sigma}_{\left(S_2\, \hat{\theta}(d_2, \xi_2)\, s_{q2}\right)} T\right)$
R4.2.1	$\left(\hat{\sigma}_{\left(S\, \hat{\theta}(d, \xi_1)\, s_q\right)} T\right) \cap \left(\hat{\sigma}_{\left(S\, \hat{\theta}(d, \xi_2)\, s_q\right)} T\right) =$ $\hat{\sigma}_{\left(S\, \hat{\theta}(d, \xi_1)\, s_q\right) \wedge \left(S\, \hat{\theta}(d, \xi_2)\, s_q\right)} T = \hat{\sigma}_{\left(S\, \hat{\theta}(d,\, \min(\xi_1, \xi_2))\, s_q\right)} T$
R4.3	$\hat{\sigma}_{\left(S_1\, \hat{\theta}(d_1, \xi_1)\, s_{q1}\right) \vee \left(S_2\, \hat{\theta}(d_2, \xi_2)\, s_{q2}\right)} T = \left(\hat{\sigma}_{\left(S_1\, \hat{\theta}(d_1, \xi_1)\, s_{q1}\right)} T\right) \cup \left(\hat{\sigma}_{\left(S_2\, \hat{\theta}(d_2, \xi_2)\, s_{q2}\right)} T\right)$
R4.3.1	$\left(\hat{\sigma}_{\left(S\, \hat{\theta}(d, \xi_1)\, s_q\right)} T\right) \cup \left(\hat{\sigma}_{\left(S\, \hat{\theta}(d, \xi_2)\, s_q\right)} T\right) =$ $\hat{\sigma}_{\left(S\, \hat{\theta}(d, \xi_1)\, s_q\right) \vee \left(S\, \hat{\theta}(d, \xi_2)\, s_q\right)} T = \hat{\sigma}_{\left(S\, \hat{\theta}(d,\, \max(\xi_1, \xi_2))\, s_q\right)} T$
R4.4	$\hat{\sigma}_{\left(S_1\, \hat{\theta}(d_1, \xi_1)\, s_{q1}\right)} \left(\hat{\sigma}_{\left(S_2\, \hat{\theta}(d_2, \xi_2)\, s_{q2}\right)} T\right) = \hat{\sigma}_{\left(S_2\, \hat{\theta}(d_2, \xi_2)\, s_{q2}\right)} \left(\hat{\sigma}_{\left(S_1\, \hat{\theta}(d_1, \xi_1)\, s_{q1}\right)} T\right)$
R4.5	$\hat{\sigma}_{\left(S\, \hat{\theta}(d, \xi)\, s_q\right)} \left(\sigma_{(A\, \theta\, a)} T\right) = \sigma_{(A\, \theta\, a)} \left(\hat{\sigma}_{\left(S\, \hat{\theta}(d, \xi)\, s_q\right)} T\right)$
R4.6.1	$\hat{\sigma}_{\left(S\, \hat{\theta}(d, \xi)\, s_q\right)} (T_1 \cup T_2) = \left(\hat{\sigma}_{\left(S\, \hat{\theta}(d, \xi)\, s_q\right)} T_1\right) \cup \left(\hat{\sigma}_{\left(S\, \hat{\theta}(d, \xi)\, s_q\right)} T_2\right)$
R4.6.2	$\hat{\sigma}_{\left(S\, \hat{\theta}(d, \xi)\, s_q\right)} (T_1 - T_2) = \left(\hat{\sigma}_{\left(S\, \hat{\theta}(d, \xi)\, s_q\right)} T_1\right) - \left(\hat{\sigma}_{\left(S\, \hat{\theta}(d, \xi)\, s_q\right)} T_2\right)$ $= \left(\hat{\sigma}_{\left(S\, \hat{\theta}(d, \xi)\, s_q\right)} T_1\right) - T_2$
R4.6.3	$\hat{\sigma}_{\left(S\, \hat{\theta}(d, \xi)\, s_q\right)} (T_1 \cap T_2) = \left(\hat{\sigma}_{\left(S\, \hat{\theta}(d, \xi)\, s_q\right)} T_1\right) \cap \left(\hat{\sigma}_{\left(S\, \hat{\theta}(d, \xi)\, s_q\right)} T_2\right)$ $= \left(\hat{\sigma}_{\left(S\, \hat{\theta}(d, \xi)\, s_q\right)} T_1\right) \cap T_2 = T_1 \cap \left(\hat{\sigma}_{\left(S\, \hat{\theta}(d, \xi)\, s_q\right)} T_2\right)$
R4.7	$\hat{\sigma}_{\left(S\, \hat{\theta}(d, \xi)\, s_q\right)} (T_1\, \Theta\, T_2) = \left(\hat{\sigma}_{\left(S\, \hat{\theta}(d, \xi)\, s_q\right)} T_1\right) \Theta\, T_2$, where $\Theta = \times$ or $\bowtie$

Table 4.2: Summary of algebraic properties to range similarity queries.

Properties of Similarity k-Nearest Neighbor Query	
k4.1	$\ddot{\sigma}_{(S\,\ddot{\theta}(d,\,k)\;s_q)}\left(\ddot{\sigma}_{(S\,\ddot{\theta}(d,\,k)\;s_q)}T\right) = \ddot{\sigma}_{(S\,\ddot{\theta}(d,\,k)\;s_q)}T$
k4.2	$\ddot{\sigma}_{(S_1\,\ddot{\theta}(d_1,\,k_1)\;s_{q1})\wedge(S_2\,\ddot{\theta}(d_2,\,k_2)\;s_{q2})}T = \left(\ddot{\sigma}_{(S_1\,\ddot{\theta}(d_1,\,k_1)\;s_{q1})}T\right) \cap \left(\ddot{\sigma}_{(S_2\,\ddot{\theta}(d_2,\,k_2)\;s_{q2})}T\right)$
k4.2.1	$\left(\ddot{\sigma}_{(S\,\ddot{\theta}(d,\,k_1)\;s_q)}T\right) \cap \left(\ddot{\sigma}_{(S\,\ddot{\theta}(d,\,k_2)\;s_q)}T\right)$ $= \ddot{\sigma}_{(S\,\ddot{\theta}(d,\,k_1)\;s_q)\wedge(S\,\ddot{\theta}(d,\,k_2)\;s_q)}T \;=\; \ddot{\sigma}_{(S\,\ddot{\theta}(d,\,\min(k_1,\,k_2))\;s_q)}T$
k4.3	$\ddot{\sigma}_{(S_1\,\ddot{\theta}(d_1,\,k_1)\;s_{q1})\vee(S_2\,\ddot{\theta}(d_2,\,k_2)\;s_{q2})}T = \left(\ddot{\sigma}_{(S_1\,\ddot{\theta}(d_1,\,k_1)\;s_{q1})}T\right) \cup \left(\ddot{\sigma}_{(S_2\,\ddot{\theta}(d_2,\,k_2)\;s_{q2})}T\right)$
k4.3.1	$\left(\ddot{\sigma}_{(S\,\ddot{\theta}(d,\,k_1)\;s_q)}T\right) \cup \left(\ddot{\sigma}_{(S\,\ddot{\theta}(d,\,k_2)\;s_q)}T\right)$ $= \ddot{\sigma}_{(S\,\ddot{\theta}(d,\,k_1)\;s_q)\vee(S\,\ddot{\theta}(d,\,k_2)\;s_q)}T \;=\; \ddot{\sigma}_{(S\,\ddot{\theta}(d,\,\max(k_1,\,k_2))\;s_q)}T$
k4.4	$\left(\ddot{\sigma}_{(S_1\,\ddot{\theta}(d_1,\,k_1)\;s_{q1})}T\right) \cap \left(\ddot{\sigma}_{(S_2\,\ddot{\theta}(d_2,\,k_2)\;s_{q2})}T\right) = \left(\ddot{\sigma}_{(S_2\,\ddot{\theta}(d_2,\,k_2)\;s_{q2})}T\right) \cap \left(\ddot{\sigma}_{(S_1\,\ddot{\theta}(d_1,\,k_1)\;s_{q1})}T\right)$ $\left(\ddot{\sigma}_{(S_1\,\ddot{\theta}(d_1,\,k_1)\;s_{q1})}T\right) \cup \left(\ddot{\sigma}_{(S_2\,\ddot{\theta}(d_2,\,k_2)\;s_{q2})}T\right) = \left(\ddot{\sigma}_{(S_2\,\ddot{\theta}(d_2,\,k_2)\;s_{q2})}T\right) \cup \left(\ddot{\sigma}_{(S_1\,\ddot{\theta}(d_1,\,k_1)\;s_{q1})}T\right)$
k4.5	$\ddot{\sigma}_{(S_1\,\ddot{\theta}(d_1,\,k_1)\;s_{q_1})\wedge(S_2\,\ddot{\theta}(d_2,\,k_2)\;s_{q_2})}T \subseteq \ddot{\sigma}_{(S_1\,\ddot{\theta}(d_1,\,k_1)\;s_{q_1})}\left(\ddot{\sigma}_{(S_2\,\ddot{\theta}(d_2,\,k_2)\;s_{q_2})}T\right)$
k4.6	$\sigma_{(A\,\theta\,a)}\left(\ddot{\sigma}_{(S\,\ddot{\theta}(d,\,k)\;s_q)}T\right) \subseteq \ddot{\sigma}_{(S\,\ddot{\theta}(d,\,k)\;s_q)}\left(\sigma_{(A\,\theta\,a)}T\right)$
k4.7	$\hat{\sigma}_{(S_2\,\hat{\theta}(d_2,\,\xi)\;s_{q2})}\left(\ddot{\sigma}_{(S_1\,\ddot{\theta}(d_1,\,k)\;s_{q1})}T\right) \subseteq \ddot{\sigma}_{(S_1\,\ddot{\theta}(d_1,\,k)\;s_{q1})}\left(\hat{\sigma}_{(S_2\,\hat{\theta}(d_2,\,\xi)\;s_{q2})}T\right)$
k4.8	$\ddot{\sigma}_{(S\,\ddot{\theta}(d,\,k)\;s_q)}\left(T_1 \cup T_2\right) \subseteq \left(\ddot{\sigma}_{(S\,\ddot{\theta}(d,\,k)\;s_q)}T_1\right) \cup \left(\ddot{\sigma}_{(S\,\ddot{\theta}(d,\,k)\;s_q)}T_2\right)$
k4.9	$\left(\ddot{\sigma}_{(S\,\ddot{\theta}(d,\,k)\;s_q)}T_1\right) - T_2 \subseteq \ddot{\sigma}_{(S\,\ddot{\theta}(d,\,k)\;s_q)}\left(T_1 \;-\; T_2\right)$ $\left(\ddot{\sigma}_{(S\,\ddot{\theta}(d,\,k)\;s_q)}T_1\right) - T_2 \subseteq \left(\ddot{\sigma}_{(S\,\ddot{\theta}(d,\,k)\;s_q)}T_1\right) - \left(\ddot{\sigma}_{(S\,\ddot{\theta}(d,\,k)\;s_q)}T_2\right)$
k4.10	$\left(\ddot{\sigma}_{(S\,\ddot{\theta}(d,\,k)\;s_q)}T_1\right) \cap T_2 \subseteq \ddot{\sigma}_{(S\,\ddot{\theta}(d,\,k)\;s_q)}\left(T_1 \cap T_2\right)$ $\left(\ddot{\sigma}_{(S\,\ddot{\theta}(d,\,k)\;s_q)}T_1\right) \cap \left(\ddot{\sigma}_{(S\,\ddot{\theta}(d,\,k)\;s_q)}\,T_2\right) \subseteq \ddot{\sigma}_{(S\,\ddot{\theta}(d,\,k)\;s_q)}\left(T_1 \;\cap\; T_2\right)$
k4.11	$\ddot{\sigma}_{(S\,\ddot{\theta}(d,\,k)\;s_q)}\left(T_1 \times T_2\right) \subseteq \left(\ddot{\sigma}_{(S\,\ddot{\theta}(d,\,k)\;s_q)}T_1\right) \times T_2$
k4.11.1	$\ddot{\sigma}_{(S_1\,\ddot{\theta}(d_1,\,k_1)\;s_{q_1})\wedge(S_2\,\ddot{\theta}(d_2,\,k_2)\;s_{q_2})}\left(T_1 \times T_2\right) \subseteq \left(\ddot{\sigma}_{(S_1\,\ddot{\theta}(d_1,\,k_1)\;s_{q_1})}\,T_1\right) \times \left(\ddot{\sigma}_{(S_2\,\ddot{\theta}(d_2,\,k_2)\;s_{q_2})}\,T_2\right)$
k4.12	$\left(\ddot{\sigma}_{(S\,\ddot{\theta}(d,\,k)\;s_q)}T_1\right) \bowtie T_2 \subseteq \ddot{\sigma}_{(S\,\ddot{\theta}(d,\,k)\;s_q)}\left(T_1 \bowtie T_2\right)$
k4.12.1	$\left(\ddot{\sigma}_{(S_1\,\ddot{\theta}(d_1,\,k_1)\;s_{q_1})}\,T_1\right) \bowtie \left(\ddot{\sigma}_{(S_2\,\ddot{\theta}(d_2,\,k_2)\;s_{q_2})}\,T_2\right) \subseteq \ddot{\sigma}_{(S_1\,\ddot{\theta}(d_1,\,k_1)\;s_{q_1})\wedge(S_2\,\ddot{\theta}(d_2,\,k_2)\;s_{q_2})}\left(T_1 \bowtie T_2\right)$

Table 4.3: Summary of algebraic properties to kNN similarity queries.

Table 4.4 presents the equivalence properties valid over the traditional operators, but that are not valid when it involves similarity kNN similarity ones. These invalid properties for kNN similarity operators can be easily proved by contradiction, using counterexamples.

4.6 Semantic Restrictions

Semantic restrictions such as "external" knowledge about stored data or about the user's expectation regarding the data or the query answer are important factors that can help improving the performance of query execution. Those factors are seldom employed to optimize queries over scalar data, because it is very costly to process them. In fact,

	Properties of Traditional Query
T1	$\sigma_{(A_1\,\theta_1\,a_1)\,\wedge\,(A_2\,\theta_2\,a_2)}\,T = \sigma_{(A_1\,\theta_1\,a_1)}\left(\sigma_{(A_2\,\theta_2\,a_2)}\,T\right)$
T2	$\sigma_{(A_1\,\theta_1\,a_1)}\left(\sigma_{(A_2\,\theta_2\,a_2)}\,T\right) = \sigma_{(A_2\,\theta_2\,a_2)}\left(\sigma_{(A_1\,\theta_1\,a_1)}\,T\right)$
T3	$\sigma_{(A\,\theta\,a)}\,(T_1 \cup T_2) = \left(\sigma_{(A\,\theta\,a)}\,T_1\right) \cup \left(\sigma_{(A\,\theta\,a)}\,T_2\right)$
T4	$\sigma_{(A\,\theta\,a)}\,(T_1 - T_2) = \left(\sigma_{(A\,\theta\,a)}\,T_1\right) - T_2 = \left(\sigma_{(A\,\theta\,a)}\,T_1\right) - \left(\sigma_{(A\,\theta\,a)}\,T_2\right)$
T5	$\sigma_{(A\,\theta\,a)}\,(T_1 \cap T_2) = \left(\sigma_{(A\,\theta\,a)}\,T_1\right) \cap T_2$ $= T_1 \cap \left(\sigma_{(A\,\theta\,a)}\,T_2\right) = \left(\sigma_{(A\,\theta\,a)}\,T_1\right) \cap \left(\sigma_{(A\,\theta\,a)}\,T_2\right)$
T6	$\sigma_{(A\,\theta\,a)}\,(T_1\,\Theta\,T_2) = \left(\sigma_{(A\,\theta\,a)}\,T_1\right)\,\Theta\,T_2 = T_1\,\Theta\,\left(\sigma_{(A\,\theta\,a)}\,T_2\right)$, where $\Theta = \times$ or $\bowtie$
T6.1	$\sigma_{(A_1\,\theta_1\,a_1)\wedge(A_2\,\theta_2\,a_2)}\,(T_1\,\Theta\,T_2) = \left(\sigma_{(A_1\,\theta_1\,a_1)}\,T_1\right)\,\Theta\,\left(\sigma_{(A_2\,\theta_2\,a_2)}\,T_2\right)$, where $\Theta = \times$ or $\bowtie$

Table 4.4: Summary of algebraic equivalence properties to traditional queries invalid to kNN similarity queries.

the cost to handle semantic restrictions is often higher than the cost of just executing the non-optimized query over scalar data, thus it makes no sense utilizing semantic restriction just to optimize queries over scalar data. However, when handling complex data and similarity queries, there are at least two factors that revert the weights of those costs.

Firstly, the similarity query execution requires inherently much more time than it is required to process traditional operators, and the same occurs when handling complex data. Therefore, the tradeoff between using or not semantic restrictions becomes more favorable to using it when similarity operators are involved than when only traditional ones are involved. In fact, when complex data must be searched by similarity, using semantic restriction almost always leads to the best alternative to improve efficiency. Therefore, using semantic restrictions becomes clearly more attractive even if only the performance gains to improve the execution performance of queries posed over complex data are considered. Moreover, using semantic restrictions about the data to identify the regions of the data space where the answer should be searched, pruning those where answers cannot be found is a good strategy to improve the performance of similarity queries.

Secondly, it must be considered that searching for similarity often involves a certain degree of variability between what is the expressed by the exact, formal expression of the similarity (that is processed by the similarity query algorithms, extracted features and distance functions) and what is expected by the user. Although that dichotomy between the formal expression of a query and the user's expectation can happen when processing scalar data with identity- and TOR-based operators, it is clearly larger when processing complex data with similarity-based operators. Therefore, taking into account semantic restrictions about the user's preference when answering a query often can improve not only the query performance, but also (and more importantly) the query quality to fulfill the user's expectation and thus the query efficiency.

In summary, whenever a similarity query is posed, it is worth to the optimizer to take into account the known semantic restrictions of the data and of the user's expectation

to evaluate a query whenever any of them can be employed either to improve the query quality (tailoring the execution to better meet the users' expectations), or to improve the query execution efficiency (using the restrictions as screening predicates to prune part of the data that is known to have no interesting data). Semantic restrictions are always translated into predicates, then they can be employed as filter in query refinement techniques.

Subsection 4.6.1 explores the usage of the known users' interests as semantic restrictions to optimize similarity queries. Subsection 4.6.2 investigates how to utilize the knowledge about the stored complex data, which was retrieved and analyzed during the execution of previous queries, using the mined pattern also as semantic restrictions to rewrite further similarity queries.

4.6.1 Preference Model Module

The preferences of each user determine its choices. Hence, the knowledge of that preferences should be obtained from the user creating a "user's profile", preferably imposing little or no burden over the user, for example employing 'relevance feedback' techniques or other non-intrusive analysis performed by the system regarding the set of queries and respective answers posed by the user. The Preference Model module proposed in this monograph receives the gathered users' interests and generates preference rules, using a preference model such as one of those mentioned in Section 3.8, and correlates the semantic information both from the user and from the complex data. Together with similarity algebra, preference rules can be used to rewrite similarity queries, obtaining results closer to the users' expectations in a faster way.

Figure 4.6 shows the process of preparing and executing queries either using or not the preference model in RDBMS. In the first step (Figure 4.6, Action ①), users' preferences are obtained from users' profiles, for example. Then, the preference model module generates preference rules, storing them in the database as part of the system catalog. When the user searches for similarity without using the preference model (Figure 4.6, Action ③), the similarity database uses only the syntax-based optimization to answer the query. Otherwise, when the preference model is used to evaluate a query (Figure 4.6, Action ②), the similarity database combines the syntax- and the semantic-based techniques to rewrite queries.

For illustration purposes, let us use again the CoPhIR database. Suppose that the user wants to find the 5 beach photos most similar to his photo, so that they were obtained in the hottest seasons. If the user's background knows how to qualify one season as hotter than another, than his 'hotness scale' should be used to refine the query, bringing the query answer closer to what the user expects. This is the way that we developed the

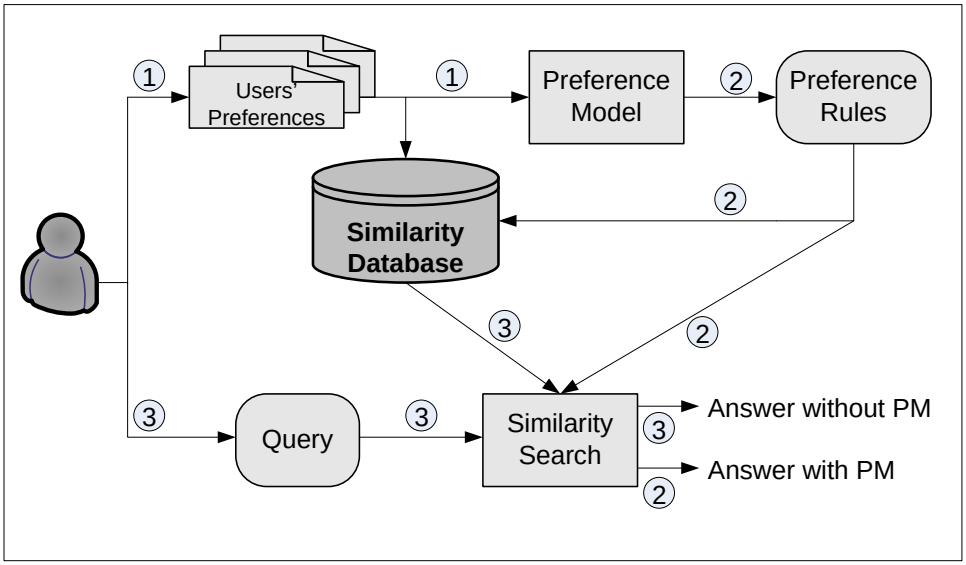


Figure 4.6: Generic flowchart to prepare and execute similarity queries considering of preference models.

similarity queries to use preferences rules and similarity algebra to obtain results closer to users' expectations.

4.6.2 Data Mining Model Module

One way to integrate DBMS and data mining techniques is to incorporate mining tools into the database engine. Accordingly, the Data Mining Model module tightly couples data mining algorithms, thus allowing similarity queries to benefit from automatically mined rules and improving both query quality and performance. The data mining model module extracts knowledge from the stored complex data retrieved by previous queries, generating mining rules. Combining the similarity algebra and the mining rules found, similarity queries can be rewritten to make them more semantically adequate to meet the users' expectations.

Figure 4.7 shows the process of preparing and posing queries either using or not the mined rules in RDBMS. In the first step (Figure 4.7, Action ①), the user sends stored data to the data mining model module. Then, mining rules are mined and stored, together with the other data, in the database (Figure 4.7, Action ②). When the user searches for similarity without using the data mining model (Figure 4.7, Action ③), the similarity database uses only the syntax-based optimization to answer the query. Otherwise, when the data mining model is used to evaluate a query (Figure 4.7, Action ②), the similarity database combines the syntax- and the semantic-based techniques to rewrite queries.

Those ideas can be explained through an example. Let us use the CoPhIR database again. Suppose that the user wants to find beach photos similar to his photo that are like his photos from tropical climate beaches. If the user has a small collection of photos

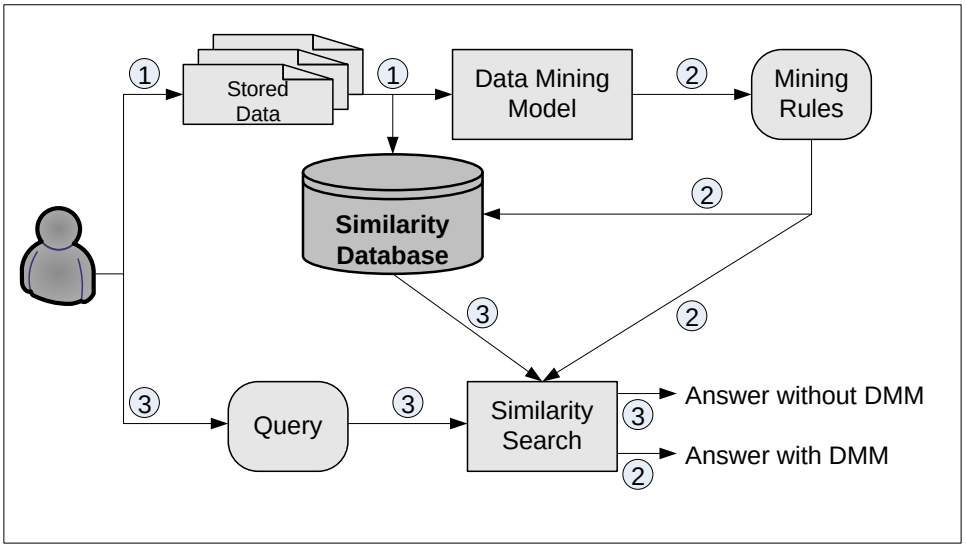


Figure 4.7: Generic flowchart to prepare and execute similarity queries considering of data mining models.

"from tropical climate beaches", he/she can use data mining techniques integrated with similarity queries to express queries that employ the results of the mining algorithms. After a data mining algorithm extracts rules that can express how to identify "photos from tropical climate beaches", the resulting rules are stored in the database, as part of the system catalog. Thereafter, the mined rules can be automatically combined to the user's expressions through similarity algebra operators to evaluate the similarity queries.

4.7 Final Comments

In this chapter we presented the techniques developed in this doctorate program to incorporate similarity queries into RDBMSs. First, similarity-based operators were included into relational model, allowing the management of complex and scalar data in an integrated way. We also presented an algorithm to generate the algebraic canonical plan, which includes every basics steps to translate traditional and similarity queries into an algebraic expression. We developed that algorithm in such a way that when k-nearest neighbors predicates are involved, the answer includes as closer to k elements as possible, aiming at pursuing to obtain the k elements that the user expects. Thereafter, the canonical plan is sent to the query optimizer. The query optimizer was extend to handle the syntax and the semantic of queries that include similarity operators. The syntax-based extension proposed includes a large number of rules that govern the similarity-based comparison operators and its integration with the existing identity and by relational comparison ones. The proposed algebra provides a powerful and flexible basis to develop semantic-based extensions that can both mine patterns and knowledge about the stored

data and identify the user's expectation about the query result content, taking them into account to improve the efficiency and the efficacy of similarity queries processing.

In Chapter 5, we discuss how these techniques were included in the SIREN similarity retrieval prototype that we are developing to include similarity queries in RDBMS taking into account the semantic of the data and the user's expectation.

Chapter

5

Similarity Retrieval Engine - Case Study

5.1 Introduction

Originally proposed by Barioni et al. [2006], the Similarity Retrieval Engine (SIREN) is a middleware between the application program and a traditional database management system (DBMS) that allows the execution of similarity queries in structured query language (SQL). In its initial version, SIREN was able to compile and directly execute similarity queries over complex data stored in relational database management systems (RDBMSs), without caring about similarity query optimization process. Currently, SIREN is being extended with the techniques presented in Chapter 4, by adding the query optimization process to optimize similarity queries, both based solely on the algebraic properties of the similarity-based operators as well as meeting user's expectation. Figure 5.1 shows the current SIREN architecture, which we called SIREN$_{op}$. In this figure, our extension is highlighted in blue.

SIREN language is based on a SQL extension that includes several new data types corresponding to complex data, such as `STILLIMAGE`, `AUDIO` and `PARTICULATE`. It also supports similarity predicates and special constructs to allow the definition of how features are extracted from complex data, how to define distance functions and how both, extracted features and distance functions are assigned to the complex attributes declared as composing the relations.

A similarity-based predicate is expressed in the same way as the existing predicates in the `WHERE` clause of the `SELECT` command, and it allows expressing all the elements of a similarity selection operator of the format $\boxed{\sigma_{c\,(S\;\theta_c(d,\,lim)\;s_q)}\;T}$ described in the previous chapter. The basic syntax to express similarity predicates in the `WHERE` clause is the following:

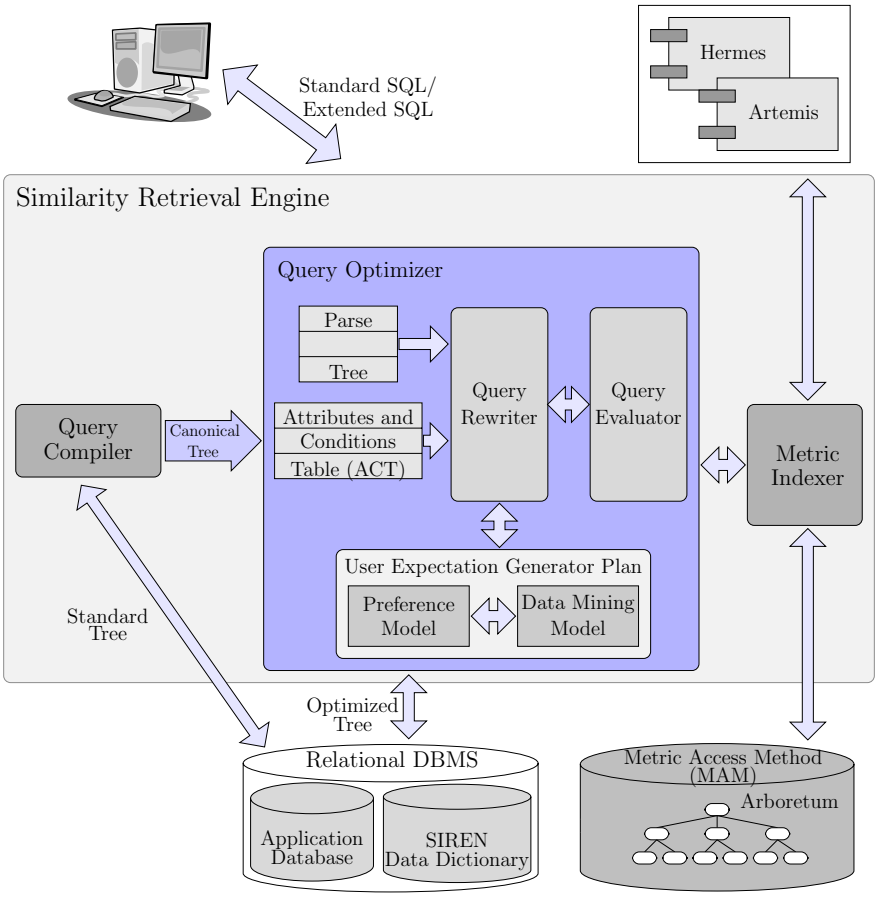


Figure 5.1: SIREN$_{op}$ architecture.

`<attr> NEAR <value> [STOP AFTER <`k`>] [RANGE <`ξ`>] [USING <distf>]`,

where `<attr>` is the complex attribute (corresponding to S), `<value>` must be an element of the corresponding complex domain (corresponding to s_q), `<`k`>` and `<`ξ`>` are the similarity threshold for k-nearest neighbors and range queries respectively (corresponding to lim), and `<distf>` is the distance function. The reserved word `NEAR` specifies that this is a similarity-based predicate (it corresponds to the generic similarity operator θ_c), whereas the reserved words `STOP AFTER` and `RANGE` specify the operator to be either a k-nearest neighbors or a range one respectively, and corresponding to either the specific $\ddot{\theta}$ or $\hat{\theta}$ operators respectively. In SIREN, `<`k`>`, `<`ξ`>` and `<distf>` are optional, and if not specified they are assumed to be 1, 1 and the single (or the one set as default) distance function defined for the attribute `<attr>`. SIREN acts as a blade between the DBMS and the application program. It intercepts and analyzes every query sent by the user to the DBMS. In case that the command has neither similarity-related operations nor references to complex data, the command is directly relayed to the DBMS, and thus SIREN is transparent to traditional operations. On the other hand, if the query has similarity-related constructions, then SIREN compiles, optimizes and executes it, sending the answer back to the user.

In this chapter, we use seven datasets obtained from real applications to illustrate the application of the concepts and to report the result of the experiments that we performed using the new version of SIREN that we extended. Table 5.1 summarizes datasets used in the experiments of this thesis.

The `DDSM_DS` dataset

This is a set of 4,612 mammography images, obtained between 1993 and 1999 from the Digital Database for Screening Mammography (DDSM) website[1] [Heath et al., 1998, 2000]. This dataset is composed of two relations: the first stores the images and metadata specific of each image, such as the exam that includes the image; the second stores data related to the exams, such as the data it was performed, etc. The schema of those relations follows.

`Mammography` = {`CaseId`, `Report`, `View`, `ImgRoi`}
`Cases` = {`Id`, `DateOfStudy`, `DateOfDigitized`, `PatientAge`, `Density`, `Hospital`}

The similarity between elements of the complex attribute `ImgRoi` is computed by the pair Haralick [Haralick et al., 1973] feature extractor and the Manhattan (L_1) distance function.

The `MammographyDS` dataset

This is a set of 1,353 medical images obtained from the Clinical Hospital at Ribeirão Preto of the *Universidade de São Paulo*. This dataset is composed of two relations: the first called `RCCMammography`, which has 658 images from mammograms exams of right breast with cranio-caudal view (CC), and the second called `RMLOMammography` that has 695 images from mammograms exams of right breast with medio-lateral oblique view (MLO). The similarity between elements of the image complex attribute is computed by the pair Texture [Felipe et al., 2003] feature extractor and the Manhattan (L_1) distance function.

The `MedImageDS` dataset

This dataset is composed of the relation `MedImage`, which has 5,180 computerized tomographies (CT) images from three human body parts (abdomen, cranium and thorax), obtained from the Clinical Hospital at Ribeirão Preto of the *Universidade de São Paulo*. The pair metric histogram [Traina et al., 2003] feature extractor and the Manhattan (L_1) distance function are used to compute the similarity between elements of the image complex attribute.

[1] DDSM: Digital Database for Screening Mammography Homepage. Accessed in: May 15, 2011. Available at: `http://marathon.csee.usf.edu/Mammography/Database.html`

The PeruDS dataset

This dataset is composed of the relation PeruDistrict that has 1,829 Peruvian districts obtained from *Peru Instituto Nacional de Estadística e Informática* (INEI). The complex attribute Coordinate is obtained from the combination of two geographical points, represented in the traditional attributes Lat (i.e. the Latitude) and Long (i.e. the Longitude). For elements of complex attribute Coordinate, the Euclidean (L_2) distance function is defined over their domain, so the similarity predicates can be answered over it.

The CitiesUSDS dataset

This dataset is composed of the relation USCities that has 25,374 American cities and their economic characteristics in Census 2000, obtained from U.S. Census Bureau website[2]. The schema of this relation follows.

USCities = {CityCode, CityName, State, Employed, Unemployed, WorkedAtHome, Retail, PerCapita, PctPovertyFam, ..., Lat, Long, Coordinate}

The complex attribute Coordinate is obtained from the combination of two geographical points, represented in the traditional attributes Lat and Long. For elements of complex attribute Coordinate, the Euclidean (L_2) distance function is defined over their domain, so the similarity predicates can be answered over it. In order to make easier to understand the experimental evaluation presented in Subsection 5.2.1, we assume that there is a function Coord(USCities.CityName), which returns the Coordinate of the city named CityName.

The LungDS dataset

This dataset is composed of the relation LungExam that has 246 lung images collected in 108 distinct computed tomography exams from the Clinical Hospital at Ribeirão Preto of the *Universidade de São Paulo* patients. The exams were separated according to their description and each image were classified by a radiologist into six distinct class (Consolidation, Emphysema, Interlobular Septal Thickening, Honeycombing, Ground-glass Opacity and Normal), in average 40 images per class, according to the radiological finding contained in each image. The similarity between elements of the image complex attribute is computed by the pair Texture [Felipe et al., 2003] feature extractor and the Manhattan (L_1) distance function.

[2]U.S. Census Bureau Homepage. Accessed in: 2011 May 15. Available at: http://www.census.gov/

Dataset Name	Cardinality	Feature Extractor	Distance Function	Description
DDSM_DS	4,612	Haralick	L_1	A set of mammography images obtained between 1993 and 1999 from the Digital Database for Screening Mammography (DDSM) website [Heath et al., 1998, 2000].
MammographyDS	1,353	Texture	L_1	A set of 658 images obtained from mammograms exams of right breast with cranio-caudal view and 695 images obtained from mammograms exams of right breast with medio-lateral oblique view.
MedImageDS	5,180	Metric Histogram	L_1	A set of medical images obtained from three human body parts (abdomen, cranium and thorax) by computerized tomographies.
PeruDS	1,829	–	L_2	A set of the Peruvian districts.
CitiesUSDS	25,374	–	L_2	A set of the American cities and their economic characteristics in Census 2000.
LungDS	246	Texture	L_1	A set of lung images collected in 108 distinct computed tomography exams and separated according to their description (Consolidation, Emphysema, Ground-glass Opacity, Interlobular Septal Thickening, Honeycombing and Normal).
WWorldDS	1,798	Texture	L_1	A set of the fourteen wonder worlds images extracted from the Flickr website, together with their metadata information.

Table 5.1: Real dataset descriptions used in the experiments.

The WWorldDS dataset

This dataset is composed of a relation WondersWorld, which has 1,798 images extracted from the Flickr[3] website, together with their metadata information such as: tags, a short

[3]Flickr Homepage. Accessed in: 2012 August 14. Available at: http://www.flickr.com/

description, the city with the corresponding latitude and longitude. These images are from the fourteen wonders, seven of the ancient and seven of the new world, plus the complex of Giza pyramid, the nowadays remaining wonder from the ancient world. There are 100 images from each of the fourteen distinct wonders and 100 more for the current Giza pyramids, retrieved from Flickr, together with the descriptions that people in general stores in that website. No structure or consistency are expected to exist in this tag system. The schema of this relation is as follows.

`WondersWorld` = {`ImageID`, `Tag`, `Description`, ..., `Training`, `Image`}

The similarity between elements of the complex attribute `Image` is computed by the pair Texture [Felipe et al., 2003] feature extractor and the Manhattan (L_1) distance function. From 1,798 images, 1,500 had the attribute `Training` set to `'False'`. The remaining 298 are images from all of the fourteen wonders and the current Giza pyramids, all of them have the attribute `Training` set to `'True'`, and correspond to examples of the kind of images that the users expects to obtain from each wonder.

Section 5.2 describes the SIREN Query Optimizer. Section 5.3 presents the SIREN Preference Model and Section 5.4 presents the SIREN Data Mining Model. Section 5.5 makes the final comments of this chapter.

5.2 The SIREN Query Optimizer

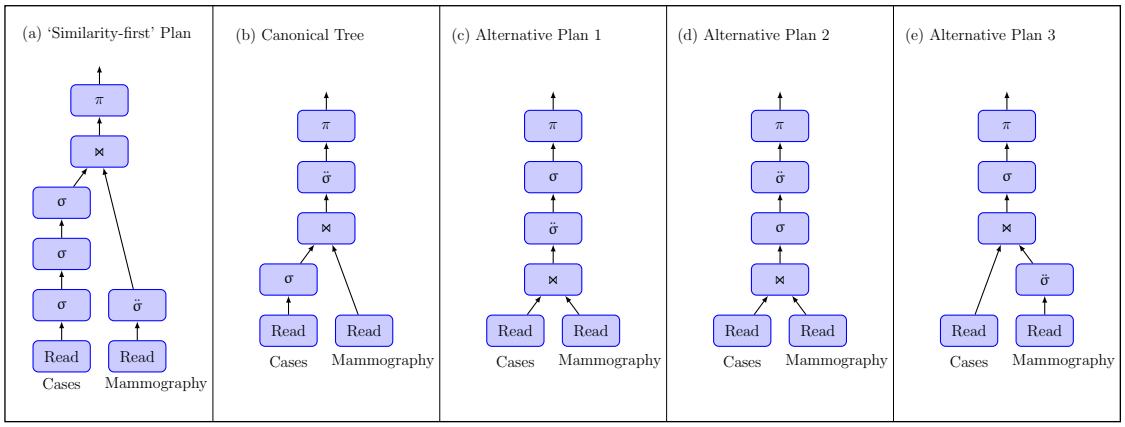
After the compilation process has been successfully executed, the canonical tree is generated, using the Algorithm 4.1 presented in Section 4.3. For illustration, suppose that, in a health-care information system and using the `DDSM_DS` dataset, a medical doctor wants to search for mammographies similar to those of her actual patient whereas specifying some special constraints, as presented in Query Q3.

Example 5.1:

Q3: "Select the 3 mammographies taken in 1993 that are the most similar to this one from my current patient (Patient X), obtained from a patient that is less then 45 years old and whose exam was taken in the Massachusetts General Hospital (MGH)".

This query can be expressed in SQL as:

```
SELECT Cases.Id, Mammography.ImgRoi
  FROM Mammography, Cases
  WHERE Mammography.IdCases = Cases.Id
    AND Cases.DateOfStudy BETWEEN '01/01/1993' AND '31/12/1993'
    AND Mammography.ImgRoi NEAR ImgRoi(PatientX) STOP AFTER 3
    AND Cases.PatientAge < 45
    AND Cases.Hospital = 'MGH'
```



Alternative Plan - Properties:

None	Property k4.12	Equation 2.2 and Property k4.12	Property k4.12 and Equation 2.2	Equation 2.2

Number of expected (k) / returned results:

3/1	3/3	3/1	3/3	3/1

Query execution time (ms):

126	116	200	117	136

Figure 5.2: 'Similarity-first', canonical, alternative plans and execution time of Query Q3.

This query involves traditional identity (`Hospital='MGH'`), traditional total ordering relationships (`AGE > 45`, `BETWEEN`), a kNN selection (`ImgRoi NEAR` s_q `STOP AFTER 3`), as well as a traditional join. Algebraically, it is represented as:

$$\sigma_{(\texttt{DateOfStudy BETWEEN '01/01/1993' AND '31/12/1993'} \wedge \texttt{PatientAge<45} \wedge \texttt{Hospital='MGH'})}\,\texttt{Cases} \bowtie \left(\ddot{\sigma}_{\texttt{ImgRoi}\,\ddot{\theta}(\texttt{Texture,3})\,\texttt{Image(PatientX)}}\,\texttt{Mammography}\right) .$$

Executing the Algorithm 4.1 over Query Q3, the canonical tree obtained is presented in Figure 5.2(b). Figure5.2(a) shows the well-accepted 'similarity-first' plan, which first executes kNN predicates and then the other ones. Table 5.2 illustrates the 'similarity-first' plan shown as a relation. It corresponds to the direct translation of the relational query

1	Read	`Mammography`		
2	Read	`Cases`		
3	σ	2		`DateOfStudy BETWEEN '01/01/1993' AND '31/12/1993'`
4	σ	3		`PatientAge < 45`
5	σ	4		`Hospital='MGH'`
6	$\ddot{\sigma}$	1		`ImgRoi NEAR ImgRoi(PatientX) STOP AFTER 3`
7	$\bowtie$	5	6	`Mammography.IdCases = Cases.Id`
8	π	7		`Cases.Id, Mammography.ImgRoi`

Table 5.2: Canonical plan, represented as a table, of the Query Q3.

into the corresponding algebra expression. It can be seen that the selection operators are

applied in the same sequence expressed in the query command, as soon as possible over each relation.

In the sequence, the canonical tree is sent as input to the SIREN Query Optimizer. The SIREN Query Optimizer processes the expression generating several alternatives applying transformations based on the properties existing for traditional predicates and on the properties of the Similarity Algebra presented in Section 4.5, which are embedded in the Query Rewriter.

Considering Query Q3, Property k4.12 and Equation 2.2 can be used to rewrite the canonical tree and to generate alternative plans as shown in Figures 5.2(c) to (e). To alleviate drawing the alternative plans in the figure, and without loss of generality, we show here the conjunction of only traditional selections transformed into a single selection. The 'Similarity-first' Plan and the Alternatives 1 and 3 can return less than k tuples, as further filtering operations are applied over the first k tuples selected, which can prune even more results. When the evaluation of the predicates of the other operations returns at least k tuples, executing kNN as the last operation warrants that the asked amount k of tuples is returned, as it occurs in the Canonical Tree and Alternative 2 that return the same number k of tuples. Figure 5.2 also shows the minimum and maximum numbers of results that each alternative plan can return, as well as the average wall clock time required for SIREN to process Query Q3 over the `DDSM_DS` dataset with several query centers. As the number $k = 3$ employed in this example is very small, the performance changes derived from each alternative plan are tiny. However, it nevertheless can be seen that the Canonical Tree and Alternative Plan 2 have a gain of about 8% when compared to the 'Similarity-first' Plan. It shows that using the inclusion properties to optimize similarity queries, we can recover the desired number of answers without compromising the required time processing.

The SIREN Query Optimizer can also use association rule mining and users' preferences during the optimization process to identify semantic restrictions, and exploit it as query refinements to improve query efficiency and efficacy. The optimization using association rules is described in Section 5.4 and using users' preferences in Section 5.3.

5.2.1 Experimental Evaluation

Aiming at further evaluating the concepts and the implemented tools related to Example 5.1, the following two set of experimental evaluations were also performed.

Equivalence-based Properties

The first set of experiments was performed using only the equivalence-based properties of the Similarity Algebra. Those properties were incorporated into a version of the SIREN query optimizer called 'SIREN$_{op}$' (SIREN + Optimization) in the experiments. The

experiments analyze the performance of SIREN and SIREN$_{op}$ to execute similarity queries. Both versions of SIREN were implemented in C++. The experiments were executed on an AMD Athlon XP 3000+ processor with 1024MB of main memory, under the Windows XP operational system. The RDBMS employed was Oracle 9i. Every test was performed using both sequential scan and a Slim-tree index. Four data sets were used to pose Queries Q4, Q5 and Q6 to SIREN.

Query Q4 was performed in the `MammographyDS` dataset.

Q4: "Given a mammography exam with images of left and right breast from cranio-caudal (RCC) and medio-lateral oblique (RMLO) views of a patient, show the exams whose mammogram texture does not differ more than 10 units from those in the original exam".

Query Q4 involves a traditional join and a range selection and its canonical algebraic expression obtained using Algorithm 4.1 is represented as $\hat{\sigma}_{\left(S\,\hat{\theta}(texture,\,0.1)\;s_q\right)}\,(RCC \bowtie RMLO)$. Property R4.7 was employed to optimize the query. Its optimized expression is $\left(\hat{\sigma}_{\left(S\,\hat{\theta}(texture,\,0.1)\;s_q\right)} RCC\right) \bowtie RMLO$.

Query Q5 was performed in the `MedImageDS` dataset.

Q5: "Given a head tomography exam of a patient showing a pathology, retrieve the 5 most similar exams not presenting a pathology, and whose texture does not differ more than 5 units from those in the target exam".

Query Q5 involves traditional selection, range selection and kNN selection. Applying Algorithm 4.1, the canonical algebraic expression is represented as: $\ddot{\sigma}_{\left(S\,\ddot{\theta}(texture,\,5)\;s_q\right)}\,\left(\hat{\sigma}_{\left(S\,\hat{\theta}(texture,\,0.05)\;s_q\right)}\,\left(\sigma_{(pathology=\text{`}N\text{'})}\,(MedImage)\right)\right)$. Properties R4.5 and k4.4 as well as their special cases should be used to optimize this query. One algebraic plan of Query Q5 can be algebraically expressed as: $\sigma_{(pathology=\text{`}N\text{'})}\,\left(\hat{\sigma}_{\left(S\,\hat{\theta}(texture,\,0.05)\;s_q\right)}\,\left(\ddot{\sigma}_{\left(S\,\ddot{\theta}(texture,\,5)\;s_q\right)}\,(MedImage)\right)\right)$.

Query Q6 was performed in the `PeruDS` dataset.

Q6: "Find the 15 districts nearest to 'Arequipa' that are not farther than 15 miles, and where the population between 21 and 64 years old is greater than the over 65 year old population".

Query Q6 also involves traditional selection, range selection and kNN selection. Applying Algorithm 4.1, the canonical algebraic expression is expressed as: $\ddot{\sigma}_{\left(S\,\ddot{\theta}(Euclidean,\,15)\;s_q\right)}\,\left(\hat{\sigma}_{\left(S\,\hat{\theta}(Euclidean,\,15)\;s_q\right)}\,\left(\sigma_{(adultpop\,>\,oldpop)}\,(PeruDistricts)\right)\right)$. Properties R4.5 and k4.4 as well as their special cases should be used to optimize these queries. One of the alternative plan of the Query Q6 can be expressed as: $\hat{\sigma}_{\left(S\,\hat{\theta}(Euclidean,\,15)\;s_q\right)}\,\left(\sigma_{(adultpop\,>\,oldpop)}\,\left(\ddot{\sigma}_{\left(S\,\ddot{\theta}(Euclidean,\,15)\;s_q\right)}\,(PeruDistricts)\right)\right)$.

The experiments evaluated the execution time of these three queries, and the results were compared for correctness. The queries were performed 30 times and the values shown are the average of performing the same query varying the query center s_q. Table 5.3 summarizes measurements executing the three queries using SIREN and SIREN$_{op}$ both using sequential scan and using the Slim-tree index structure.

As we can see in Table 5.3, the optimization process can make Query Q4 about 30% faster both when a sequential scan or a Slim-tree index is employed. The gain obtained with Query Q5 was about 65% using a Slim-tree index and 63% using sequential scan. Query Q6 gain was about 64% using sequential scan and about 63% when using a Slim-tree index.

	SIREN		SIREN$_{op}$	
	Sequential scan	Slim tree	Sequential scan	Slim tree
Q4	354.70	331.20	246.90	231.30
Q5	948.50	765.60	351.70	270.40
Q6	604.70	443.20	218.80	165.70

Table 5.3: Performance of Queries **Q**4, **Q**5 and **Q**6 (total time in milliseconds).

The experiments reported in Table 5.3 show that the performance gains obtained by the optimization techniques are always greater than those obtained just using the metric access method. Moreover, using both techniques, optimization + MAM always leads to the better performance.

Inclusion-based Properties

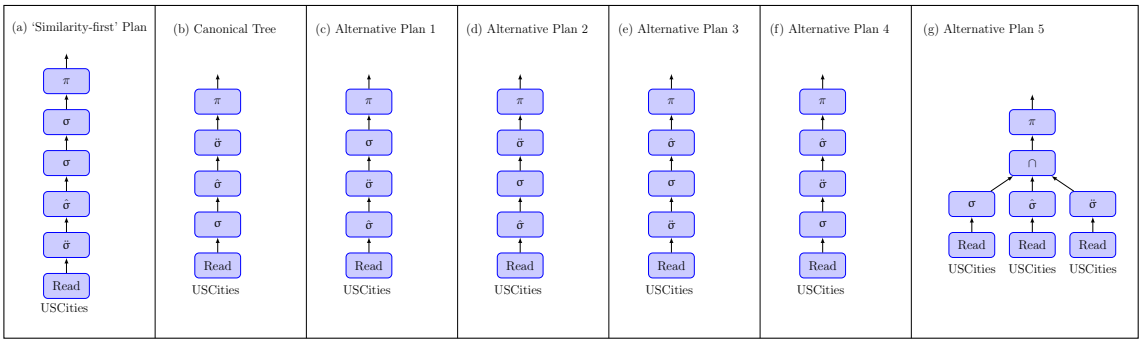
The second set of experiments was performed using the equivalence- and inclusion-based properties of the Similarity Algebra. SIREN was implemented in C++, and the experiments were evaluated using an Intel Core 2 Quad 2.83GHz processor with 4GB of main memory, under the Windows XP operational system. SIREN was configured to process the traditional part of the queries in Oracle 9i. These experiments apply the 'similarity-first', canonical and the alternative plans to execute Query Q7, using the `CitiesUSDS` dataset. Remember that the 'similarity-first' plan is the one in which the kNN predicates should be the first to be executed.

Q7: "Find the 5 cities nearest to 'New York city-NY', whose distances from 'Albany city-NY' are not farther than 210 km, considering the Euclidean distance L_2, having the per capita income greater than 22,400 and the percentage of families in poverty level smaller than or equal to 18.5".

Query Q7 involves traditional, similarity range and kNN selections. It can be expressed as follows.

$$\sigma_{(PerCapita>22400 \wedge PctPovertyFam\leq 18.5)}(\hat{\sigma}_{(\texttt{Coordinate}\ \hat{\theta}(L_2, 1.9)\ Coord(\text{Albany})}(\ddot{\sigma}_{\texttt{Coordinate}\ \ddot{\theta}(L_2,5)\ Coord(\text{New York})}\ USCities)) \ .$$

Figure 5.3 presents the 'similarity-first', the canonical, which is the result of applying the Algorithm 4.1 in the original query, and five alternative execution plans that result from the query rewriting processing of Query Q7. This figure also shows the number of expected (k) and returned results for each plan, and their evaluation time in milliseconds (ms). The time reported corresponds to the average of execution of 10 queries like Q7 for distinct query centers.



Alternative Plan - Properties:						
None	Property R4.5, Property k4.6 and Property k4.7	Property k4.7	Property k4.7 and Property k4.6	Property R4.5	Property R4.5 and Property k4.6	Property k4.4
Number of expected (k / returned results:						
5/1	5/5	5/3	5/5	5/1	5/4	5/1
Query execution time (ms):						
117	117	125	141	124	125	403

Figure 5.3: 'Similarity-first' plan, canonical tree, alternative plans and execution time of Query Q7.

Let us compare the results and how the 'similarity-first' and the canonical plans differ. For this query, the evaluation time of all alternative plans are greater than the 'similarity-first' and canonical ones. As the optimization process goal is to identify the algebraic expression that may be evaluated with the lowest computational cost, the alternative plans are discarded by the query optimizer.

Although the evaluation times for the 'similarity-first' and for the Canonical plans are the same, the 'similarity-first' plan can return less than k tuples, as further selection operations are applied over the first k tuples selected, pruning more results. When the evaluation of the predicates of the remaining selections after returns at least k tuples, executing the kNN as the last operation (Canonical tree) warrants that the asked amount k of tuples are returned. However, it is worth noticing that the same k tuples are always returned by the 'similarity-first' plan. Starting with the 'Similarity-first' Plan of Figure5.3,

the canonical plan can be generated by applying the commutative property between the range and traditional operators, i.e., Property R4.5, and then, Property k4.6 and Property k4.7, respectively.

5.3 The SIREN Preference Model

The SIREN Preference Model module is developed as a new SIREN functional component. Users' preferences are expressed to SIREN using a user's profile. The Preference Model module collects the users' preferences and generates conditional preference rules (cp-rules), analyzing the semantic information extracted by the preference rules processing and comparing it to the complex data distributions, aiming at finding correlations that can be useful both to speed up query processing and to improve the efficacy of the answer. Together with the rules of the similarity algebra already embedded in the Query Rewriter, these rules are used to rewrite the similarity queries aiming either at simplifying the query execution, enabling to find faster answers, or to better follow the users' expectations, enabling to find better answers.

Following we present the syntax of the SQL extension that we developed to enable expressing user's preference in SIREN. In this section and the next one, we employ the Extended Backus-Naur Form (EBNF) notation, a widely adopted notation for the specification of program languages to present the syntax of the proposed extension.

A new preference model is defined by the `CREATE PREFERENCE MODEL` statement, as follows.

```
<create_preference_model_statement>::=
    CREATE PREFERENCE MODEL <model_name>
           FROM <relation_name>
           AS <preference_list>
           [‘[’<attribute_list>‘]’];
```

The new preference model is called `<model_name>`. It assigns the list of cp-rules defined in `<preference_list>` to the relation `<relation_name>`. The `<model_name>` parameter must be unique. Each rule is declared following the `IF <antecedent> THEN <consequent>` syntax. Multiple rules are assumed to compose a conjunction. Cp-rules can be created with or without antecedents, which are terms in the form `attribute = value` connected by the `AND` keyword. The consequents are always a preference relation between the values of the given attribute. The optional parameter `<attribute_list>`, which is represented between brackets, states that the attributes in the list are not involved in the rule – every involved attributes must have the same value in both tuples compared.

A preference model is dropped by the `DROP PREFERENCE MODEL` statement, whose syntax is as follows.

```
<drop_preference_model_statement>::=
    DROP PREFERENCE MODEL <model_name>;
```

where <model_name> is the preference model to be dropped.

Having created a preference model, the set of cp-rules are validated and the preference model becomes ready to be used. However, a Preference Model is attached to a relation of the database schema, not to a specific user. Therefore, it is necessary that each user explicitly chooses the preference model that his/her wants to attach to his/her personal query environment.

The SET MODIFICATION statement controls what modifications are enabled in each user query environment, as follows.

```
<set_modification_model_statement>::=
    SET MODIFICATION [ADD | REMOVE | UPDATE]
                     [ALL | <model_name>];
```

This statement is used to enable or disable the specified preference model in the user's environment (using the ADD or REMOVE clauses), or to update existing preference models (using the UPDATE clause). When SET MODIFICATION ADD <model_name> is posed, the current set of cp-rules from the <model_name> model is added to the users environment. When SET MODIFICATION UPDATE <model_name> is posed, the preference model associated to the <model_name> model is re-evaluated (for example, due to changing configurations in the profile). When SET MODIFICATION REMOVE <model_name> is issued, the current set of cp-rules from the <model_name> model is removed from the user's environment. The ALL option is used to add, remove or update all preference model from the user in the users environment.

Whenever there are preference models enabled in the users environment, all queries issued by the user are rewritten following those models. Therefore, the SET MODIFICATION command allows the user to control when queries should be modified, and which models must be employed to modify each query. If the user adds a model in his/her environment and other model was already added, the SIREN query rewriter asserts that both models are consistent and uses them to rewrite queries as a conjunction; otherwise, only the last added model is used to rewrite queries.

When there are SET MODIFICATION commands active, each similarity query posed is automatically rewritten taking into account the rules enabled in the user's environment. The rewritten query is enabled adding the ACCORDING TO PREFERENCES clause after the WHERE condition in the SELECT statement, as follows.

```
<according_clause>::=
   ACCORDING TO PREFERENCES ([n,] <model_name_list>);
```

This clause allows performing an additional filtering over tuples returned after the execution of the clauses `FROM` and `WHERE`. The remaining tuples are those satisfying all the users' preferences specified by the model names in `<model_name_list>`. The optional parameter `n`, which is defined in the user's profile, enables selecting the `n` most preferred tuples, respecting the preference hierarchy. Therefore, it is possible to select complex data with varying similarity degrees. The semantic of a kNN query using parameter `n` corresponds to first select the k complex data most similar to the query element, and thereafter, among them, select up to `n` most preferred tuples following the users profile preference hierarchy.

Figure 5.4 summarizes the whole process of preparing and posing queries either using or not using the preference rules, over a database. In the first step (Figure 5.4, Action ①), the user specifies its preferences in the user's profile. The preference rules are generated and stored in the database, as part of the system catalog. Notice that a preference model is attached to the `METRIC <metric_name>` employed to search the database, thus it can be applied to any attribute in any relation sharing the same domain and this the same rules. When searching for similarity with preference model disabled (Figure 5.4, Action ②), the "similarity search" engine uses only the multimedia database to answer the query; otherwise, when the preference model is enabled (Figure 5.4, Action ③), the "similarity search" engine uses the rules together with the multimedia database to evaluate the queries.

5.3.1 Experimental Evaluation

The SIREN prototype and its Preference Model module were implemented in C++, and the experiments were evaluated using an Intel Core 2 Quad 2.83GHz processor with 4GB of main memory, under the Windows XP operational system. The RDBMS used to process the traditional part of the query was the PostgreSQL 8.4.

The experiments were performed using the `LungDS` dataset. It was employed two user profiles, one more general and another more specific, and both are compared with a plain similarity query, with no preferences attached. They were prepared to represent the fact that lung-related diseases are strongly correlated to the seasons, thus a medical doctor can prefer to analyze exams taken at some season over others. Moreover, it is also known that "Consolidation" findings are more common at the driest seasons, thus users' preferences are driven by that knowledge. The more specific preference model

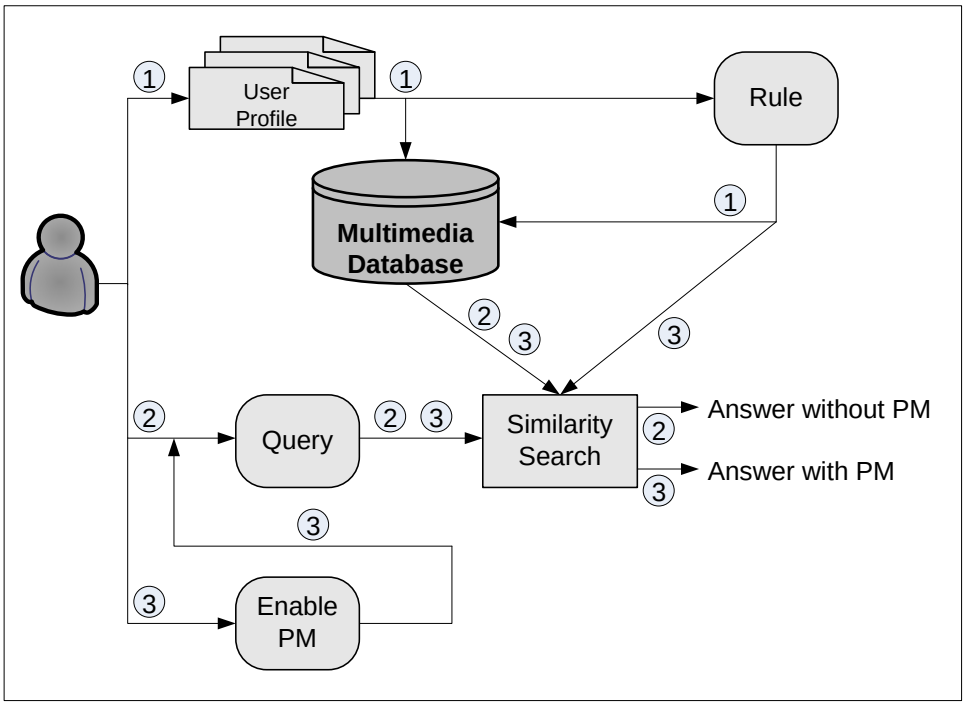


Figure 5.4: Processes to prepare and execute similarity queries considering of preference models.

is the `LungPref`, in which a radiologist's profile is defined regarding lung sickness like bronchiolitis and pneumonia, as follows.

```
CREATE PREFERENCE MODEL LungPref
   FROM LungExams AS
   IF class = 'Consolidation' THEN
      season = 'winter' > season = 'autumn' [id, sex, age, date] AND
   IF class = 'Consolidation' THEN
      season = 'autumn' > season = 'spring' [id, sex, age, date] AND
   IF class = 'Consolidation' THEN
      season = 'spring' > season = 'summer' [id, sex, age, date];
```

Then, the `LungPref` preference model shows that the radiologist states when searching for computed tomographies lung exams, if the image classification has the "Consolidation" finding, then the user prefers images of the driest seasons. For example, the condition `season = 'winter' > season = 'autumn'` means that a tuple meeting condition `season = 'winter'` is preferred over those meeting condition `season = 'autumn'`. The clause `[id, sex, age, date]` expresses that those four attributes are irrelevant for the preference evaluation, so they are not involved in the tuple comparisons. The generic preference model is called `DrySeasonsPref`, and in its definition the user just defines that he/she prefers dry and cold weather, saying nothing about computed tomography findings preference, as follows.

```
CREATE PREFERENCE MODEL DrySeasonsPref
   FROM LungExams AS
   season = 'winter' > season = 'autumn' [id, sex, age, date] AND
   season = 'autumn' > season = 'spring' [id, sex, age, date] AND
   season = 'spring' > season = 'summer' [id, sex, age, date];
```

Suppose that a radiologist is searching for pulmonary diseases in an image, and asks for similar previous cases images. Example 5.2 illustrates the corresponding similarity query sent by the radiologist to SIREN in the `LungDS` dataset.

Example 5.2:

Q8: "Among the 10 images most similar to this lung computed tomography having the word 'Consolidation' in its report, the radiologist prefers images obtained in the driest seasons".

```
SELECT id, age, Image
FROM LungExams
WHERE Image NEAR 'C:\PatientExam1.jpg' STOP AFTER 10;
```

However, just creating a preference model with the `CREATE PREFERENCE MODEL` command does not enable SIREN to modify queries. Thus, to enable obtaining a preference-improved answer, the user must first enable the query rewriting using the `SET MODIFICATION` command, as follows.

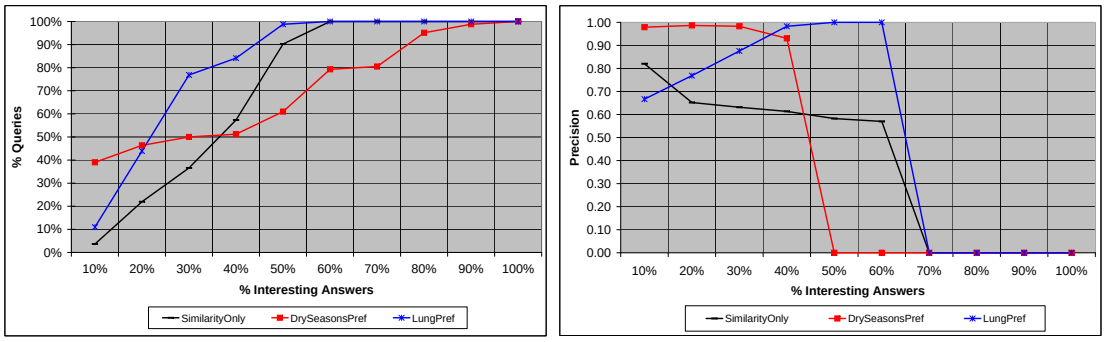
```
SET MODIFICATION ADD LungPref;
```

Thus, when the user environment is enabled, SIREN automatically rewrites the query adding `ACCORDING TO PREFERENCE` clause after the `WHERE` condition. In that case, SIREN answers the query based on the preferences specified in the user's profile, which processes the `LungPref` preference model.

Query Q8 was posed to SIREN asking for $k = 50$ elements for eighty-two distinct query centers, covering 1/3 of the database. The same query set was posed with the preference model disabled, and with the `LungPref` or the `DrySeasonsPref` preference models enabled. Figure 5.5a presents the percentage of interesting answers obtained, considering as interesting the images that have the same class of query center, both in the similarity-only query answer and in the preference-enabled similarity query answers. In the graphic, the percentage of interesting answer is the proportion of the relevant images – Ri – obtained regarding the total number of images of the same class in the database – Tsc (Equation 5.1). The precision is the proportion of relevant images – Ri – obtained regarding the total number of images in the database – Ti (Equation 5.2).

$$Percentage\ Interesting\ Answers = \frac{Ri}{Tsc} \tag{5.1}$$

$$Precision = \frac{Ri}{Ti} \tag{5.2}$$



(a) **(b)**

Figure 5.5: (a) Percentage of correct answer in the similarity-only and preference similarity queries; (b) Precision vs. Percentage (%) Interesting Answers.

Analyzing the results obtained, processing the preferences improves the amount of queries that return more interesting images. In fact, only 37% of the plain similarity queries were able to retrieve 30% of the interesting answers, whereas 50% of the queries meeting the `DrySeasonsPref` preference model and 77% of the queries using the `LungPref` preference model retrieve at least 30% of the interesting answers. On the other hand, Figure 5.5a shows that if the preference model is more generic, it is not able to significantly improve the answers for large amounts of returned images. In fact, 90% of the similarity-only queries and 100% of the `LungPref` preference model return up to 50% of the correct answers but `DrySeasonsPref` obtains only 61% of interesting answers. This is due to the generic preferences find preferred images in classes much distinct from the originally intended class.

Figure 5.5b shows the precision of the answers achieved for varying percentage of interesting answers filtered by the preference evaluation, again considering similarity-only queries and queries with `LungPref` or the `DrySeasonsPref` preference models enabled. When no preference is enabled, the graphic corresponds to the Precision vs. Recall, where the similarity criterion employed achieves precision from 80% to 60% for up to 50% of recall (percentile of interesting answers). It must be remembered that this query searches over the entire database. However, when preference models are enabled, the graphics show that the amount of interesting answers for the same percentile of precision increases considerably. In fact, for the more specific `LungPref` preference model, it achieves 100% of precision for up to 30% of interesting answers. The less specific `DrySeasonsPref` preference model also improves precision for most percentiles of interesting answers, but it reveals an interesting behavior for the range from 10% to 50% of interesting answers: the precision increases for increasingly interesting answers. This is due to the preference-based part of the query answering process working over the intermediate results obtained by the similarity-based part of the query answering process. Thus, as the amount of intermediate

results increases, the preference-based part is able to find better answers. However, this effect does not continue indefinitely. As the similarity process continues to retrieve progressively farther answers, they do not meet the preference anymore, and the benefit obtained by any of them becomes negligible.

Another interesting behavior revealed when we analyze Figure 5.5b is that more selective preference models are better to improve answers for low-cardinality answers. In fact, the figure shows that, in the beginning, the more specific `LungPref` preference model outperforms the `DrySeasonsPref` one. This effect remains whenever the database has sufficient number of interesting answers to be retrieved. This is why the precision of the more specific preference model drops before the more generic does: the amount of images meeting the more specific preference model is smaller than the amount of images that can meet the more generic one.

Those two behavior are highly related to the meaning that the users expect from the answer, thus this is a good indication that the techniques we developed in fact are able to explore the semantic of the user's expectations to improve the similarity answers quality.

5.4 The SIREN Data Mining Model

The SIREN Data Mining Model module, which has the Apriori and the Omega [Ribeiro et al., 2008] algorithms included as a coupled way, was also developed as a new functional component of SIREN. It extracts knowledge from the database, and generates mining rules, correlating semantic information from the textual description to the low-level extracted features. Together with the similarity algebra existing in the Query Rewriting, these mining rules are used to rewrite similarity queries, aiming at improving both the efficiency and the efficacy of similarity query answering.

A data mining model provides the specification of particular data structures, which are stored in the database catalog, generating constraints about the data sets associated with these structures that can be employed during the query answer processing to find physical access paths that speed up the process. A new data mining model is defined by the `CREATE DATA MINING MODEL` statement, as follows.

```
<create_mining_model_statement>::=
    CREATE MINING MODEL <model_name>
           ON <relation_name> (<attrib_name>)
           [WHERE <predicate>]
           [METRIC <metric_name>]
           USING <data_mining_alg_name>[(<parameter_list>)]
```

The `CREATE MINING MODEL` command creates a new data mining model called `<model_name>` over the complex or traditional attribute (`<attrib_name>`) of the relation

(<relation_name>), based on the data mining algorithm (<data_mining_alg_name>). The <model_name> parameter must be unique. The optional clause WHERE <predicate> allows defining that only the elements in the relation that meets the specified <predicate> are employed to evaluate the mining algorithm. The optional clause METRIC specifies which metric (<metric_name>) associated to the complex attribute will be used to create the data mining model. Each data mining algorithm indicated in the <data_mining_alg_name> clause must be individually developed and integrated to the SIREN data mining model module, in which including a new algorithm requires recompiling SIREN. The parameters for the data mining algorithm are optional and depend on the particular algorithm specified. All required parameters are included in the <parameter_list> in the <data_mining_alg_name> clause. Whenever the <parameter_list> of an algorithm is modified, a new data mining model can be created. In this way, it is possible to execute the composition between different models.

The new version of SIREN that we implemented includes the Apriori and the Omega algorithms. The syntax to employ both of them has a number of required and of optional parameters, and all of them are specified together in the same statement. The syntax to specify them in the CREATE MINING MODEL statement is described following.

```
<data_mining_alg_name>::=
    APRIORI (<attribute_class>, <nclass>, <sup>, <conf>
            [, <minint>, <maxmrgint>, <maxkpatt>])
```

The required parameters are: the attribute employed to classify the images (<attribute_class>), the number of classes (<nclass>), and the value of minimum support (sup) and minimum confidence (<conf>) for the association rule mining algorithm. The optional parameters are those employed to tune the Omega algorithm: the minimal interval size (<minint>), the maximum acceptable inconsistence to merge consecutive intervals (<maxmrgint>), and the maximum acceptable inconsistence to retain an attribute (<maxkpatt>).

An existing data mining model can be dropped with the DROP MINING MODEL statement, whose syntax is:

```
<drop_mining_model_statement>::=
    DROP MINING MODEL <model_name>
```

where <model_name> is the name of the data mining model to be dropped.

After a data mining is created, the parameters of the model are also specified and the database is processed by the specified mining algorithm to generate the corresponding rewriting rules, thus the data mining model becomes ready to be used. However, it is necessary that the user explicitly assigns this new model to his/her own query environment. As mentioned in Section 5.3, in order to control what data mining model is

available to change similarity queries, the `SET MODIFICATION` statement must be issued and its syntax is the same presented in that section.

Figure 5.6 summarizes the whole process of preparing and posing queries, either using or not using the mined rules, over a database already loaded with images. In the first step (Figure 5.6, Action ①), the user sends interesting images to the Omega and the Apriori algorithms. The association rules are mined and stored, together with the interesting images, in the database. Notice that the interesting images must be previously stored in the database, possibly together with other images, regardless of them being interesting or not. The interesting images can be stored in the same relation or in any other relation having the same attribute structure. After retrieved, the rules are also stored in the database, as part of the system catalog. Notice that a data mining model is attached to the `METRIC <metric_name>` employed to search the database, thus it can be applied to any attribute in any relation sharing the same image domain and the same rules mined. When searching for similarity with data mining model is disabled (Figure 5.6, Action ②), the "similarity search" engine uses only the image database to answer the query; otherwise, when the data mining model is enabled (Figure 5.6, Action ③), the "similarity search" engine uses the rules together with the image database to evaluate the queries.

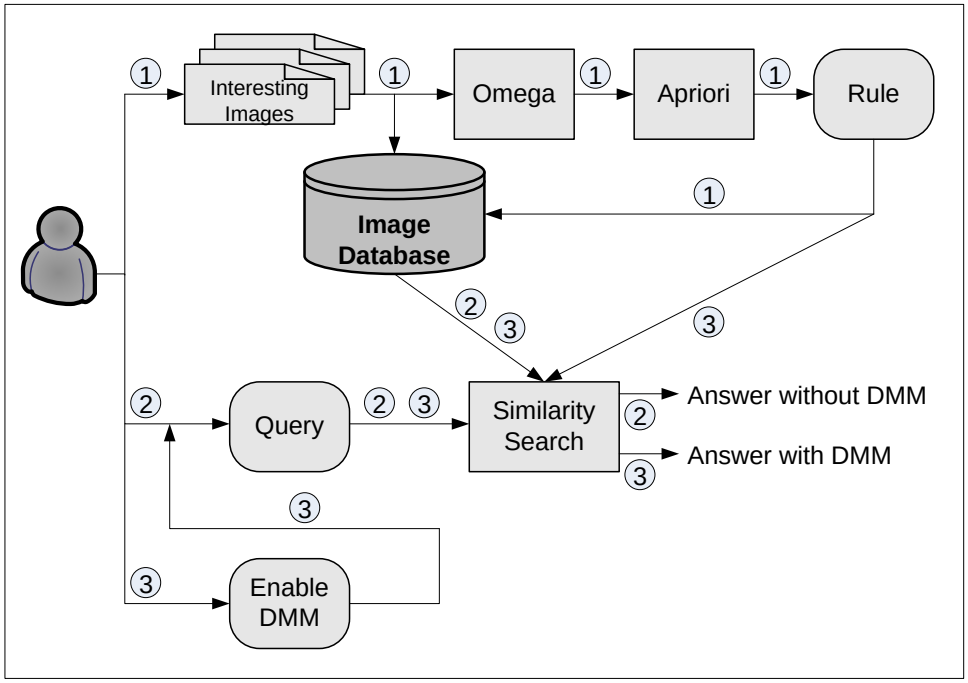


Figure 5.6: Processes to prepare and execute similarity queries considering of data mining models.

5.4.1 Experimental Evaluation

SIREN and its data mining module are implemented in C++, and the experiments were evaluated using an Intel Core 2 Quad 2.83GHz processor with 4GB of main memory,

under the Windows XP operational system. The RDBMS used to process the traditional part of the query was Oracle 9i. The time spent to execute the queries and the quality of the answer are used to evaluate the efficiency and efficacy of the technique.

The experiments were performed using the `WWorldDS` dataset. Suppose that the user is only interested in photos that share a specific characteristic, for which he/she has an initial training subset already marked in the attribute `Training` (that is, `Training = 'True'`). Thus, the user must create a data mining model to evaluate its training subset. Assuming that texture is adequate to discriminate among images that the user is interested in or not, the following command can be issued.

```
CREATE MINING MODEL WondersWorldAssociationRulesModel
    ON WondersWorld (Image) METRIC texture
    WHERE Training = 'True'
    USING APRIORI (Tag, 15, 1, 100, 2, 0.1, 0.42);
```

This command creates a data mining model using the Apriori algorithm to extract association rules from the `Image` attribute of the `WondersWorld` relation. The Apriori algorithm processes only the subset of tuples from the relation where the attribute `Training` has value `'True'`. Also, the Apriori algorithm is executed with a minimum of 1% as the threshold for support and 100% for confidence. The attribute used to classify the `WondersWorld` relation is `Tag`, and this relation has 15 classes. The Omega algorithm is executed using a minimal threshold for interval size of 2, a maximum threshold for merging consecutive intervals of 0.1 and a maximum threshold for keeping an attribute of 0.42. Those threshold were defined following the Omega recommendations [Ribeiro et al., 2008].

When the `CREATE MINING MODEL` command is issued, the 298 images that express the users' expectations (`Training = 'True'`) are retrieved from the `WondersWorld` relation and submitted to the Omega and the Apriori algorithms. These algorithms process the images and generate the rules to be employed by the query rewriter module to process further queries. A total of 1,799 rules were generated. The number of images submitted for training should not be very large, as the Apriori and the Omega algorithms do not scale well. Therefore, the suggestion is selecting from 20 to 200 images of each class as an amount adequate to create the rules. The average time to create this data mining model was 460 milliseconds. It is worth remembering that creating a data mining model is performed only once, and further queries that use it are not affected by this number. Thereafter, the rules generated by a data mining model can be applied over very large image sets, so although using a small amount of training images, this technique is scalable to very large datasets.

Just executing the `CREATE MINING MODEL` command does not enable SIREN to modify the queries. The user must enable query rewriting using the `SET MODIFICATION` command.

Initially considering the data mining model-driven query rewriting disabled, it was submitted several $k\text{NN}_q$ where k is the number of images to be retrieved, which varies from 1 to 20 for each query center. Also, ten queries with distinct query centers were posed for each value of k. Following, it was submitted several R_q, where the range radius of images to be retrieved varies from 0.1 to 0.8 for each query center. Again, ten queries with distinct query centers were posed for each value of the range radius.

Finally, the data mining model-driven query rewriting was enabled, issuing the `SET MODIFICATION` command as follows, and the same sets of $k\text{NN}_q$ and R_q were issued again.

```
SET MODIFICATION ADD WondersWorldAssociationRulesModel;
```

Figure 5.7 summarizes in a data flow how the two commands `CREATE MINING MODEL` and `SET MODIFICATION` respectively prepare and enable the rules to be employed for query rewriting.

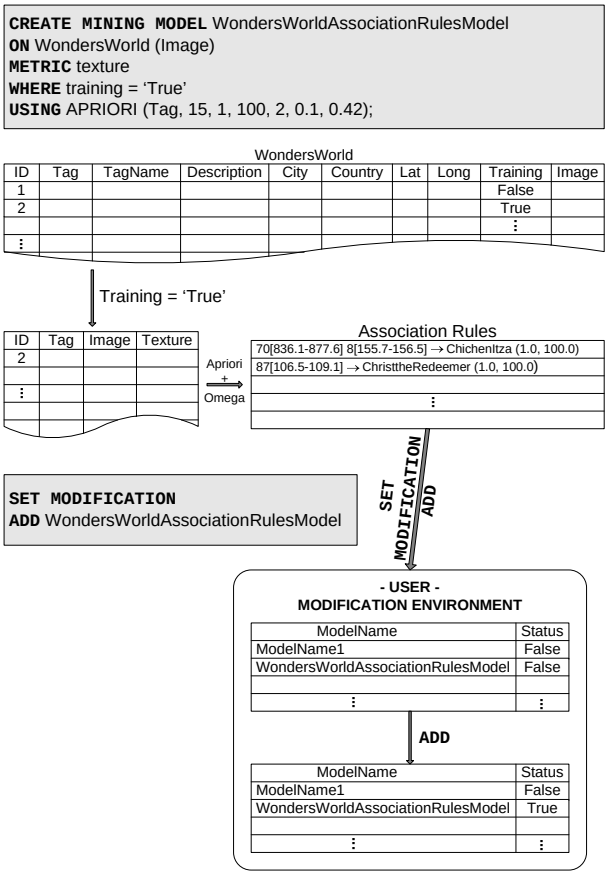


Figure 5.7: Data flow showing how the mining rules are enabled using the `CREATE MINING MODEL` and `SET MODIFICATION` commands.

Table 5.4 shows the average values obtained for the R_q and Table 5.5 presents the average values obtained for the $k\text{NN}_q$. Both tables show the average number of relevant images obtained ('Rel'), the average number of non-relevant images obtained ('Non-Rel') and the average time (in milliseconds) to execute one corresponding query, varying values

of range radius in the R_q, and varying values of k in the $k\text{NN}_q$. The same sets of measurements were performed for both kinds of queries, enabling the query rewriting using the rules mined by the data mining model (the values shown as "With DMM" in both tables) and disabling them (the values shown as "Without DMM" in both tables). "Without DMM" corresponds to the plain execution of the traditional similarity query, that is, without data mining model rule-based query rewriting.

Analyzing the R_q results shown in Table 5.4, it can be seen that for the same range radius with the query rewrite disabled, SIREN returns several images in which the user is not interested in. For example, with a range radius $\xi = 0.2$ it retrieves only 3 interesting images but 22 non-interesting; with a range radius $\xi = 0.8$ all the 20 interesting images are returned, but other 188 non-interesting images are retrieved together. When the query rewrite is enabled, for the same range radius SIREN always returns just the interesting images within the given distance. Table 5.4 also shows that the time required to execute queries for both enabling or disabling the query rewriting does not change significantly. This is due to the fact that the query processing in SIREN is always fast, and the communication between SIREN and Oracle takes the most significant time.

Average Range	Without DMM			With DMM		
	Rel	Non-Rel	Time (ms)	Rel	Non-Rel	Time (ms)
0.1	1	0	47	1	0	42
0.2	3	22	47	3	0	47
0.3	7	54	47	7	0	47
0.4	12	85	52	12	0	52
0.5	15	113	47	15	0	47
0.6	19	141	47	19	0	47
0.7	19	165	47	19	0	47
0.8	20	188	47	20	0	47

Table 5.4: Results from several range values - range queries (average)

Analyzing the $k\text{NN}_q$ results shown in Table 5.5, it can be seen that, again, for the same number of relevant images retrieved with the query rewrite disabled, SIREN also returns several images that are not relevant to the user. It is important to remember that kNN and traditional predicates are not commutative. Therefore, if the user asks for a kNN query with the option for query rewriting disabled, it will be returned k images, although possibly not every image will be interesting, therefore the number of interesting images returned is at most k, but it is often less than k relevant images. To perform this experiment with the query rewritten disabled, we repeated the experiment with higher values of k, until the desired number of interesting images was obtained. Column "Non-Rel" from the "Without DMM" experiment reports the average number of k required to obtain the corresponding number of interesting images.

Table 5.5 shows, for example, that to retrieve 2 interesting images, an average of 12 images including non-interesting ones should be asked for; that is, the kNN must be issued asking for, at least, $k = 12$. To retrieve all the 20 interesting images, it should be asked for an average of 215 images. When the query rewrite is enabled, only the required number of images must be effectively asked, that is, k can be set exactly to the desired value. As it occurs regarding range queries, the time to process both queries is equivalent.

Average kNN	Without DMM			With DMM		
	Rel	Non-Rel	Time (ms)	Rel	Non-Rel	Time (ms)
1	1	3	47	1	0	42
2	2	12	42	2	0	42
3	3	22	47	3	0	47
4	4	27	37	4	0	47
5	5	30	42	5	0	47
6	6	33	47	6	0	47
7	7	38	47	7	0	47
8	8	46	47	8	0	47
9	9	58	41	9	0	41
10	10	67	47	10	0	42
11	11	71	42	11	0	47
12	12	75	42	12	0	47
13	13	85	47	13	0	47
14	14	98	47	14	0	42
15	15	108	36	15	0	47
16	16	118	42	16	0	47
17	17	123	47	17	0	47
18	18	127	42	18	0	47
19	19	141	42	19	0	47
20	20	215	42	20	0	47

Table 5.5: Results from several k values - kNN queries (average)

Figure 5.8 shows an example of ten images that have the `Training` attribute set to `'True'`, that is, some of the images submitted to the association rule mining algorithms Apriori and Omega. Also, this figure presents examples of the results obtained when a $k\text{NN}_q$ is executed both with data mining model disabled and enabled, using $k = 10$ over the full database regardless of the `Training` attribute setting.

5.5 Final Comments

This chapter presented the application of the techniques developed in this doctorate program into the Similarity Retrieval Engine - SIREN. This case study showed that it is feasible to integrate similarity-based operators with traditional ones, without causing too much impact in neither the optimization process nor the query language structure,

whereas nonetheless providing a powerful and flexible basis to support similarity queries. Moreover, semantic restrictions, which are not employed in the optimization process of the scalar data because of their associated high processing costs, can be successfully employed for the query optimization of complex data, improving both the efficiency and the efficacy of similarity queries.

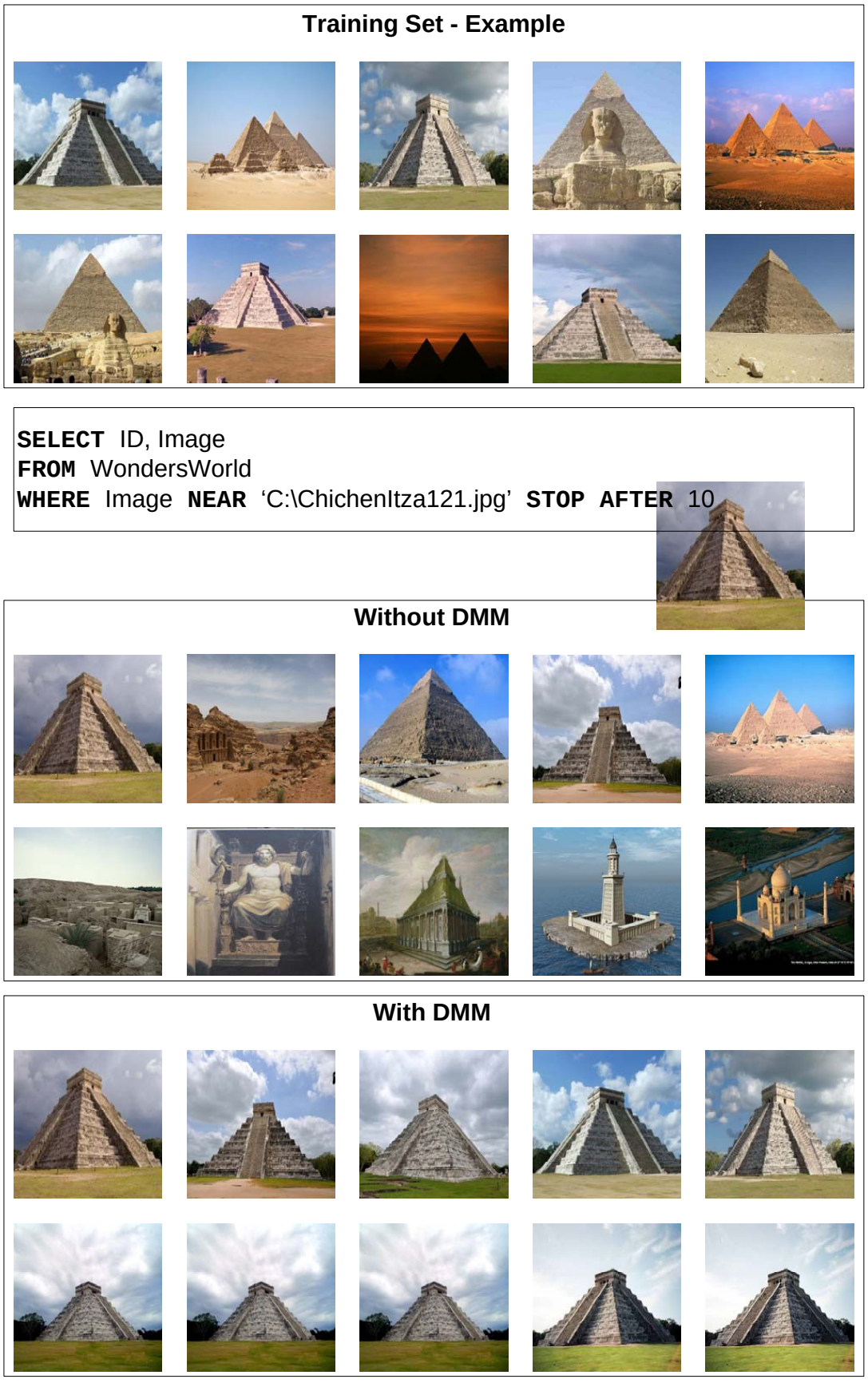


Figure 5.8: Results of 10NN over query enabling and disabling the use of a data mining model. The training set example was obtained from the `WondersWorld` relation with the attribute `Training = 'True'` and used to generate the rules.

Chapter

6

Conclusion

6.1 Final Considerations

With the advent of applications that use complex data such as multimedia, spatial, time series and genetic sequence, relational database management systems (RDBMSs) are being increasingly requested to store and recover these data types. However, for RDBMSs to efficiently retrieve complex data, it is of paramount importance that their query optimizer provide support for queries based on similarity predicates, seamlessly integrating them to the identity- and TOR-based predicates that the traditional query optimizers were developed to handle. This work targets improving that support.

6.2 Main Contributions

This thesis contributes for the similarity query optimization process, exploiting algebraic properties and semantic restrictions that can be successfully employed by the RDBMS optimizer module to improve the efficiency and quality of the query answering execution.

Our first contribution was to define a model to include the similarity-based operators into the relational model, in a way that both complex and simple attributes can be queried by similarity, identity and relational comparisons. As our second contribution, the canonical plan generation algorithm that translates SQL queries into algebraic expressions was extended to also accept the similarity-based constructions, precisely defining the canonical plan that is the input to the query optimizer.

Thereafter, the third contribution was to establish a complete set of algebraic rules to handle queries that mix similarity and non-similarity based conditions. To this intent, we defined two new algebraic operators based on the k-nearest neighbors and on the

similarity range queries, that are the similarity-based counterpart of the identity- and TOR-based traditional selection operator. The equivalence properties of expressions that employ the new operators were identified, taking into account the properties relating only similarity-based operators or relating any composition of similarity, identity- and TOR-based operators, including the selection, set-theoretical and cross product operators. The identified equivalence properties revealed that there are too few rules that can be employed for the query optimization process, thus severely restricting the opportunities to achieve adequate optimized query access plans.

Therefore, our fourth contribution was to identify other, non-equivalence-based properties, that could achieve that goal. The result was the inclusion-based properties. They allow generating an alternative expression that, although not equivalent to the intended one, is assured to include all of its elements. Thus, if the alternative expression can be evaluated much faster than the intended one, it can be worth to use it followed by a final filtering processing that drops its false positives, leading to an overall faster way to obtain the correct answer. The relational algebra together with the equivalence and inclusion-based properties of the algebraic operators defines what we called the "Similarity Algebra", which was published in Ferreira et al. [2009] and Ferreira et al. [2011].

Similarity is a concept that is highly related to the human perception of how to compare things. Therefore, the human understanding of things stored as data elements in the database, often referred as the "semantic information" related to the data, is an important asset to be used to improve the quality of similarity queries. We assumed that semantic information can be expressed as predicates, restricting the range of the values that attributes of the stored elements can assume. As a consequence, identification of users' interests can be roughly expressed as restricting the ranges of selected attributes from the elements in the query results, which ultimately mean that taking the user's interest into account to answer queries corresponds to create techniques to automatically include the corresponding predicates as part of the queries issued by the user. Those predicates involves similarity, identity- and TOR-based ones, thus our similarity algebra becomes a powerful tool to aid in supporting semantic restrictions on similarity-based retrieval to improve query quality.

Besides the user's knowledge about the data and the related applications, there is also patterns in the stored data that can be useful to be taken into account to speed up query processing. Thus, pre-processing the data with data mining techniques which retrieve useful patterns describing the data distribution that can be represented as predicates of any kind (similarity, identity and TOR-based). Those predicates are thereafter automatically included in the query plan, and they can also help speeding up query processing, specially the queries that also use similarity based predicates, which are clearly more time-consuming and thus more worth to handle by those techniques.

Automatically including predicates to act as filters based both on users' preferences and on data distribution can help improving both the query quality and the query efficiency.

To evaluate our techniques, we started with an existing prototype of a RDBMS extended to handle similarity query — the SIREN engine — modifying it to include our proposed concepts and assumptions. The main modifications included: defining new language clauses to express the new concepts, correspondingly extending the query interpreter; extending the query optimizer to handle similarity queries based on syntax and on semantic optimization; and implementing the required data retrieval algorithms in the relational engine.

Our fifth contribution was to extend the query optimizer to handle the syntax-based optimization extensions. It embodies the complete similarity algebra to rewrite similarity predicates either alone or mixed with traditional ones. The similarity algebra enables the query rewriter to generate multiple expressions of the same query, generating equivalent or "inclusion-based plus pos-filtering" physical access plans. After generating several alternative expressions, the query optimizer is able to estimate their costs and choose the one with the lowest computational cost. The result is a new version of the Similarity Retrieval Engine (SIREN$_{op}$) able to handle queries composed of any combination of predicates that express identity, TOR or similarity queries, and using the similarity algebra to optimize the execution.

Our sixth contribution was to explore the semantic of users' interests associated to similarity queries. To this intent, we used a version of the Postgres DBMS extended to handle a preference model based on conditional preference rules (cp-rules), and extended SIREN$_{op}$ with a Preference Model module. The user can access this feature expressing his/her interests using an SQL extended to handle cp-rules, whose interpreter is available in the cp-Postgres extension, and that was further extended to allow tailoring cp-rules to similarity search, whose interpreter was included in SIREN$_{op}$. This semantic-based extension of SIREN$_{op}$ applies the user's interest expressed as cp-rules to improve the query quality (tailoring the execution to better meet the users' expectations). The technique that allows employing user's preferences to optimize similarity queries was published in Ferreira et al. [2010a]. This work was developed in cooperation with the *Universidade Federal de Uberlândia* (UFU).

Finally, our seventh contribution was to explore data distribution information about hidden patterns in the data, using data mining techniques. To this intent, we extended SIREN$_{op}$ with a Data Mining Module, which includes the Apriori and the Omega [Ribeiro et al., 2008] algorithms working in a cooperative way. The user can access this feature expressing his/her interests using an SQL extended to handle "Data Mining Models", a concept we developed to allow expressing data mining tasks that can be executed over the data in a way similar to the index creation. The data gathered by a mining task is stored in the database, and can be employed to speed up further queries over the mined data.

The data mining extension applies the rules that describes the mined rules as restrictions over the data to evaluate the queries to improve the query execution efficiency (using the restrictions as screening predicates to prune part of the data that is known to have no interesting data). The technique that allows using association rules to optimize similarity queries was published in Ferreira et al. [2010b].

6.3 Future Works

Including similarity queries as a new kind of predicate that can be seamlessly integrated to all the existing resources the current Relational Database Management Systems is a powerful tool that opens several possibilities of both theoretical, core database research and applied research, besides the many application of its results to specific application areas. Our work provides a solid theoretical foundation for this support, as it adds new functionality to applications developed over the relational model, easing the representation of similarity queries without in fact changing the model. Therefore, our results can be applied to extend any tool that had been developed using the relational model and its derivatives. However, although fundamental for supporting similarity queries with a solid foundation over relational DBMS, that support requires further development to complete its development, both in the theoretical and in the applied research point of view. We highlight here some of them.

6.3.1 Future Applied Research

- **Extension of the Similarity Algebra to handle binary similarity operators:** We provided a complete set of properties to support the similarity-based version of the algebraic "select" operator σ. However, predicates are employed also in the traditional algebraic "join" operator of the relational algebra. The literature has showing that there are at least three similarity join operators (similarity range join, k-nearest neighbors join and k-closest neighbors join), whose properties was never studied. Although a basic understanding of those join operators can be derived throughout the properties we developed for similarity selection combined to the cross product operator, specific properties for each of the three similarity joins are yet to be studied.

- **Extension of the Similarity Algebra to handle similarity-based set-theoretical operators:** A set is usually defined as a collection where each element occurs just once. This concept embodies the idea that the representation of two elements can be compared to determine its identity. If identity is exchanged by similarity, a new concept should be created: the one that defines a collection where each element is not similar to any other "given a similarity threshold". This

concept is very useful for many applications and also enables the development of theoretical concepts that can aid in several areas. For example, it can be employed to drop photos too much similar in a photo database, helping to reconcile data from the same subject obtained from distinct sources, identify security issues in computer systems, and so on.

- **Extension of the Similarity Algebra to handle similarity-based aggregate and grouping operators:** The relational grouping is based on identifying groups of tuples compared by identity of some of their attributes. Again, exchanging identity by similarity enable the analysis of data regarding similarity. However, a precise description of the similarity-based aggregate and grouping operators was only barely done in the literature, and identifying their properties is yet to be studied.

- **Extension of the Similarity Algebra to handle with diversity operators:** Several applications are finding that performing similarity retrieval over very large databases retrieves too much elements that are too much similar to each other. Therefore, a new requirement is emerging, mainly in web-based applications: retrieving elements similar to the query center but diverse among themselves. The new "similarity with diversity" operators that are being proposed follow the same structure of the k-nearest neighbors and range query operators, but follow distinct properties. Given the high relevance and interest of those operators, it is worth to further study their properties and integrate them with our proposed similarity algebra.

6.3.2 Future Theoretical Research

- **Development of index structure to speed up the execution of similarity combined to traditional operations in DBMS:** Traditionally, indexes such as B-tree and hash structures are employed to retrieve data based on identity and on TOR comparisons, and more than one attribute can be indexed in the same structure. Similarity comparisons can also employ metric structures, but most of those structures can handle just one attribute at a time, and those that handle more than one can compare them only regarding the same similarity predicate. Thus, developing an index structure able to handle similarity **and** identity- or TOR-based comparisons is a very useful tool to improve the efficiency of RDBMS executing similarity queries.

- **Development of selectivity and cost estimation techniques for similarity predicates, considering local data parameters:** The optimization process is highly sensitive to the precision of the statistics employed to estimate the selectivity if the predicates and the execution costs of the physical access methods. Traditional

data processing often rely on histograms describing the data distribution over the attribute ranges, either based on the ordering of the attribute values or on the spatial distribution of spatial data. However, metric data have very few works studying this issue, and often only global metrics are collected. Thus, targeting the development of a description model of the data distributed in a metric space is an important endeavor. Using data mining tools to perform the retrieval of the distribution description, as we did in this work, is a first approach, but it need to be further studied so the results can be more broadly used to generic metric spaces.

- **Development of algorithms to handle similarity-based set-theoretical operators:** After a precise definition of what should be a "similarity-based set", index structures and algorithms to execute the similarity-based operations that are guaranteed to not have similar elements in a given threshold can surely improve the query efficiency and efficacy. Given that similarity-based sets are specially useful to handle very large databases, the development of such algorithms spot as specially interesting for similarity-based queries.

Bibliography

Adali, S., Bonatti, P. A., Sapino, M. L., and Subrahmanian, V. S. (1998). A multi-similarity algebra. In *ACM SIGMOD International Conference on Management of Data*, volume 1, pages 402–413, Seattle, Washington, USA. ACM Press.

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

Adali, S., Sapino, M. L., and Marshall, B. (2007). A rank algebra to support multimedia mining applications. In *International Workshop on Multimedia Data Mining (MDM)*, pages 1–9, San Jose, CA, USA. ACM.

Agrawal, R., Imielinski, T., and Swami, A. N. (1993). Mining association rules between sets of items in large databases. In Buneman, P. and Jajodia, S., editors, *ACM SIGMOD International Conference on Management of Data*, volume 1, pages 207–216, Washington, D.C. ACM Press.

Agrawal, R. and Srikant, R. (1994). Fast algorithms for mining association rules. In *International Conference on Very Large Databases (VLDB)*, Santiago de Chile, Chile.

Aronovich, L. and Spiegler, I. (2007). CM-tree: A dynamic clustered index for similarity search in metric databases. *Data & Knowledge Engineering (DKE)*, 63(3):919–946.

Atnafu, S., Brunie, L., and Kosch, H. (2001). Similarity-based algebra for multimedia database systems. In *Australasian Conference on Database Technologies (ACD)*, pages 115–122, Queensland, Australia. IEEE Computer Society.

Atnafu, S., Chbeir, R., Coquil, D., and Brunie, L. (2004). Integrating similarity-based queries in image DBMSs. In *ACM Symposium on Applied Computing (SAC)*, pages 735–739, Nicosia, Cyprus. ACM Press.

Baeza-Yates, R. A., Cunto, W., Manber, U., and Wu, S. (1994). Proximity matching using fixed-queries trees. In *Combinatorial Pattern Matching (CPM)*, volume 807 of *Lecture Notes in Computer Science*, pages 198–212, Asilomar, CA. Springer Verlag.

Baioco, G. B., Traina, A. J. M., and Traina Jr., C. (2007). MAMCost: Global and local estimates leading to robust cost estimation of similarity queries. In *International Conference on Scientific and Statistical Database Management (SSDBM)*, page 6, Banff, Canada. ACM Press.

Barioni, M. C. N., Razente, H. L., Traina, A. J. M., and Traina Jr., C. (2006). SIREN: A similarity retrieval engine for complex data. In Dayal, U., Whang, K.-Y., Lomet, D. B., Alonso, G., Lohman, G. M., Kersten, M. L., Cha, S. K., and Kim, Y.-K., editors, *Demo session of the International Conference on Very Large Data Bases (VLDB)*, pages 1155–1158, Seoul, South Korea. ACM Press.

Barioni, M. C. N., Razente, H. L., Traina, A. J. M., and Traina Jr., C. (2009). Seamlessly integrating similarity queries in SQL. *Software: Practice and Experience (SPE)*, 39(4):355–384.

Beecks, C., Assent, I., and Seidl, T. (2011). Content-based multimedia retrieval in the presence of unknown user preferences. In *International Conference on Advances in Multimedia Modeling (MMM)*, volume 6523 of *Lecture Notes in Computer Science*, pages 140–150, Taipei, Taiwan. Spring-Verlag.

Belohlavek, R., Opichal, S., and Vychodil, V. (2007). Relational algebra for ranked tables with similatiries properties and implementation. In *International Symposium on Intelligent Data Analysis (IDA)*, volume 4723 of *Lecture Notes in Computer Sciences*, pages 140–151, Ljubljana, Slovenia. Springer.

Belohlavek, R., Urbanova, L., and Vychodil, V. (2011). Similarity of query results in similarity-based databases. In Yao, J., Ramanna, S., Wang, G., and Suraj, Z., editors, *International Conference on Rough Sets and Knowledge Technology (RSKT)*, volume 6954 of *Lecture Notes in Computer Science*, pages 258–267, Banff, Canada. Springer.

Belohlavek, R. and Vychodil, V. (2009). Logical foundations for similarity-based databases. In Chen, L., Liu, C., Liu, Q., and Deng, K., editors, *International Conference on Database Systems for Advanced Applications - Workshops: MCIS & WDPP (DASFAA Workshops)*, volume 5667 of *LNCS*, pages 137–151, Brisbane, Australia. Springer.

Belohlavek, R. and Vychodil, V. (2010). Query systems in similarity-based databases - logical foundations, expressive power, and completeness. In *ACM Symposium on Applied Computing (SAC)*, pages 1648–1655, Sierre, Switzerland. ACM.

Belussi, A. and Faloutsos, C. (1995). Estimating the selectivity of spatial queries using the correlation fractal dimension. In *International Conference on Very Large Databases (VLDB)*, pages 299–310, Zurich, Switzerland. Morgan Kaufmann.

Böhm, C. (2000). A cost model for query processing in high dimensional data spaces. *ACM Transactions on Database Systems (TODS)*, 25(2):129 – 178.

Bolettieri, P., Esuli, A., Falchi, F., Lucchese, C., Perego, R., Piccioli, T., and Rabitti, F. (2009). CoPhIR: a test collection for content-based image retrieval. *Computing Research Repository (CoRR)*, abs/0905.4627v2:1–15.

Bozkaya, T. and Özsoyoglu, Z. M. (1997). Distance-based indexing for high-dimensional metric spaces. In *ACM SIGMOD International Conference on Management of Data*, pages 357–368, Tucson, AZ. ACM Press.

Bozkaya, T. and Özsoyoglu, Z. M. (1999). Indexing large metric spaces for similarity search queries. *ACM Transactions on Database Systems (TODS)*, 24(3):361–404.

Braunmüller, B., Ester, M., Kriegel, H.-P., and Sander, J. (2000). Efficiently supporting multiple similarity queries for mining in metric databases. In *IEEE International Conference on Data Engineering (ICDE)*, pages 256–267, San Diego, CA. IEEE Computer Society.

Brin, S. (1995). Near neighbor search in large metric spaces. In *International Conference on Very Large Databases (VLDB)*, pages 574–584, Zurich, Switzerland. Morgan Kaufmann.

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

Carélo, C. C. M., Pola, I. R. V., Ciferri, R. R., Traina, A. J. M., Traina Jr., C., and de Aguiar Ciferri, C. D. (2009). The onion-tree: quick indexing of complex data in the main memory. In *East-European Conference on Advances in Databases and Information Systems (ADBIS)*, pages 235–252.

Carélo, C. C. M., Pola, I. R. V., Ciferri, R. R., Traina, A. J. M., Traina Jr., C., and de Aguiar Ciferri, C. D. (2011). Slicing the metric space to provide quick indexing of complex data in the main memory. *Information Systems(IS)*, 36(1):79–98.

Chakrabarti, K., Ortega-Binderberger, M., Mehrotra, S., and Porkaew, K. (2004). Evaluating refined queries in top-k retrieval systems. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 16(1):256–270.

Chalhoub, G., Chbeir, R., and Yétongnon, K. (2006). Flexible shape-based query rewriting. In *International Conference on Flexible Query Answering Systems (FQAS)*, volume 4027 of *Lecture Notes in Computer Science*, pages 427–440, Milan, Italy. Springer Berlin / Heidelberg.

Chang, K. C.-C. and Hwang, S.-w. (2002). Minimal probing: supporting expensive predicates for top-k queries. In *ACM SIGMOD International Conference on Management of Data*, pages 346–357, Madison, Wisconsin. ACM Press.

Chaudhuri, S. (1998). Data mining and database systems: Where is the intersection? *Data Engineering Bulletin*, 21(1):4–8.

Chaudhuri, S., Gravano, L., and Marian, M. (2004). Optimizing top-k selection queries over multimedia repositories. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 16(8):992–1009.

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

Ciaccia, P., Montesi, D., Penzo, W., and Trombetta, A. (2000). Imprecision and user preferences in multimedia queries: A generic algebraic approach. In *International Symposium on Foundations of Information and Knowledge Systems (FolKS)*, volume 1762 of *Lecture Notes in Computer Science*, pages 50–71, Burg (Spreewald), Germany. Springer-Verlag.

Ciaccia, P., Montesi, D., Penzo, W., and Trombetta, A. (2001). Fuzzy query language for multimedia data. In *Design and Management of Multimedia Information Systems: Opportunities and Challenges*, pages 201–213. Idea Group Publishing (IGI Publishing), Hershey, PA, USA.

Ciaccia, P., Patella, M., and Zezula, P. (1997). M-tree: An efficient access method for similarity search in metric spaces. In *International Conference on Very Large Databases (VLDB)*, pages 426–435, Athens, Greece. Morgan Kaufmann.

Codd, E. F. (1970). A relational model of data for large shared data banks. *Communications of the ACM (CACM)*, 13(6):377–387.

Codd, E. F. (1972). Relational completeness of data base sublanguages. *Database Systems*, 987(17041):65–98.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2001). *Introduction to Algorithms*. The MIT Press, 2nd edition.

Date, C. J. (2009). *SQL and Relational Theory - How to Write Accurate SQL Code*. O'Reilly Media.

de Amo, S. and Ribeiro, M. R. (2009). CPref-SQL: A query language supporting conditional preferences. In *ACM Symposium on Applied Computing (SAC)*, pages 1573–1577, Honolulu, Hawaii, USA. ACM Press.

Döller, M. and Kosch, H. (2005). Approximating the selectivity of multimedia range queries. In *IEEE International Conference on Multimedia and Expo (ICME)*, pages 382–385, Amsterdam, The Netherlands. IEEE Computer Society.

Fagin, R. and Wimmers, E. L. (1997). Incorporating user preferences in multimedia queries. In *International Conference Database Theory (ICDT)*, volume 1186 of *Lecture Notes in Computer Science*, pages 247–261, Delphi, Greece. Springer.

Felipe, J. C., Traina, A. J. M., and Traina Jr., C. (2003). Retrieval by content of medical images using texture for tissue identification. In *16th IEEE Symposium on Computer-based Medical Systems*, pages 175–180, New York. IEEE Computer Society.

Ferreira, M. R. P., Ponciano-Silva, M., Traina, A. J. M., Traina Jr., C., de Amo, S., Pereira, F. S. F., and Chbeir, R. (2010a). Integrating user preference to similarity queries over medical images datasets. In Dillon, T., Rubin, D., Gallagher, W., Sidhu, A., and Tsymbal, A., editors, *IEEE International Symposium on Computer-Based Medical Systems (CBMS)*, pages 486–491, Perth, Australia. IEEE Computer Society.

Ferreira, M. R. P., Ribeiro, M. X., Traina, A. J. M., Chbeir, R., and Traina Jr., C. (2010b). Adding knowledge extracted by association rules into similarity queries. *Journal of Information and Data Management (JIDM)*, 1(3):391– 406.

Ferreira, M. R. P., Santos, L. F. D., Traina, A. J. M., Dias, I., Chbeir, R., and Traina Jr., C. (2011). Algebraic properties to optimize kNN queries. *Journal of Information and Data Management (JIDM)*, 2(3):385–400.

Ferreira, M. R. P., Traina, A. J. M., Dias, I., Chbeir, R., and Traina Jr., C. (2009). Identifying algebraic properties to support optimization of unary similarity queries. In Arenas, M. and Bertossi, L., editors, *Alberto Mendelzon International Workshop on Foundations of Data Management (AMW)*, volume 450 of *CEUR Workshop Proceedings*, pages 1–10, Arequipa, Peru. CEUR-WS.

Ferreira, M. R. P., Traina Jr., C., and Traina, A. J. M. (2007). An efficient framework for similarity query optimization. In *ACM International Symposium on Advances in Geographic Information Systems (ACM GIS)*, pages 396–39, Seattle, Washington.

Garcia-Molina, H., Ullman, J. D., and Widom, J. (2000). *Database System Implementation.* Prentice Hall, New Jersey.

Graefe, G. (2011). Modern B-tree techniques. *Foundations and Trends in Databases (FTDB)*, 3(4):203–402.

Gunopulos, D., Kollios, G., Tsotras, V. J., and Domeniconi, C. (2005). Selectivity estimators for multidimensional range queries over real attributes. *The International Journal on Very Large Databases*, 14(2):137 – 154.

Hadjieleftheriou, M., Yu, X., Koudas, N., and Srivastava, D. (2008). Hashed samples: Selectivity estimators for set similarity selection queries. *Proceedings of the VLDB Endowment (PVLDB)*, 1(1):201–212.

Han, J. and Kamber, M. (2006). *Data Mining: Concepts and Techniques*. The Morgan Kaufmann Series in Data Management Systems. Morgan Kaufmann Publishers, San Francisco, CA, USA, second edition edition.

Haralick, R. M., Shanmugam, K., and Dinstein, I. (1973). Textural features for image classification. *TSMC*, 3:610–621.

Heath, M., Bowyer, K., Kopans, D., Kegelmeyer-Jr., P., Moore, R., Chang, K., and Munishkumaran, S. (1998). Current status of the digital database for screening mammography. In *International Workshop on Digital Mammography (IWDM)*, pages 457–460, Nijmegen, Netherlands. Kluwer Academic Publishers.

Heath, M., Bowyer, K., Kopans, D., Moore, R., and Kegelmeyer-Jr., P. (2000). The digital database for screening mammography. In *International Workshop on Digital Mammography (IWDM)*, pages 212–218, Toronto, Canada. Medical Physics Publishing.

Herstel, T. and Schmitt, I. (2005). Relation-collapse: An optimisation technique for the similarity algebra SA. In *East European Conference on Advances in Databases and Information Systems (ADBIS)*, volume 3631 of *Lecture Notes in Computer Science*, pages 29–42, Tallinn, Estonia. Spring-Verlag.

Hjaltason, G. R. and Samet, H. (2003). Index-driven similarity search in metric spaces. *ACM Transactions on Database Systems (TODS)*, 21(4):517 – 580.

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

Jarke, M. and Koch, J. (1984). Query optimization in database systems. *ACM Computing Surveys (CSUR)*, 16(2):111–152.

Jiang, B., Pei, J., Lin, X., Cheung, D. W., and Han, J. (2008). Mining preferences from superior and inferior examples. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 390 – 398, Las Vegas, Nevada, USA. ACM.

Korn, F., Sidiropoulos, N., Faloutsos, C., Siegel, E. L., and Protopapas, Z. (1996). Fast nearest neighbor search in medical image databases. In *International Conference on Very Large Databases (VLDB)*, pages 215–226, Bombay, India. Morgan Kaufmann.

Kosch, H. (2010). Optimizing similarity-based image joins in a multimedia database. In *International Workshop on Very-Large-Scale Multimedia Corpus, Mining and Retrieval (VLS-MCMR)*, pages 37–42, Firenze, Italy. ACM.

Lee, J.-H., Chun, S.-J., and Park, S. (2003). Selectivity estimation for optimizing similarity query in multimedia databases. In *International Conference on Intelligent Data Engineering and Automated Learning (IDEAL)*, volume 2690 of *Lecture Notes in Computer Science*, pages 638–644, Hong Kong, China. Springer.

Li, C., Chang, K. C.-C., Ilyas, I. F., and Song, S. (2005). RankSQL: query algebra and optimization for relational top-k queries. In *ACM SIGMOD International Conference on Management of Data*, pages 131–142, Baltimore, Maryland. ACM Press.

Lima, E. L. (1993). *Espaços Métricos*. Instituto de Matemática Pura e Aplicada.

Liu, B., Wang, Z., Yang, X., Wang, W., and Shi, B. (2006). A bottom-up distance-based index tree for metric space. In *Rough Sets and Knowledge Technology (RSKT)*, volume 4062 of *Lecture Notes in Computer Science*, pages 442–449, Chongquing, China. Springer.

Liu, L. and Özsu, M. T., editors (2009). *Encyclopedia of Database Systems*. Springer.

Maier, D. (1983). *The Theory of Relational Databases*. Computer Society Press.

Manjunath, B. S., Salembier, P., and Sikora, T., editors (2002). *Introduction to MPEG-7: Multimedia Content Description Interface*. Wiley, 1 edition.

Montesi, D. and Penzo, W. (2000). Taking care of vagueness and user preferences for effective similarity queries on multimedia data. In *Italian Symposium on Advanced Database Systems (SEBD)*, pages 303 – 316, L'Aquila, Italy.

Montesi, D. and Trombetta, A. (1999). Similarity search through fuzzy relational algebra. In *International Workshop on Database & Expert Systems Applications (DEXA Workshop)*, pages 235–239, Florence, Italy. IEEE Computer Society.

Montesi, D., Trombetta, A., and Dearnley, P. A. (2003). A similarity based relational algebra for web and multimedia data. *Information Processing & Management (IPM)*, 39(2):307–322.

Navarro, G. (1999). Searching in metric spaces by spatial approximation. In *String Processing and Information Retrieval Symposium (SPIRE)*, pages 141–148, Cancun, Mexico. IEEE Computer Society.

Navarro, G. (2002). Searching in metric spaces by spatial approximation. *The International Journal on Very Large Databases*, 11(1):28–46.

Navarro, G. and Paredes, R. U. (2011). Fully dynamic metric access methods based on hyperplane partitioning. *Information Systems (IS)*, 36(4):734–747.

Ocsa, A. and Cuadros-Vargas, E. (2007). DBM*-tree: an efficient metric access method. In *ACM Southeast Regional Conference (ACMSE)*, pages 401–406, Winston-Salem, North Carolina. ACM Press.

Paredes, R. U. and Navarro, G. (2009). EGNAT: A fully dynamic metric access method for secondary memory. In *International Workshop on Similarity Search and Applications (SISAP)*, pages 57–64. IEEE.

Paredes, R. U., Navarro, G., Barrientos, R. J., and Marín, M. (2006). An index data structure for searching in metric space databases. In *International Conference on Computational Science (ICCS)*, volume 3991 of *Lecture Notes in Computer Science*, pages 611–617, UK. Springer.

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

Picariello, A. and Sapino, M. L. (2002). A fuzzy algebra for image data bases. In *International Workshop on Multimedia Information Systems (MIS)*, pages 86–95, Tempe, Az, USA. Arizona State University.

Pola, I. R. V., Traina Jr., C., and Traina, A. J. M. (2007). The MM-tree: A memory-based metric tree without overlap between nodes. In *East-European Conference on Advances in Databases and Information Systems (ADBIS)*, volume 4690/2007 of *Lecture Notes in Computer Sciences*, pages 157–171, Varna, Bulgaria. Springer Verlag.

Ramakrishnan, R. and Gehrke, J. (2003). *Database Management Systems*. McGraw-Hill Book Company, New York, NY, 3rd edition.

Ribeiro, M. X., Ferreira, M. R. P., Traina Jr., C., and Traina, A. J. M. (2008). Data pre-processing: a new algorithm for feature selection and data discretization. In *International Conference on Soft Computing as Transdisciplinary Science and Technology (CSTST)*, volume 1, pages 252–257, Cergy-Pontoise, France. ACM Press.

Santos Filho, R. F., Traina, A. J. M., Traina Jr., C., and Faloutsos, C. (2001). Similarity search without tears: The OMNI family of all-purpose access methods. In *IEEE International Conference on Data Engineering (ICDE)*, pages 623–630, Heidelberg, Germany. IEEE Computer Society.

Schmitt, I. and Schulz, N. (2004). Similarity relational calculus and its reduction to a similarity algebra. In *International Symposium on Foundations of Information and Knowledge Systems (FolKS)*, volume 2942 of *Lecture Notes in Computer Science*, pages 252–272, Wilhelminenburg Castle, Austria. Spring-Verlag.

Schnaitter, K., Spiegel, J., and Polyzotis, N. (2009). Depth estimation for ranking query optimization. *The International Journal on Very Large Data Bases*, 18(2):521–542.

Selinger, P. G., Astrahan, M. M., Chamberlin, D. D., Lorie, R. A., and Price, T. G. (1979). Access path selection in a relational database management system. In *ACM SIGMOD International Conference on Management of Data*, volume 1, pages 23–34, Boston, Massachusetts. ACM Press.

Silva, Y. N., Aly, A. M., Aref, W. G., and Larson, P.-A. (2010a). SimDB: a similarity-aware database system. In *ACM SIGMOD International Conference on Management of Data*, pages 1243–1246, Indianapolis, Indiana, USA. ACM.

Silva, Y. N., Aref, W. G., and Ali, M. H. (2009). Similarity group-by. In *International Conference on Data Engineering (ICDE 2009)*, ICDE 2009, pages 904–915. IEEE.

Silva, Y. N., Aref, W. G., and Ali, M. H. (2010b). 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., and Tsotras, V. J., editors, *International Conference on Data Engineering (ICDE)*, pages 892–903, Long Beach, CA, USA. IEEE.

Skopal, T., Pokorný, J., and Snásel, V. (2004). PM-tree: Pivoting metric tree for similarity search in multimedia databases. In *East European Conference Advances in Databases and Information Systems (ADBIS - Local Proceedings)*, pages 1–16, Budapest, Hungary.

Stefanidis, K., Koutrika, G., and Pitoura, E. (2011). A survey on representation, composition and application of preferences in database systems. *ACM Transactions on Database Systems (TODS)*, 36(3):19:1–19:45.

Traina, A. J. M. and Traina Jr., C. (2003). Similarity search in multimedia databases. In *Handbook of Video Databases - Design and Applications*, volume 1, pages 711–738. CRC Press.

Traina, A. J. M., Traina Jr., C., Bueno, J. M., Chino, F. J. T., and Marques, P. M. d. A. (2003). Efficient content-based image retrieval through metric histograms. *World Wide Web Journal (WWWJ)*, 6(2):157–185.

Traina Jr., C., Santos Filho, R. F., Traina, A. J. M., Vieira, M. R., and Faloutsos, C. (2007). The OMNI-family of all-purpose access methods: A simple and effective way to make similarity search more efficient. *The International Journal on Very Large Databases (VLDB)*, 16(4):483–505.

Traina Jr., C., Traina, A. J. M., and Faloutsos, C. (2000a). Distance exponent: a new concept for selectivity estimation in metric trees. In *IEEE International Conference on Data Engineering (ICDE)*, page 195, San Diego - CA. IEEE CS Press.

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

Traina Jr., C., Traina, A. J. M., Seeger, B., and Faloutsos, C. (2000b). Slim-trees: High performance metric trees minimizing overlap between nodes. In *International Conference on Extending Database Technology (EDBT)*, volume 1777 of *Lecture Notes in Computer Science*, pages 51–65, Konstanz, Germany. Springer Verlag.

Traina Jr., C., Traina, A. J. M., Vieira, M. R., Arantes, A. S., and Faloutsos, C. (2006). Efficient processing of complex similarity queries in RDBMS through query rewriting. In *ACM International Conference on Information and Knowledge Management (CIKM)*, pages 4–13, Arlington - VA, USA. ACM Press.

Uhlmann, J. K. (1991). Satisfying general proximity/similarity queries with metric trees. *Information Processing Letters (IPL)*, 40(4):175–179.

Vieira, M. R., Traina Jr., C., Chino, F. J. T., and Traina, A. J. M. (2010). DBM-Tree: A dynamic metric access method sensitive to local density data. *Journal of Information and Data Management (JIDM)*, 1(1):111–127.

Vieira, M. R., Traina Jr., C., Traina, A. J. M., and Chino, F. J. T. (2004). DBM-tree: A dynamic metric access method sensitive to local density data. In *Brazilian Symposium on Databases (SBBD)*, volume 1, pages 33–47, Brasília, DF. SBC.

Wang, J. T.-L. and Shasha, D. (1990). Query processing for distance metrics. In *International Conference on Very Large Databases (VLDB)*, pages 602–613, Brisbane, Australia. Morgan Kaufmann.

Wilson, N. (2004). Extending CP-nets with stronger conditional preference statements. In *National Conference on Artificial Intelligence (AAAI)*, pages 735–741, San Jose, CA, USA. AAAI Press.

Yan, F., Hou, W.-C., Jiang, Z., Luo, C., and Zhu, Q. (2007). Selectivity estimation of range queries based on data density approximation via cosine series. *Data & Knowledge Engineering (DKE)*, 63(3):855–878.

Yianilos, P. N. (1993). Data structures and algorithms for nearest neighbor search in general metric spaces. In *Annual ACM/SIGACT-SIAM Symposium on Discrete Algorithms (SODA)*, pages 311–321, Austin, TX.

Yu, C. T. and Meng, W. (2002). *Principles of Database Query Processing for Advanced Applications*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Zezula, P., Amato, G., Dohnal, V., and Batko, M. (2006). *Similarity Search: The Metric Space Approach*. Advances in Database Systems. Springer, NY, USA.

Zisman, A. (1993). *A Árvore-B e uma fronteira de implementação*. Dissertação de mestrado, Universidade de São Paulo.

Appendix

A

The CoPhIR Dataset

The Content-based Photo Image Retrieval[1] (CoPhIR) dataset consists of digital images, visual descriptors and related metadata extracted from around 106 millions images of the Flickr[2] photo-sharing system [Bolettieri et al., 2009]. This collection stores the metadata information, the features extracted using the Moving Picture Experts Group-7 (MPEG-7) [Manjunath et al., 2002] visual descriptors and the links to original images in Flickr into extensible markup language (XML) files (one for each image). The five MPEG-7 visual descriptors used in CoPhIR are: Scalable Color, Color Structure, Color Layout, Edge Histogram and Homogeneous Texture Descriptors. Each entry of the metadata contains textual information of the photo (e.g. id, url, title, description, the spatial location where the photo was taken), the author (e.g. name, location, upload date), the user-provided tags, the comments of other users and all information stored in the exchangeable image file format (EXIF) header of the image file.

For illustration purposes, we use in this thesis, as a running example, a subset of CoPhIR database that mix traditional and complex attributes in the same relation. Thus, the database schema used in this thesis is presented in Example A.1. In this example, the traditional attributes are colored by red, while the complex attribute are colored blue. The primary key attributes are underlined.

Example A.1:

$$\texttt{CoPhIRdb} = \{\underline{\texttt{UserId}}, \underline{\texttt{PhotoId}}, \texttt{Title}, \texttt{Description}, \texttt{Tags}, \texttt{Lat}, \texttt{Long}, \texttt{Country}, \texttt{Image}, \texttt{Coordinate}\}$$

Without loss of generality, we use the Manhattan (L_1) distance function to compute the similarity between elements of the complex attribute `Image`, because this function

[1] CoPhIR website. Available at: `http://cophir.isti.cnr.it/`. Accessed in: July 02, 2012.
[2] Flickr website. Available at: `http://www.flickr.com`. Accessed in: July 02, 2012.

is a metric and makes it intuitively easy to understand the examples presented in this thesis. The features of `Image` complex attribute are extracted using the 'Dominant Color Descriptor' of MPEG-7. In this way, the pair <Dominant Color Descriptor, L_1> defines the metric `DominantColorL1`.

Analogously, for elements of complex attribute `Coordinate`, the Euclidean (L_2) distance function is defined over their domain, so the similarity predicates can be answered over it. The complex attribute `Coordinate` is obtained from the combination of two geographical points, represented in the traditional attributes `Lat` (i.e. the Latitude) and `Long` (i.e. the Longitude). In order to make easier to understand the examples, we assume that there is a function `Coord(CoPhIRdb.Country)`, which returns the `Coordinate` of the country named `Country`.

Therefore, in Chapter 2, we exemplify the concepts using a subset of this database. That is, we use only the traditional attributes, i.e. the red ones in Example A.1. On the other hand, in Chapters 4, we demonstrate the concepts using the whole database (traditional and complex attributes).