Identifying Algebraic Properties to Support Optimization of Unary Similarity Queries*

Mônica Ribeiro Porto Ferreira[1], Agma J. M. Traina[1],
Ires Dias[2], Richard Chbeir[3], and Caetano Traina Junior[1]

[1] Computer Science Dept., ICMC-Univ. of São Paulo, São Carlos-SP, Brazil,
[2] Mathematics Dept., ICMC-Univ. of São Paulo, São Carlos-SP, Brazil,
[3] LE2I Laboratory UMR-CNRS Univ. of Bourgogne, Dijon, France
{monika,agma,iresdias,caetano}@icmc.usp.br
{richard.chbeir}@u-bourgogne.fr

Abstract. Conventional operators for data retrieval are either based on exact matching or on total order relationship among elements. Neither of them is appropriate to manage complex data, such as multimedia data, time series and genetic sequences. In fact, the most meaningful way to compare complex data is by similarity. However, the Relational Algebra, employed in the Relational Database Management Systems (RDBMS), cannot express similarity criteria. In order to address this issue, we provide here an extension of the Relational Algebra, aimed at representing similarity queries in algebraic expressions. This paper identifies fundamental properties to allow the integration of the unary similarity operators into the Relational Algebra to handle similarity-based operators, either alone or combined with the existing (exact matching and/or relational) operators. We also show how to take advantage of such properties to optimize similarity queries, including these properties into a similarity query optimizer developed for a Similarity Retrieval Engine, which uses an existing RDBMS to answer similarity queries.

Key words: similarity algebra, algebraic properties, query optimization, unary similarity queries

1 Introduction

In 1970, Codd [1] introduced the relational model, which is the foundation for most of the actual commercial DataBase Management Systems (DBMS). It is based on the mathematical relation theory: the database is represented as a set of relations, where each relation is a table with tuples (or rows) and attributes (or columns). The domain of possible values for each attribute is restricted by the data types.

Initially, the relational model supported only traditional data, i.e., numerical and string data types. Elements of these types can be compared using exact matching ($=$ and $\neq$) and relational ($<$, $>$, $\leq$ and $\geq$) operators. Now, with the

* This work has been supported by FAPESP, CNPq and CAPES/Fulbright

advent of multimedia and spatial applications, the Relational DBMS (RDBMS) must be able to support new data types, operators and kinds of queries. Thus, similarity emerges as the natural way to compare elements in complex domains, such as images, audios, videos, genomic sequences, and time series, and consequently handling operations based on similarity (or distance) between data becomes a must. To illustrate this, let us take the following examples:

Q1: In a health-care information system: "*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 texture do not differ more than 10 units from those in the exam*".

Q2: In a health-care information system: "*Given a head tomography exam of a patient showing a pathology, retrieve the 5 exams most similar not presenting pathology, and that texture do not differ more than 5 units from those in the exam*".

Q3: In Geographic Information Systems (GIS): "*Find the 15 districts nearest to 'Arequipa' that are not farther than 15 miles, and where the population having between 21 and 64 year is greater than 65-year-old population and over*".

To solve similarity-based queries, several extensions of relational algebra have been provided in the literature aimed at including the similarity functionality in RDBMS from various perspectives. The first algebra to consider this issue has been the *Multi-Similarity Algebra* (MSA), presented by Adali et al. [2]. It has been designed to integrate different interpretations of similarity values coming from multiple similarity implementation in a common framework. However, it remains at a higher abstraction level and thus does not address the problem of an "operational" algebra usable for modeling, optimizing, and processing queries with similarity-based operations [3]. Therefore, it is not fully consistent to the relational model.

Other works have associated similarity to uncertainty and provided fuzzy logic-based methods to solve this [4, 5]. The problem of those approaches is that they assume that complex data manipulation involves evaluation of their similarity, but this does not mean that these data or the similarity evaluation are 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.

Likewise, other approaches have been based on the notion of ranking, i.e., ordering among tuples or elements [6, 7]. It is true that they are consistent to the relational model and can be applied to similarity queries considering the distance functions as the ranking criterion, but they depend on ranking criteria that are independent from queries, whereas the ranking criterion of a similarity query varies with the query.

None of these previous works has addressed optimizations based on query rewriting for the similarity-based select operators in complex expressions. Traina et al. [8] proposed an extension of relational algebra considering complex similarity queries with two or more similarity predicates combined with Boolean operators. However, they have only treated queries with the same query element

(unique center), which is very restrictive and does not cover all cases occurring in a RDBMS.

In this paper, we present the fundamental properties of a *Similarity Algebra* aiming at integrating both unary similarity operators with the relational algebra, which allows optimizing similarity queries in relational DBMS. The properties allow handling queries including any number of query centers, and suitable to support both similarity-based and traditional operators in the same query.

The remainder of this paper is structured as follows. Section 2 presents the Similarity Algebra. Section 3 shows experimental results conducted to evaluate the relevance of our approach. Finally, Section 4 concludes this paper and draws our future steps.

2 Similarity Algebra

2.1 Preliminaries

In order to execute similarity queries in relational DBMS, it is necessary to provide a measurement of how to quantify similarity between two elements. Usually, it is done by defining a distance function d, which is the basis to create a metric space $M = \langle \mathbb{S}, d \rangle$, where $\mathbb{S}$ denotes the universe of valid elements (domain) and d is a function $d : \mathbb{S} \times \mathbb{S} \rightarrow \mathbb{R}^+$ that expresses a "distance" between elements of $\mathbb{S}$. The distance function d must satisfy the following properties: (*i*) symmetry: $d(s_1, s_2) = d(s_2, s_1)$; (*ii*) non-negativity: $0 < d(s_1, s_2) < \infty$, if $s_1 \neq s_2$ and $d(s_1, s_1) = 0$; and (*iii*) triangular inequality: $d(s_1, s_2) \leq d(s_1, s_3) + d(s_3, s_2), \forall s_1, s_2, s_3 \in \mathbb{S}$.

An attribute is comparable by similarity only if it is associated to a similarity measure d. Although distance functions can theoretically be assigned to any attribute, they are of utter importance when applied to complex attributes. Therefore, without loss of generality, we call complex attributes[1] and, correspondingly, its domains, those associated to distance functions, and the others we call simple attributes.

Relations that have complex attributes should follow the same properties and definitions of traditional relations. In this paper, we employ the following notation to express relations. 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 an complex attribute in a domain $\mathbb{S}_j$ in a metric space that allows comparisons using complex operators; and T be an relation with any number of both simple and complex attributes. That is, let $\mathbb{T} = \{\mathbb{A}_1, \ldots, \mathbb{A}_m, \mathbb{S}_1, \ldots \mathbb{S}_p\}$ be a relational schema, a relation $T \subset \mathbb{T}$ is a set of elements represented as tuples $T = \{A_1, \ldots, A_m, S_1, \ldots S_p\}$, which has for each tuple $t = \langle a_1, \ldots, a_m, s_1, \ldots s_p \rangle$ values a_h $(1 \leq h \leq m)$ obtained in the domain $\mathbb{A}_h$ and values s_j $(1 \leq j \leq p)$ obtained in the domain $\mathbb{S}_j$. Thus, let $t_i(S_j)$ $(1 \leq i \leq n)$ be the value of the S_j complex attribute of the i^{th} tuple in the relation, and correspondingly let $t_i(A_h)$ be the value of the A_h simple attribute. To alleviate the notation of handling several attributes in a

[1] Distinctly from object-oriented models, we employ here the term "complex attribute" to refer to those having a distance function assigned. Examples are images, audios, geographical coordinates, genomaic sequences, etc.

relation, in the remainder of the paper, we will use just S and $\mathbb{S}$ to refer to a complex attribute S_j and its respective domain $\mathbb{S}_j$, and A and $\mathbb{A}$ refer respectively to a simple attribute A_h and its respective domain $\mathbb{A}_h$ whenever the focus of the text is over only one attribute.

2.2 Unary similarity queries operations

Traditional selections follow the format $\sigma_{(A\ \theta\ a)}\,T$, where θ is a comparison operator valid in the domain $\mathbb{A}$ of the attribute A, and 'a' is either a constant taken in the domain of A or the value of another attribute from the same domain of A in the same tuple. Similarity selections follow the same format: $\sigma_{c\,(S\ \theta_c\ s_q)}\,T$, where σ_c represents a similarity selection, θ_c is a similarity operator valid in the domain $\mathbb{S}$ of the attribute S and 's_q' is either a constant taken in the domain of S or the value of another attribute from the same domain of S in the same tuple.

There are two similarity operators commonly employed: range and k-nearest neighbor. As their properties can be different from those of the traditional selection, we initially use the symbols $\hat{\sigma}$ and $\ddot{\sigma}$ to represent range and kNN selections, and $\hat{\theta}$ and $\ddot{\theta}$ to represent range and kNN operators, respectively. They are described as follows.

Definition 1. ***Range query -*** *R_q: Let S be a complex attribute taken in 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 element. The query $\hat{\sigma}_{\left(S\ \hat{\theta}(d,\xi)\ s_q\right)}T$ returns every tuple $t_i \in T$ such that $d\left(t_i\left(S\right), s_q\right) \leq \xi$. That is:*

$$\hat{\sigma}_{\left(S\ \hat{\theta}(d,\xi)\ s_q\right)}\,T = \left\{t_i \in T \mid d\left(t_i\left(S\right), s_q\right) \leq \xi\right\}\ . \tag{1}$$

Definition 2. ***k-Nearest Neighbor query -*** *kNN: Let S be a complex attribute taken in 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 element. The query $\ddot{\sigma}_{\left(S\ \ddot{\theta}(d,k)\ s_q\right)}\,T$ returns the tuples from T whose value of the attribute S is one of the k elements in S nearest to the query element s_q based on the distance function d. That is:*

$$\ddot{\sigma}_{\left(S\ \ddot{\theta}(d,k)\ s_q\right)}\,T{=}T' = \left\{t_i \in T \mid \forall\, t_j \in \left[T - T'\right], T' = t_{i=1,\ldots,k},\right.$$
$$\left. d\left(t_i\left(S\right), s_q\right) \leq d\left(t_j\left(S\right), s_q\right)\right\}\ . \tag{2}$$

2.3 Algebraic properties

The query optimizer of RDBMSs employs algebraic equivalences to rewrite queries into equivalent expressions which are expected to be executed faster. Selections are important operations because they reduce the size of relations. In the subsections following, we identify algebraic properties useful to rewrite expressions of both range operator $\hat{\theta}$ and k-nearest neighbor operator $\ddot{\theta}$. Due to space restriction, formal proofs of these properties are omitted here (they can be found in Ferreira et al. [9]).

2.3.1 Range Selection - $\hat{\sigma}$.

Properties 1 and 2 apply conjunctive and disjunctive conditions involving only $\hat{\sigma}$ operations, respectively.

Property 1. *Conjunctions of $\hat{\theta}$ operators can be rewritten into a cascade of individual $\hat{\sigma}$ operations or a sequence of intersection operations, i.e.,*

$$\hat{\sigma}_{\left(S_1\ \hat{\theta}(d_1,\xi_1)\ s_{q1}\right) \wedge \left(S_2\ \hat{\theta}(d_2,\xi_2)\ s_{q2}\right)}\ T\ =\ \hat{\sigma}_{\left(S_1\ \hat{\theta}(d_1,\xi_1)\ s_{q1}\right)}\left(\hat{\sigma}_{\left(S_2\ \hat{\theta}(d_2,\xi_2)\ s_{q2}\right)}\ T\right)$$
$$= \left(\hat{\sigma}_{\left(S_1\ \hat{\theta}(d_1,\xi_1)\ s_{q1}\right)}\ T\right) \cap \left(\hat{\sigma}_{\left(S_2\ \hat{\theta}(d_2,\xi_2)\ s_{q2}\right)}\ T\right)\ . \tag{3}$$

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

Property 1.1. *Special case where $s_{q1} = s_{q2} = s_q$.*

$$\left(\hat{\sigma}_{\left(S\ \hat{\theta}(d,\xi_1)\ s_q\right)}\ T\right)\ \cap\ \left(\hat{\sigma}_{\left(S\ \hat{\theta}(d,\xi_2)\ s_q\right)}\ T\right)\ =$$
$$\hat{\sigma}_{\left(S\ \hat{\theta}(d,\xi_1)\ s_q\right) \wedge \left(S\ \hat{\theta}(d,\xi_2)\ s_q\right)}\ T\ =\ \hat{\sigma}_{\left(S\ \hat{\theta}(d,\min(\xi_1,\xi_2))\ s_q\right)}\ T\ . \tag{4}$$

Property 2. *Disjunctions of $\hat{\theta}$ operators can be rewritten into a sequence of union operations as follows.*

$$\hat{\sigma}_{\left(S_1\ \hat{\theta}(d_1,\xi_1)\ s_{q1}\right) \vee \left(S_2\ \hat{\theta}(d_2,\xi_2)\ s_{q2}\right)}\ T\ =$$
$$\left(\hat{\sigma}_{\left(S_1\ \hat{\theta}(d_1,\xi_1)\ s_{q1}\right)}\ T\right)\ \cup\ \left(\hat{\sigma}_{\left(S_2\ \hat{\theta}(d_2,\xi_2)\ s_{q2}\right)}\ T\right)\ . \tag{5}$$

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

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

$$\left(\hat{\sigma}_{\left(S\ \hat{\theta}(d,\xi_1)\ s_q\right)}\ T\right)\ \cup\ \left(\hat{\sigma}_{\left(S\ \hat{\theta}(d,\xi_2)\ s_q\right)}\ T\right)\ =$$
$$\hat{\sigma}_{\left(S\ \hat{\theta}(d,\xi_1)\ s_q\right) \vee \left(S\ \hat{\theta}(d,\xi_2)\ s_q\right)}\ T\ =\ \hat{\sigma}_{\left(S\ \hat{\theta}(d,\max(\xi_1,\xi_2))\ s_q\right)}\ T\ . \tag{6}$$

Properties 3 and 4 explore the commutativity of $\hat{\sigma}$ operation with its composition and traditional operation.

Property 3. *The R_q selection operation commutes under its composition, i.e.,*

$$\hat{\sigma}_{\left(S_1\ \hat{\theta}(d_1,\xi_1)\ s_{q1}\right)}\left(\hat{\sigma}_{\left(S_2\ \hat{\theta}(d_2,\xi_2)\ s_{q2}\right)}\ T\right) = \hat{\sigma}_{\left(S_2\ \hat{\theta}(d_2,\xi_2)\ s_{q2}\right)}\left(\hat{\sigma}_{\left(S_1\ \hat{\theta}(d_1,\xi_1)\ s_{q1}\right)}\ T\right). \tag{7}$$

Property 4. *The R_q selection operation and the traditional selection operation commute under their composition, i.e.,*

$$\hat{\sigma}_{\left(S\ \hat{\theta}(d,\xi)\ s_q\right)}\left(\sigma_{(A\ \theta\ a)}\ T\right) = \sigma_{(A\ \theta\ a)}\left(\hat{\sigma}_{\left(S\ \hat{\theta}(d,\xi)\ s_q\right)}\ T\right)\ . \tag{8}$$

As $\hat{\sigma}$ operation is commutative with σ operation, Properties 1 and 2 can also be employed for these operations. Therefore, we can use Properties 1 and 2 with either the $\hat{\sigma}$ operation only or $\hat{\sigma}$ and σ operations.

The next set of properties involving $\hat{\sigma}$ allows pushing range selection through the traditional binary operators: union ($\cup$), intersection ($\cap$), difference ($-$), cross product ($\times$) and join ($\bowtie$). Property 5 shows that $\hat{\sigma}$ is distributive over the set binary operators $\cup$, $\cap$ and $-$. The relations T_1 and T_2 must be union compatible.

Property 5. *The operator $\hat{\sigma}$ is distributive over the set binary operators $\cup$, $-$ and $\cap$ as follows.*

Property 5.1. *For union, the following expression holds:*

$$\hat{\sigma}_{\left(S\ \hat{\theta}(d,\xi)\ s_q\right)}\left(T_1\ \cup\ T_2\right)\ =\ \left(\hat{\sigma}_{\left(S\ \hat{\theta}(d,\xi)\ s_q\right)}\ T_1\right)\ \cup\ \left(\hat{\sigma}_{\left(S\ \hat{\theta}(d,\xi)\ s_q\right)}\ T_2\right)\ . \tag{9}$$

Property 5.2. *For difference, the following expression holds:*

$$\begin{aligned}\hat{\sigma}_{\left(S\ \hat{\theta}(d,\xi)\ s_q\right)}\left(T_1\ -\ T_2\right)\ &=\ \left(\hat{\sigma}_{\left(S\ \hat{\theta}(d,\xi)\ s_q\right)}\ T_1\right)\ -\ \left(\hat{\sigma}_{\left(S\ \hat{\theta}(d,\xi)\ s_q\right)}\ T_2\right)\\ &=\ \left(\hat{\sigma}_{\left(S\ \hat{\theta}(d,\xi)\ s_q\right)}\ T_1\right)\ -\ T_2\ .\end{aligned} \tag{10}$$

Property 5.3. *For intersection, the following expression holds:*

$$\begin{aligned}\hat{\sigma}_{\left(S\ \hat{\theta}(d,\xi)\ s_q\right)}\left(T_1\ \cap\ T_2\right)\ &=\ \left(\hat{\sigma}_{\left(S\ \hat{\theta}(d,\xi)\ s_q\right)}\ T_1\right)\ \cap\ \left(\hat{\sigma}_{\left(S\ \hat{\theta}(d,\xi)\ s_q\right)}\ T_2\right)\\ &=\ \left(\hat{\sigma}_{\left(S\ \hat{\theta}(d,\xi)\ s_q\right)}\ T_1\right)\ \cap\ T_2\\ &=\ T_1\ \cap\ \left(\hat{\sigma}_{\left(S\ \hat{\theta}(d,\xi)\ s_q\right)}\ T_2\right)\ .\end{aligned} \tag{11}$$

Regarding the binary join ($\bowtie$) and cross product ($\times$) operators, $\hat{\sigma}$ must be distributed to the relation that has the complex attribute mentioned in the condition. This is represented in Property 6.

Property 6. *When the complex attribute mentioned in the range predicate belongs to only one of the joined relations, the operation $\hat{\sigma}$ is distributive over $\bowtie$ or $\times$. Let T_1 be the relation that has the complex attribute S. Thus:*

$$\hat{\sigma}_{\left(S\ \hat{\theta}(d,\xi)\ s_q\right)}\left(T_1\ \theta\ T_2\right)\ =\ \left(\hat{\sigma}_{\left(S\ \hat{\theta}(d,\xi)\ s_q\right)}\ T_1\right)\ \theta\ T_2\ , \tag{12}$$

for any $\theta = \bowtie$ or $\times$.

Properties 1 to 6 show that range selection shares the same algebraic equivalences as the traditional selection. Moreover, Property 4 shows the commutativity property between similarity-based selections and traditional ones. This is an important result, as it allows the RDBMS query optimizer to treat range selection as traditional selection. Therefore, we can use the symbol σ instead of $\hat{\sigma}$ to represent range selections, only using $\hat{\theta}$ to represent the range operator, without lost of generality.

2.3.2 *k*-Nearest Neighbor Selection - $\ddot{\sigma}$.

Distinctly from range and traditional selections, kNN selections have only three properties to rewrite algebraic expressions. Property 7 regards the conjunctive selection conditions, as follows:

Property 7. *Conjunctions of $\ddot{\theta}$ operators can be rewritten into a sequence of intersection operations but they cannot be rewritten as a cascade of individual $\ddot{\sigma}$ operations, i. e.,*

$$\ddot{\sigma}_{\left(S_1\ \ddot{\theta}(d_1,k_1)\ s_{q1}\right) \wedge \left(S_2\ \ddot{\theta}(d_2,k_2)\ s_{q2}\right)} T = \left(\ddot{\sigma}_{\left(S_1\ \ddot{\theta}(d_1,k_1)\ s_{q1}\right)} T\right) \cap \left(\ddot{\sigma}_{\left(S_2\ \ddot{\theta}(d_2,k_2)\ s_{q2}\right)} T\right) \ ; \tag{13}$$

$$\ddot{\sigma}_{\left(S_1\ \ddot{\theta}(d_1,k_1)\ s_{q1}\right) \wedge \left(S_2\ \ddot{\theta}(d_2,k_2)\ s_{q2}\right)} T \neq \ddot{\sigma}_{\left(S_1\ \ddot{\theta}(d_1,k_1)\ s_{q1}\right)} \left(\ddot{\sigma}_{\left(S_2\ \ddot{\theta}(d_2,k_2)\ s_{q2}\right)} T\right) \ . \tag{14}$$

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

Property 7.1. *Special case where $s_{q1} = s_{q2} = s_q$.*

$$\left(\ddot{\sigma}_{\left(S\ \ddot{\theta}(d,k_1)\ s_q\right)} T\right) \cap \left(\ddot{\sigma}_{\left(S\ \ddot{\theta}(d,k_2)\ s_q\right)} T\right) = \ddot{\sigma}_{\left(S\ \ddot{\theta}(d,k_1)\ s_q\right) \wedge \left(S\ \ddot{\theta}(d,k_2)\ s_q\right)} T = \ddot{\sigma}_{\left(S\ \ddot{\theta}(d,\min(k_1,k_2))\ s_q\right)} T \ . \tag{15}$$

For disjunctive conditions, Property 8 is valid.

Property 8. *Disjunctions of $\ddot{\theta}$ operators can be rewritten into a sequence of union operations as follows (this property requires that the relation T is a set because, in this way, duplications will be correctly eliminated):*

$$\ddot{\sigma}_{\left(S_1\ \ddot{\theta}(d_1,k_1)\ s_{q1}\right) \vee \left(S_2\ \ddot{\theta}(d_2,k_2)\ s_{q2}\right)} T = \left(\ddot{\sigma}_{\left(S_1\ \ddot{\theta}(d_1,k_1)\ s_{q1}\right)} T\right) \cup \left(\ddot{\sigma}_{\left(S_2\ \ddot{\theta}(d_2,k_2)\ s_{q2}\right)} T\right) \ . \tag{16}$$

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

Property 8.1. *Special case where $s_{q1} = s_{q2} = s_q$.*

$$\left(\ddot{\sigma}_{\left(S\ \ddot{\theta}(d,k_1)\ s_q\right)} T\right) \cup \left(\ddot{\sigma}_{\left(S\ \ddot{\theta}(d,k_2)\ s_q\right)} T\right) = \ddot{\sigma}_{\left(S\ \ddot{\theta}(d,k_1)\ s_q\right) \vee \left(S\ \ddot{\theta}(d,k_2)\ s_q\right)} T = \ddot{\sigma}_{\left(S\ \ddot{\theta}(d,\max(k_1,k_2))\ s_q\right)} T \ . \tag{17}$$

The commutativity property should not be applied to $\ddot{\theta}$ operator when query elements s_{q1} and s_{q2} are distinct. However, the following property holds.

Property 9. *For complex conditions, each selection should be executed separately and the intersection (for conjunctions) or the union (for disjunctions) of results must be returned, since the operator $\ddot{\sigma}$ is not commutative neither with other selection operators nor with itself. That is, for conjunctive conditions:*

$$\left(\ddot{\sigma}_{(S_1\ \ddot{\theta}(d_1,k_1)\ s_{q1})} T\right) \cap \left(\ddot{\sigma}_{(S_2\ \ddot{\theta}(d_2,k_2)\ s_{q2})} T\right) = \left(\ddot{\sigma}_{(S_2\ \ddot{\theta}(d_2,k_2)\ s_{q2})} T\right) \cap \left(\ddot{\sigma}_{(S_1\ \ddot{\theta}(d_1,k_1)\ s_{q1})} T\right) \ ; \tag{18}$$

and for disjunctive conditions:

$$\left(\ddot{\sigma}_{(S_1\ \ddot{\theta}(d_1,k_1)\ s_{q1})}\ T\right)\ \cup\ \left(\ddot{\sigma}_{(S_2\ \ddot{\theta}(d_2,k_2)\ s_{q2})}\ T\right)\ =$$
$$\left(\ddot{\sigma}_{(S_2\ \ddot{\theta}(d_2,k_2)\ s_{q2})}\ T\right)\ \cup\ \left(\ddot{\sigma}_{(S_1\ \ddot{\theta}(d_1,k_1)\ s_{q1})}\ T\right)\ . \tag{19}$$

The same property can be employed combining either "$\ddot{\sigma} \cap / \cup \hat{\sigma}$" or "$\ddot{\sigma} \cap / \cup \sigma$".

When query elements s_{q1} and s_{q2} are the same, there are special cases where the kNN selection operation becomes commutative with range and self selection operations. First, for a composition of kNN selection operation, this expression can be rewritten in the conjunction of kNN condition; therefore, only the kNN selection with the smallest k condition needs to be executed. Second, for a composition of kNN and range selection operation, this expression can also be rewritten in the conjunction of kNN and range condition; then, the intersection of the results from both basic operators should be executed. Finally, for the disjunction of range and kNN condition, the union of the results from both basic operators needs to be executed.

For the set of properties involving traditional binary operators, no property involving $\ddot{\sigma}$ exists, because $\ddot{\sigma}$ is not distributive over these operators.

As kNN selection operations accepts only three properties (7, 8 and 9) and five special cases (over the same query element), they do not allow optimization algorithms equivalent to traditional and range selections. Thus, specific optimization algorithms should be implemented in the query optimizer to optimize this kind of selection. The *kAndRange* and the *kOrRange* algorithms [8] are examples specifically created to handle the commutativity property of *Range* and kNN query over the same query element.

3 Experimental Results

In this section, we present experiments comparing the evaluation of the similarity queries both optimized and not optimized using the properties of the Similarity Algebra presented in Section 2. To obtain the measurements, this algebra was incorporated into the SIREN query optimizer. SIREN is a similarity retrieval engine that allows expressing similarity queries in SQL and executing them [10]. We call the new version of SIREN able to perform optimization on queries involving similarity as SIREN+O. The experiments analyze the performance of SIREN and SIREN+O to execute similarity queries. As we will see here, the first results show that the proposed algebra leads SIREN+O to perform faster than SIREN.

The test framework was implemented in C++, and the experiments ran 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. Due to space limitations, we only highlight here the performance regarding total time (in milliseconds) as it summarizes the whole computational cost. Four data sets were used:

- *RCCMammography*: a set of 658 medical images obtained from mammograms of right breast with cranio-caudal view (CC). They were compared using the texture distance function [11];
- *RMLOMammography*: a set of 695 medical images obtained from mammography exams of right breast with medio-lateral oblique view (MLO). They were also compared using the same texture distance function [11].
- *MedImage*: a set of 5,180 medical images obtained from three human body parts (abdomen, cranium and thorax) by computerized tomographies (CT). They were compared using the metric histogram distance function [12].
- *PeruDistricts*: a set of 1,829 Peruvian districts. They were compared using the Euclidean distance function.

The first three sets were obtained from the Clinical Hospital at Ribeirão Preto of the University of São Paulo and the last set was obtained from Peru Instituto Nacional de Estadística e Informática (INEI).

The experiments evaluated the execution time of Queries **Q**1 over RCCMammography and RMLOMammography data sets, **Q**2 over MedImage data set and **Q**3 over PeruDistricts data set stated in Section 1. The queries were performed 30 times and the values shown are the average of performing the same query, but varying query elements s_q.

Table 1 summarizes the results executing SIREN and SIREN+O using both sequential scan and the Slim tree index, a well-known index structure for metric data [13].

Query **Q**1 involves a traditional join and a range selection and it can be algebraically expressed as $\hat{\sigma}_{(S\ \hat{\theta}(texture,0.1)\ s_q)}\ (RCC \bowtie RMLO)$. Property 6 was employed to optimize the query. Its gain was about *30.39%* using sequential scan and *30.16%* using a Slim-tree index.

Both Queries **Q**2 and **Q**3 involve traditional selection, range selection and kNN selection. Therefore, Properties 4 and 9 as well as their special cases should be used to optimize these queries. **Q**2 can be algebraically expressed as: $\sigma_{(pathology=`N')}\ \left(\hat{\sigma}_{(S\ \hat{\theta}(texture,0.05)\ s_q)}\ \left(\ddot{\sigma}_{(S\ \ddot{\theta}(texture,5)\ s_q)}\ (MedImage)\right)\right)$. The gain obtained was about *64.68%* using a Slim-tree index and *62.92%* using sequential scan.

Q3 can be expressed as: $\sigma_{(adultpop > oldpop)}\ (\hat{\sigma}_{(S\ \hat{\theta}(Euclidean,15)\ s_q)}\ (\ddot{\sigma}_{(S\ \ddot{\theta}(Euclidean,15)\ s_q)}(PeruDistricts)))$, and the gain obtained was about *63.82%* using sequential scan and *62.61%* using a Slim-tree index.

	SIREN		SIREN+O	
	Sequential scan	Slim tree	Sequential scan	Slim tree
Q1	354.70	331.20	246.90	231.30
Q2	948.50	765.60	351.70	270.40
Q3	604.70	443.20	218.80	165.70

Table 1. Performance of Queries **Q**1, **Q**2 and **Q**3 (total time in milliseconds).

4 Conclusion

Nowadays, storing and retrieving multimedia data is a requirement that must be provided by RDBMS. In order to allow query compilers to optimize a similarity query execution, we presented here the properties holding for the unary similarity operators, that is, for Range and k-Nearest Neighbor Selection operations. We also presented the experiments conducted to show the performance obtained with SIREN query optimizer using Similarity Algebra (SIREN+O) reducing total time in up to *64%* over the performance of queries in SIREN without the algebra regardless of the usage of an index. As a follow-up of this paper, we are currently working on the properties to extend the Similarity Algebra to support similarity join. This will surely open the possibility to support the storage and retrieval of complex data in RDBMS. We are also working on developing better statistics that can be measured over data, to create heuristics able to control the DBMS query optimizer for similarity queries.

References

1. Codd, E.F.: A relational model of data for large shared data banks. CACM **13**(6) (1970) 377–387
2. Adali, S., Bonatti, P., Sapino, M., Subrahmanian, V.: A multi-similarity algebra. In: SIGMOD, ACM Press (1998) 402–413
3. Atnafu, S., Chbeir, R., Coquil, D., Brunie, L.: Integrating similarity-based queries in image DBMSs. In: SAC, ACM Press (2004) 735–739
4. Belohlávek, R., Opichal, S., Vychodil., V.: Relational algebra for ranked tables with similarity: properties and implementation. In: IDA. Volume 4723 of LNCS., Springer Verlag (2007) 140–151.
5. Ciaccia, P., Montesi, D., Penzo, W., Trombetta, A.: Imprecision and user preferences in multimedia queries: a generic algebraic approach. In: FoIKS. Volume 1762 of LNCS., Springer Verlag (2000) 50–71
6. Adali, S., Bufi, C., Sapino, M.L.: Ranked relations: query languages and query processing methods for multimedia. MTAJ **24**(3) (2004) 197–214
7. Li, C., Chang, K.C.C., Ilyas, I.F., Song, S.: RankSQL: query algebra and optimization for relational top-k queries. In: SIGMOD, ACM Press (2005) 131–142
8. Traina-Jr., C., Traina, A.J.M., Vieira, M.R., Arantes, A.S., Faloutsos, C.: Efficient processing of complex similarity queries in RDBMS through query rewriting. In: CIKM, ACM Press (2006) 4–13
9. Ferreira, M.R.P., Traina-Jr., C., Traina, A.J.M., Dias, I.: Extending SQL to support unary similarity queries. Technical Report 325, ICMC/USP (2008)
10. Barioni, M.C.N., Razente, H.L., Traina, A.J.M., Traina-Jr., C.: SIREN: A similarity retrieval engine for complex data. In: VLDB, ACM Press (2006) 1155–1158
11. Felipe, J.C., Traina, A.J.M., Traina-Jr., C.: Retrieval by content of medical images using texture for tissue identification. In: CBMS, IEEE Computer Society (2003) 175–180
12. Traina, A.J.M., Traina-Jr., C., Bueno, J.M., Chino, F.J.T., Azevedo-Marques, P.: Efficient content-based image retrieval through metric histograms. WWW **6**(2) (2003) 157–185
13. Traina-Jr., C., Traina, A.J.M., Seeger, B., Faloutsos, C.: Slim-trees: High performance metric trees minimizing overlap between nodes. In: EDBT. Volume 1777 of LNCS., Springer Verlag (2000) 51–65