Properties of the Unary Similarity Operator Integrated to the Relational Algebra*

Mônica Ribeiro Porto Ferreira[1], Caetano Traina Junior[1],
Agma Juci Machado Traina[1], and Ires Dias[2]

[1] Computer Science Dept., ICMC-Univ. of Sao Paulo Sao Carlos, SP, 13560-Brazil,
{monika, caetano, agma}@icmc.usp.br
[2] Mathematics Dept., ICMC-Univ. of Sao Paulo, Sao Carlos, SP, 13560-Brazil,
iresdias@icmc.usp.br

Abstract. Conventional operators for data comparison are either based on exact matching or on total order relationship among elements. Neither of them are appropriate to manage complex data, such as multimedia data (e.g., images, audio and large texts), time series and genetic sequences. In fact, the most 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 support, an extension to the Relational Algebra is under development at GBdI-ICMC-USP (Databases and Images Group), aiming to represent similarity queries in algebraic expressions. This technical report defines a formal framework which embodies the unary similarity operators into Relational Algebra, and precisely define their algebraic properties when used in query expressions either alone or combined with the existing exact matching and relational operators. We also show how to take advantage of such properties to optimize queries that include similarity operators, presenting a similarity query optimizer developed for SIREN (the Similarity Retrieval Engine), which uses a existing RDBMS to answer similarity queries.

1 Introduction

In 1970, Codd [1] introduced the relational model, which is the base of the actual commercial Database Management Systems (DBMS). It is based on the mathematical relation theory: the database is presented 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. Currently, with the advent of new applications, the relational DBMS (RDBMS) should be able to support new data types, operators and kinds of queries. For instance, in a Geographic Information System (GIS),

* This work has been supported by FAPESP (under grant 05/03341-7), CNPq, and CAPES/-Fulbright.

the RDBMS was adapted to store spatial data types and the corresponding operators and queries that involve the notion of spatial dimension [2].

However, there are many complex domains, such as images, audios, videos, genomic sequences, and time series, where the conventional operators do not apply and the notion of spatial dimension does not hold either. Therefore, similarity emerges as the natural way to compare pairs of elements in complex domains and the degree of similarity between data is the most important factor [3].

With the growing interest to store multimedia data in RDBMS and the need to retrieve them by content, the algebraic properties of similarity operators must be precisely stated, in order to allow query compilers to optimize similarity query execution. To illustrate interesting queries that require similarity comparison, following we give examples using a dataset for GIS with demographic data, and a dataset of images, one of the most common type of multimedia data.

"*Find the 5 nearest cities to 'São Carlos - SP', which latitude = -22.02 and longitude = 47.89, provided they are not father then 0.3 distance units, considering Euclidean distance function (L_2)*".

"*In a set of X-Ray exams, show the X-Ray exams of any patient that are the most similar to X-Ray exam from patient X, using color distribution*".

In order to execute this query, it is necessary provide a measurement of how to quantify similarity between two elements. Usually, it is done 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 $\mathbb{S}$, i.e., the smaller distance means that the elements are closer or more similar. 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 $s_1 \neq s_2$ e $d(s_1, s_1) = 0$; and (*iii*) triangular inequality: $d(s_1, s_2) \leq d(s_1, s_3) + d(s_3, s_2), \forall s_1, s_2, s_3 \in \mathbb{S}$, where $s_1, s_2, s_3 \in \mathbb{S}$.

As traditional queries, similarity ones are based on comparison condition called similarity condition, which should be unary (similarity selection) or binary (similarity join). This work presents the Similarity Algebra and their properties for unary similarity operators. The goal of this work is to incorporate unary similarity queries into RDBMS.

The remainder of this technical report is structured as follows. The Section 2 summarizes existing related works. Section 3 presents the Similarity Algebra. Finally, Section 4 concludes this technical report.

2 Background

It makes sense to include similarity queries into RDBMS even if its only benefit would be to support complex data besides traditional ones. One way to offer this support is including similarity operators into relational algebra and processing similarity queries in each module efficiently (query compilation, query optimization and query execution) of the RDBMS query processing engine.

In order to enable similarity queries into RDBMS, Barioni et al. [4] proposed a similarity retrieval engine, called SIREN (*SImilarity Retrieval ENgine*), that allows expressing similarity queries in SQL and executing them. SIREN is a service in a three tier architecture implemented between a RDBMS and the application programs. It intercepts every SQL command sent from the application, analyzes and treats every similarity-related constructions and references to complex elements, if there exist. SQL commands which contain neither similarity construction nor references to complex elements are sent to the RDBMS and the answers are relayed from the RDBMS back to the application program. Commands containing similarity-related constructions are processed at SIREN [5].

In its first version, SIREN did not had an optimization module implemented; its mains component was the query compiler, which makes the lexical and the syntactical analysis. For this reason, it did not process similarity queries efficiently. To solve this problem, Ferreira et al. [6] proposed the SIREN query optimizer, which is analogous to RDBMS query optimizer. Figure 1 shows SIREN architecture.

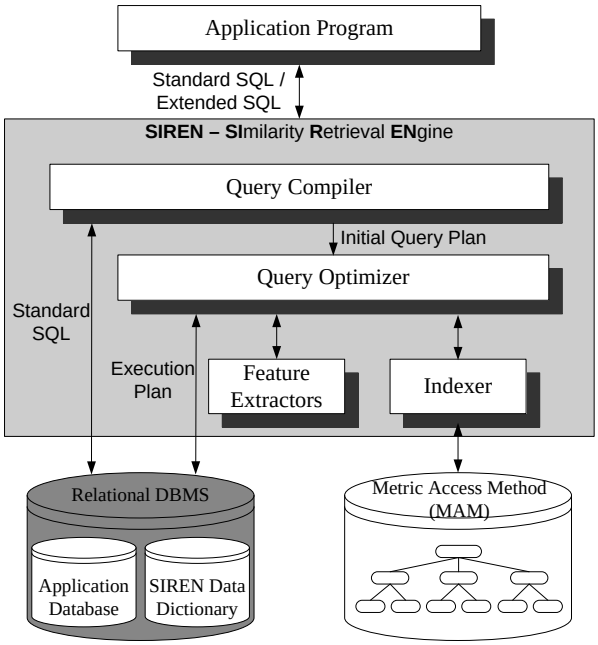

Fig. 1. SIREN architecture.

In a RDBMS, the query optimizer is responsible for generating alternative plans employing algebraic properties; evaluating each plan to estimate its execution cost; and choosing the plan with the minimum expected computational cost to be executed. The optimization is achieved rewriting the original query into another expression whose

evaluation cost is expected to be smaller, using algebraic properties to ensure that the rewritten expression is equivalent to the original one.

The optimizer is able to rewrite queries based on operators presented in the algebra. In the literature, there are several extensions of relational algebra aiming to include similarity concepts in RDBMS from various perspectives, as followed.

The first algebra to consider similarity queries was a *Multi-Similarity Algebra* (MSA), presented by Adali et al. [7]. This algebra has been designed to integrate different interpretation 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 [8]. Therefore, it is not fully consistent to the relational model.

Other works associated similarity to uncertainty and provided fuzzy logic-based methods to solve this [9, 10]. 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, there are approaches based on the notion of ranking, i.e., ordering among tuples or elements [11, 12]. 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 the 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. [13] propose an extension of relational algebra considering complex similarity queries with two or more similarity predicates combined with Boolean operators $\wedge$ (and), $\vee$ (or) and $\neg$ (not). However, they have only treated queries with a unique element center, i.e., with the same query element, which is very restrictive and does not cover all cases occurring in a RDBMS.

In this work, a *Similarity Algebra* aiming at integrating both unary similarity operators with relational algebra, which allows optimizing similarity queries in RDBMS. Then, the proposed algebra was incorporated into SIREN query optimizer to allow it to optimize similarity queries.

3 Similarity Algebra

3.1 Definitions

An attribute is comparable by similarity only if it is associated to a similarity measure (that is, to a distance function 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 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 technical report, 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 an relation 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 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 relation, in the remainder of the technical report, we will use just S and $\mathbb{S}$ to refer to complex attribute S_j and its respective domain $\mathbb{S}_j$, and A and $\mathbb{A}$ to refer to simple attribute A_h and its respective domain $\mathbb{A}_h$ whenever the focus of the text is over only one attribute. Figure 2 shows the extended relational model to similarity queries.

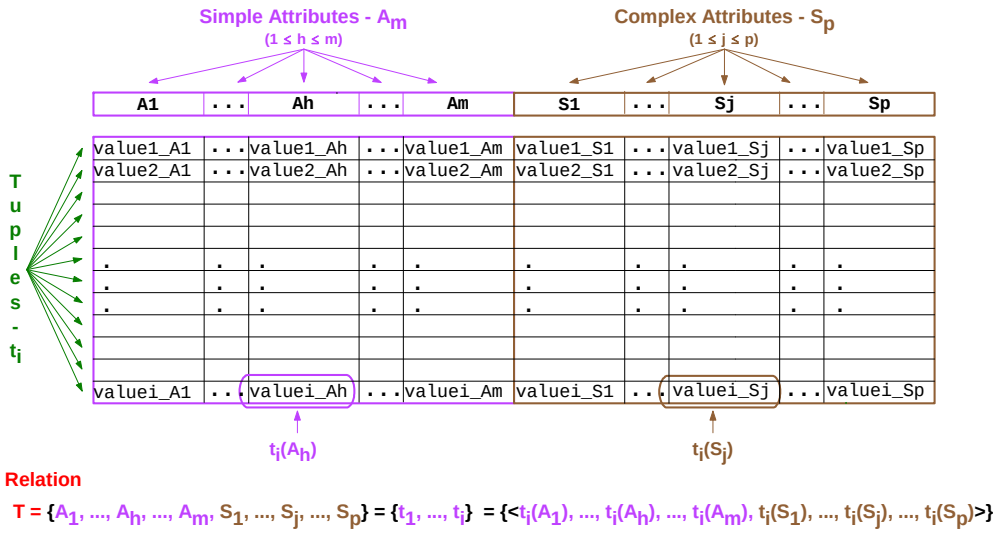

Fig. 2. Extended relational model - Similarity queries

3.2 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 selections, 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. The range, kNN and their variations are described as follows. The first definition showed is the **Range query**.

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 $\hat{\sigma}_{(S\;\hat{\theta}(d,\xi)\;s_q)}\,T$ returns every tuple $t_i \in T$ such that $d(t_i(S), s_q) \leq \xi$. This can be formally defined as:*

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

There is a special case of Range query, called **Point query**, in which the similarity threshold is $\xi = \mathbf{0}$. The goal of this query is to verify if $s_q \in T$.

The complementary operation of Range query is called **Reversed Range query** - R_q^{-1}.

Definition 2. ***Reversed Range query*** **-** *R_q^{-1}: 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 $\hat{\sigma}_{(S\;\hat{\theta}^{-1}(d,\xi)\;s_q)}\,T$ returns every tuple $t_i \in T$ such as $d(t_i(S), s_q) > \xi$. This can be formally defined as:*

$$\hat{\sigma}_{\left(S\;\hat{\theta}^{-1}(d,\xi)\;s_q\right)}\,T = \left\{t_i \in T \mid d\left(t_i\left(S\right), s_q\right) > \xi\right\}\ . \tag{2}$$

The k**-Nearest Neighbor query** - kNN is defined as follow.

Definition 3. ***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 $\ddot{\sigma}_{(S\;\ddot{\theta}(d,k)\;s_q)}\,T$ returns the tuples from T whose value of the attributes S is one of the k elements in S nearest to the query element s_q based on the distance function d. This can be defined as:*

$$\ddot{\sigma}_{\left(S\;\ddot{\theta}(d,k)\;s_q\right)}\,T = T' = \left\{t_i \in T \mid \forall\, t_g \in [T - T'], T' = t_{i=1,\ldots,k},\right.$$
$$\left. d\left(t_i\left(S\right), s_q\right) \leq d\left(t_g\left(S\right), s_q\right)\right\}\ . \tag{3}$$

The complementary operation of k-Nearest Neighbor query is the k**-Farthest Neighbor query** - kFN.

Definition 4. ***k-Farthest Neighbor query*** **-** *kFN: 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*

$\ddot{\sigma}_{(S\ \ddot{\theta}_F(d,k)\ s_q)}\, T$ *returns the tuples from* T *whose value of the attribute* S *is one of the* k *elements in* S *farthest to the query element* s_q *based on the distance function* d*. This can be defined as:*

$$\ddot{\sigma}_{\left(S\ \ddot{\theta}_F(d,k)\ s_q\right)}\, T = T' = \{t_i \in T \mid \forall\, t_g \in [T - T'], T' = t_{i=1,\ldots,k}, \\ d\left(t_i\left(S\right), s_q\right) > d\left(t_g\left(S\right), s_q\right)\} \ . \tag{4}$$

3.3 Algebraic Properties

The query optimizer of RDBMS 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 present algebraic properties useful to rewrite expressions of both range / reversed range operators ($\hat{\theta}$ / $\hat{\theta}^{-1}$) and k-nearest / k-farthest neighbor operators ($\ddot{\theta}$ / $\ddot{\theta}_F$).

3.3.1 Range / Reversed Range Selection - $\hat{\sigma}$.

The properties definitions present in this subsection use the range $\hat{\theta}$ operator. The same property definitions can be used for the reversed range $\hat{\theta}^{-1}$ operator.

The Property 1 is valid by conjunctive selection condition as follows:

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.,*

$$\hat{\sigma}_{\left(S_1\ \hat{\theta}(d_1,\xi_1)\ s_{q1}\right)\ \wedge\ \left(S_2\ \hat{\theta}(d_2,\xi_2)\ s_{q2}\right)}\, T \overset{1}{=} \left(\hat{\sigma}_{\left(S_1\ \hat{\theta}(d_1,\xi_1)\ s_{q1}\right)}\, T\right) \cap \left(\hat{\sigma}_{\left(S_2\ \hat{\theta}(d_2,\xi_2)\ s_{q2}\right)}\, T\right) \\ \overset{2}{=}\ \hat{\sigma}_{\left(S_1\ \hat{\theta}(d_1,\xi_1)\ s_{q1}\right)} \left(\hat{\sigma}_{\left(S_2\ \hat{\theta}(d_2,\xi_2)\ s_{q2}\right)}\, T\right). \tag{5}$$

Proof. We use the Definition 1 to prove that:

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

On the other hand,

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

□

This property is valid regardless of query elements s_{q1} and s_{q2} being the same or not. A special case exists if the conjunction of R_q conditions are executed over the same query element, i.e., $s_{q1} = s_{q2} = s_q$, then only the R_q selection with the smallest ξ condition needs to be executed [13]. That is,

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

$$\left(\hat{\sigma}_{\left(S\ \hat{\theta}(d,\xi_1)\ s_q\right)} T\right) \cap \left(\hat{\sigma}_{\left(S\ \hat{\theta}(d,\xi_2)\ s_q\right)} T\right) = \hat{\sigma}_{\left(S\ \hat{\theta}(d,\xi_1)\ s_q\right) \wedge \left(S\ \hat{\theta}(d,\xi_2)\ s_q\right)} T\ =\ \hat{\sigma}_{\left(S\ \hat{\theta}(d,\min(\xi_1,\xi_2))\ s_q\right)} T. \tag{6}$$

For disjunctive conditions, the Property 2 is valid.

Property 2. *Disjunctions of* $\hat{\theta}$ *operators can be rewritten into a sequence of union operations as follows. This property requires that the relation* T *be a set because, in this way, duplications will be correctly eliminated.*

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

Proof. We can use the Definition 1 to prove that:

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

□

This property is valid regardless of query elements s_{q1} and s_{q2} being the same or not. A special case exists if the disjunction of R_q conditions are executed over the same query element, i.e., $s_{q1} = s_{q2} = s_q$, then only the R_q selection with the largest ξ condition needs to be executed [13]. That is,

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

$$\left(\hat{\sigma}_{\left(S\ \hat{\theta}(d,\xi_1)\ s_q\right)} T\right) \cup \left(\hat{\sigma}_{\left(S\ \hat{\theta}(d,\xi_2)\ s_q\right)} T\right) = \hat{\sigma}_{\left(S\ \hat{\theta}(d,\xi_1)\ s_q\right) \vee \left(S\ \hat{\theta}(d,\xi_2)\ s_q\right)} T\ =\ \hat{\sigma}_{\left(S\ \hat{\theta}(d,\max(\xi_1,\xi_2))\ s_q\right)} T. \tag{8}$$

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.,*

$$\hat{\sigma}_{\left(S_1\ \hat{\theta}(d_1,\xi_1)\ s_{q1}\right)} \left(\hat{\sigma}_{\left(S_2\ \hat{\theta}(d_2,\xi_2)\ s_{q2}\right)} T\right) = \hat{\sigma}_{\left(S_2\ \hat{\theta}(d_2,\xi_2)\ s_{q2}\right)} \left(\hat{\sigma}_{\left(S_1\ \hat{\theta}(d_1,\xi_1)\ s_{q1}\right)} T\right). \tag{9}$$

Proof. This proof is directly obtained by Property 1 and Definition 1.

□

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

$$\hat{\sigma}_{\left(S\ \hat{\theta}(d,\xi)\ s_q\right)}\left(\sigma_{(A\ \theta\ a)}\ T\right) = \sigma_{(A\ \theta\ a)}\left(\hat{\sigma}_{\left(S\ \hat{\theta}(d,\xi)\ s_q\right)}\ T\right). \tag{10}$$

Proof. We use the Definition 1 to prove that:

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

□

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 to push range selection through the 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 e 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:*

$$\hat{\sigma}_{\left(S\ \hat{\theta}(d,\xi)\ s_q\right)}\left(T_1\ \cup\ T_2\right) = \left(\hat{\sigma}_{\left(S\ \hat{\theta}(d,\xi)\ s_q\right)}\ T_1\right)\ \cup\ \left(\hat{\sigma}_{\left(S\ \hat{\theta}(d,\xi)\ s_q\right)}\ T_2\right). \tag{11}$$

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

$$\begin{aligned}
\hat{\sigma}_{\left(S\ \hat{\theta}(d,\xi)\ s_q\right)}\left(T_1\ -\ T_2\right) &= \left(\hat{\sigma}_{\left(S\ \hat{\theta}(d,\xi)\ s_q\right)}\ T_1\right)\ -\ \left(\hat{\sigma}_{\left(S\ \hat{\theta}(d,\xi)\ s_q\right)}\ T_2\right) \\
&= \left(\hat{\sigma}_{\left(S\ \hat{\theta}(d,\xi)\ s_q\right)}\ T_1\right)\ -\ T_2.
\end{aligned} \tag{12}$$

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

$$\begin{aligned}
\hat{\sigma}_{\left(S\ \hat{\theta}(d,\xi)\ s_q\right)}\left(T_1\ \cap\ T_2\right) &= \left(\hat{\sigma}_{\left(S\ \hat{\theta}(d,\xi)\ s_q\right)}\ T_1\right)\ \cap\ \left(\hat{\sigma}_{\left(S\ \hat{\theta}(d,\xi)\ s_q\right)}\ T_2\right) \\
&= \left(\hat{\sigma}_{\left(S\ \hat{\theta}(d,\xi)\ s_q\right)}\ T_1\right)\ \cap\ T_2 \\
&= T_1\ \cap\ \left(\hat{\sigma}_{\left(S\ \hat{\theta}(d,\xi)\ s_q\right)}\ T_2\right).
\end{aligned} \tag{13}$$

Proof. Let Property 5.1 and Definition 1, we prove that:

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

For Properties 5.2 and 5.3, the proof is realized in the analogous way.

□

Regarding the binary join ($\bowtie$) and cross product ($\times$) operators, $\hat{\sigma}$ must be distributed to the relation that has all 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 complex attribute S. Thus:*

$$\hat{\sigma}_{\left(S\ \hat{\theta}(d,\xi)\ s_q\right)}\ (T_1\ \theta\ T_2)\ =\ \left(\hat{\sigma}_{\left(S\ \hat{\theta}(d,\xi)\ s_q\right)}\ T_1\right)\ \theta\ T_2, \tag{14}$$

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

Proof. Let $\theta = \times$, $S \in T_1$ and Definition 1, we prove that:

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

The same proof can be applied if $S \in T_2$. For $\theta = \bowtie$, the proof is realized in analogous way.

□

If the R_q condition is conjunctive, in which S_1 is a complex attribute of relation T_1 and S_2 is a complex attribute of relation T_2, Properties 1 and 6 can be used to prove,

$$\hat{\sigma}_{\left(S_1\ \hat{\theta}(d,\xi)\ s_q\right)\ \wedge\ \left(S_2\ \hat{\theta}(d,\xi)\ s_q\right)}\ (T_1\ \theta\ T_2) = \left(\hat{\sigma}_{\left(S_1\ \hat{\theta}(d,\xi)\ s_q\right)}\ T_1\right)\ \theta\ \left(\hat{\sigma}_{\left(S_2\ \hat{\theta}(d,\xi)\ s_q\right)}\ T_2\right), \tag{15}$$

for any $\theta = \times$ ou $\bowtie$; that is, the distributive property applied to the traditional selection operation can also be used to range selection operation. In this way, the Equivalence 15 complete the Property 6.

In summary, Properties 1 to 6 show that range selection shares the same algebraic equivalences as the traditional selection. Moreover, Property 4 shows 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 loss of generality.

3.3.2 *k*-Nearest / *k*-Farthest Neighbor Selection - $\ddot{\sigma}$.

The properties definitions presented in this subsection regards the k-nearest neighbor $\ddot{\theta}$ operator. The same property definitions can be used for the k-farthest neighbor $\ddot{\theta}_F$ operator.

Valid transformation properties to kNN queries are proved algebraically and invalid transformation properties are proved by contradiction. The relations `CitySaoCarlos` and `CityAraraquara` are used in examples. These relations have geographic informations (latitude and longitude) about 12 cities around "São Carlos-SP" (see Figure 3a) and 17 cities around "Araraquara-SP" (see Figure 3b). Elements (black square) connected by black line into query element belong to answer set in all examples.

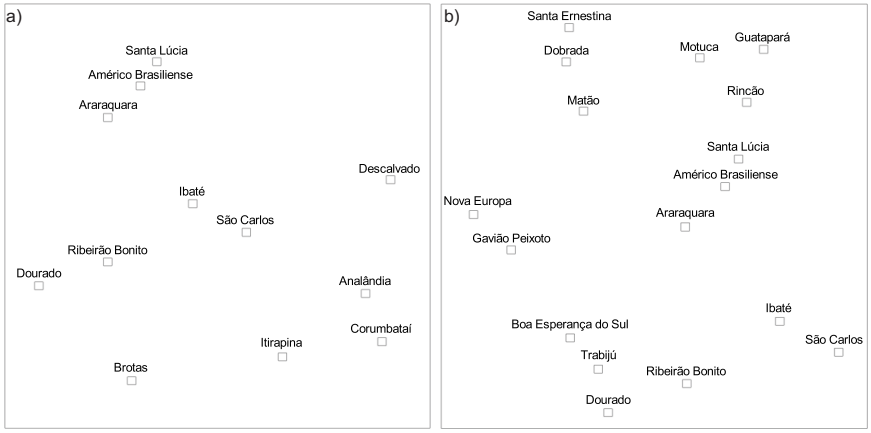

Fig. 3. Relations `CitySaoCarlos` and `CityAraraquara`. a) Relation `CitySaoCarlos`: 12 cities around "São Carlos-SP". b) Relation `CityAraraquara`: 17 cities around "Araraquara-SP".

In the relational algebra, the following expression is valid:

$$\begin{aligned}\sigma_{(A_1\ \theta_1\ a_1)\wedge(A_2\ \theta_2\ a_2)}\ T &= \left(\sigma_{(A_1\ \theta_1\ a_1)}\ T\right) \cap \left(\sigma_{(A_2\ \theta_2\ a_2)}\ T\right) \\ &= \sigma_{(A_1\ \theta_1\ a_1)}\left(\sigma_{(A_2\ \theta_2\ a_2)}\ T\right). \end{aligned} \tag{16}$$

However, when σ is changed by $\ddot{\sigma}$, conjunctions of $\ddot{\theta}$ operators can only be transformed into a sequence of intersection operations, but they cannot be transformed into a cascade of individual $\ddot{\sigma}$ operations. Thus, Property 7 is valid for the conjunctive selection conditions, as follows.

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

$$\ddot{\sigma}_{\left(S_1\ \ddot{\theta}(d_1,k_1)\ s_{q1}\right)\wedge\left(S_2\ \ddot{\theta}(d_2,k_2)\ s_{q2}\right)}\ T = \left(\ddot{\sigma}_{\left(S_1\ \ddot{\theta}(d_1,k_1)\ s_{q1}\right)}\ T\right) \cap \left(\ddot{\sigma}_{\left(S_2\ \ddot{\theta}(d_2,k_2)\ s_{q2}\right)}\ T\right); \tag{17}$$

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

Proof. We use the Definition 3 to prove the Equivalence 17:

$$\begin{aligned}
&\ddot{\sigma}_{\left(S_1\ \ddot{\theta}(d_1,k_1)\ s_{q1}\right)\wedge\left(S_2\ \ddot{\theta}(d_2,k_2)\ s_{q2}\right)}\,T\\
=&\{t_i \in T |\ \forall\, t \in [T-T_1], T_1 = t_{i=1,\ldots,k_1}, d_1(t_i(S_1), s_{q1}) \leq d_1(t(S_1), s_{q1}) \wedge\\
&\qquad \forall\, t \in [T-T_2], T_2 = t_{i=1,\ldots,k_2}, d_2(t_i(S_2), s_{q2}) \leq d_2(t(S_2), s_{q2})\}\\
=&\{t_i \in T |\ \forall\, t \in [T-T_1], T_1 = t_{i=1,\ldots,k_1}, d_1(t_i(S_1), s_{q1}) \leq d_1(t(S_1), s_{q1})\} \cap\\
&\{t_i \in T \mid \forall\, t \in [T-T_2], T_2 = t_{i=1,\ldots,k_2}, d_2(t_i(S_2), s_{q2}) \leq d_2(t(S_2), s_{q2})\}\\
=&\left(\ddot{\sigma}_{\left(S_1\ \ddot{\theta}(d_1,k_1)\ s_{q1}\right)}\,T\right) \cap \left(\ddot{\sigma}_{\left(S_2\ \ddot{\theta}(d_2,k_2)\ s_{q2}\right)}\,T\right).
\end{aligned}$$

□

Proof. We use the following counterexample to prove the Inequality 18.

Example 1. *"Select 5 nearest cities of São Carlos-SP, which latitude* $=-22.02$ *and longitude* $=47.89$*, and Araraquara-SP, which latitude* $=-21.79$ *and longitude* $=48.18$*, considering Euclidean distance* L_2*".*

$$\ddot{\sigma}_{\left(\texttt{coordenada}\ \ddot{\theta}(L_2,5)\ (-22.02,47.89)\right)\wedge\left(\texttt{coordenada}\ \ddot{\theta}(L_2,5)\ (-21.79,48.18)\right)}\texttt{CitySaoCarlos} =$$
$$\ddot{\sigma}_{\left(\texttt{coordenada}\ \ddot{\theta}(L_2,5)\ (-22.02,47.89)\right)}\left(\ddot{\sigma}_{\left(\texttt{coordenada}\ \ddot{\theta}(L_2,5)\ (-21.79,48.18)\right)}\texttt{CitySaoCarlos}\right).$$

The execution of the first algebraic expression returns 2 tuples: {`Ribeirão Bonito, Ibaté`}, and the second expression returns 5 tuples: {`Ribeirão Bonito, Ibaté, Araraquara, Américo Brasiliense, Santa Lúcia`}. So, this proof confirm that Property 7 is valid. □

This property is valid regardless of query elements s_{q1} and s_{q2} being the same or not. A special case exists if the conjunction of kNN conditions are executed over the same query element, i.e., $s_{q1} = s_{q2} = s_q$, then only the kNN selection with the smallest k condition needs to be executed [13]. That is,

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

$$\begin{aligned}
&\left(\ddot{\sigma}_{\left(S\ \ddot{\theta}(d,k_1)\ s_q\right)}\,T\right) \cap \left(\ddot{\sigma}_{\left(S\ \ddot{\theta}(d,k_2)\ s_q\right)}\,T\right) =\\
&\ddot{\sigma}_{\left(S\ \ddot{\theta}(d,k_1)\ s_q\right)\wedge\left(S\ \ddot{\theta}(d,k_2)\ s_q\right)}\,T\ =\ \ddot{\sigma}_{\left(S\ \ddot{\theta}(d,\min(k_1,k_2))\ s_q\right)}\,T.
\end{aligned} \tag{19}$$

For disjunctive conditions, the relational algebra considers valid the following expression:

$$\sigma_{(A_1\ \theta_1\ a_1)\ \vee\ (A_2\ \theta_2\ a_2)}\,T\ =\ \left(\sigma_{(A_1\ \theta_1\ a_1)}\,T\right) \cup \left(\sigma_{(A_2\ \theta_2\ a_2)}\,T\right). \tag{20}$$

If σ is changing by $\ddot{\sigma}$, the expression 20 continue to be valid, generating the Property 8.

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.*

$$\ddot{\sigma}_{\left(S_1\ \ddot{\theta}(d_1,k_1)\ s_{q1}\right) \vee \left(S_2\ \ddot{\theta}(d_2,k_2)\ s_{q2}\right)}\, T = \left(\ddot{\sigma}_{\left(S_1\ \ddot{\theta}(d_1,k_1)\ s_{q1}\right)} T\right) \cup \left(\ddot{\sigma}_{\left(S_2\ \ddot{\theta}(d_2,k_2)\ s_{q2}\right)} T\right). \quad (21)$$

Proof. We use the Definition 3 to prove that:

$$\begin{aligned}
&\ddot{\sigma}_{\left(S_1\ \ddot{\theta}(d_1,k_1)\ s_{q1}\right) \vee \left(S_2\ \ddot{\theta}(d_2,k_2)\ s_{q2}\right)}\, T \\
=&\{t_i \in T |\ \forall\, t \in [T - T_1], T_1 = t_{i=1,\ldots,k_1}, d_1(t_i(S_1), s_{q1}) \leq d_1(t(S_1), s_{q1}) \vee \\
&\qquad \forall\, t \in [T - T_2], T_2 = t_{i=1,\ldots,k_2}, d_2(t_i(S_2), s_{q2}) \leq d_2(t(S_2), s_{q2})\} \\
=&\{t_i \in T |\ \forall\, t \in [T - T_1], T_1 = t_{i=1,\ldots,k_1}, d_1(t_i(S_1), s_{q1}) \leq d_1(t(S_1), s_{q1})\} \cup \\
&\{t_i \in T |\ \forall\, t \in [T - T_2], T_2 = t_{i=1,\ldots,k_2}, d_2(t_i(S_2), s_{q2}) \leq d_2(t(S_2), s_{q2})\} \\
=&\left(\ddot{\sigma}_{\left(S_1\ \ddot{\theta}(d_1,k_1)\ s_{q1}\right)}\, T\right) \cup \left(\ddot{\sigma}_{\left(S_2\ \ddot{\theta}(d_2,k_2)\ s_{q2}\right)}\, T\right).
\end{aligned}$$

□

This property is valid regardless of query elements s_{q1} and s_{q2} being the same or not. A special case exists if the disjunction of kNN conditions are executed over the same query element, i.e., $s_{q1} = s_{q2} = s_q$, then only the kNN selection with the largest k condition needs to be executed [13]. That is,

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

$$\begin{aligned}
&\left(\ddot{\sigma}_{\left(S\ \ddot{\theta}(d,k_1)\ s_q\right)}\, T\right) \cup \left(\ddot{\sigma}_{\left(S\ \ddot{\theta}(d,k_2)\ s_q\right)}\, T\right) = \\
&\ddot{\sigma}_{\left(S\ \ddot{\theta}(d,k_1)\ s_q\right) \vee \left(S\ \ddot{\theta}(d,k_2)\ s_q\right)}\, T \;=\; \ddot{\sigma}_{\left(S\ \ddot{\theta}(d,\max(k_1,k_2))\ s_q\right)}\, T. \qquad (22)
\end{aligned}$$

In relational algebra, the commutativity of σ, expressed by the following expression, is valid.

$$\sigma_{(A_1\ \theta_1\ a_1)}\left(\sigma_{(A_2\theta_2\ a_2)}\, T\right) = \sigma_{(A_2\ \theta_2\ a_2)}\left(\sigma_{(A_1\ \theta_1\ a_1)}\, T\right). \quad (23)$$

Nonetheless, if σ is changed by $\ddot{\sigma}$ and the query elements are distinct, that is, $s_{q1} \neq s_{q2}$, the Expression 23 does not hold. Properties 9, 10 and 11 present the non-commutativity of k-nearest neighbor selection operation with traditional selection operation, range selection operation and self selection operation.

Property 9. *The kNN selection operation does not commute under its composition, i.e.,*

$$\ddot{\sigma}_{\left(S_1\ \ddot{\theta}(d_1,k_1)\ s_{q1}\right)}\left(\ddot{\sigma}_{\left(S_2\ \ddot{\theta}(d_2,k_2)\ s_{q2}\right)}\, T\right) \neq\ \ddot{\sigma}_{\left(S_2\ \ddot{\theta}(d_2,k_2)\ s_{q2}\right)}\left(\ddot{\sigma}_{\left(S_1\ \ddot{\theta}(d_1,k_1)\ s_{q1}\right)}\, T\right). \quad (24)$$

Proof. We use the following counterexample to prove this property.

Example 2. *"Select 5 nearest cities of São Carlos-SP, which latitude $= -22.02$ and longitude $=47.89$, and Araraquara-SP, which latitude $=-21.79$ and longitude $=48.18$, considering Euclidean distance L_2".*

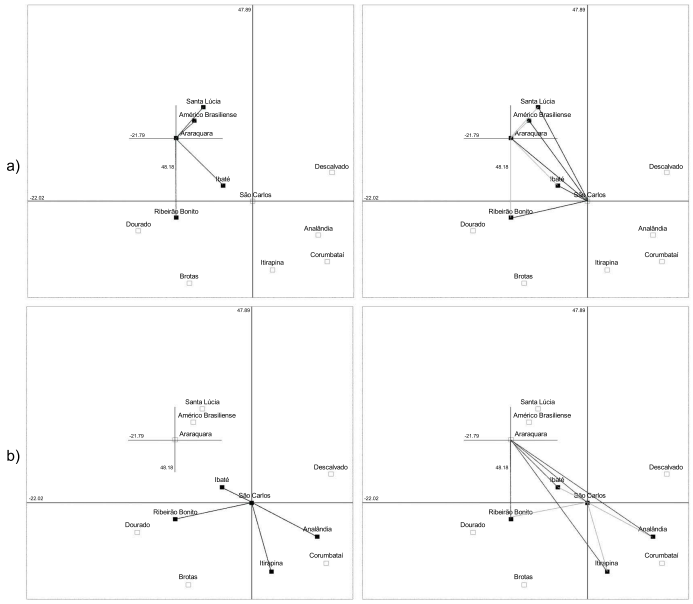

Fig. 4. Non-commutativity of kNN selection operation with self selection operation (query elements distinct). Query presents in Example 2: a) kNN query first executed with s_{q2} = "Araraquara" and, then, with s_{q1} = "São Carlos" in the relation `CitySaoCarlos`. b) kNN query executed first with s_{q1} = "São Carlos" and with s_{q2} = "Araraquara" in the relation `CitySaoCarlos`.

$$\ddot{\sigma}_{\left(\texttt{coordenada}\ \ddot{\theta}(L_2,5)\ (-22.02,47.89)\right)}\left(\ddot{\sigma}_{\left(\texttt{coordenada}\ \ddot{\theta}(L_2,5)\ (-21.79,48.18)\right)}\texttt{CitySaoCarlos}\right) =$$
$$\ddot{\sigma}_{\left(\texttt{coordenada}\ \ddot{\theta}(L_2,5)\ (-21.79,48.18)\right)}\left(\ddot{\sigma}_{\left(\texttt{coordenada}\ \ddot{\theta}(L_2,5)\ (-22.02,47.89)\right)}\texttt{CitySaoCarlos}\right).$$

However, when algebraic expressions are executed, the results are not the same. This can be observed in the Figure 4.

In the Figure 4a, first it is executed kNN selection with s_{q2} = "Araraquara" in the relation `CitySaoCarlos` and, over its results, kNN selection with s_{q1} = "São Carlos". The answer of this query is {`Ribeirão Bonito, Ibaté, Araraquara, Américo Brasiliense, Santa Lúcia`}. In the Figure 4b, first it is executed kNN selection with s_{q1} = "São Carlos" in the relation `CitySaoCarlos` and, then, kNN selection with s_{q2} = "Araraquara". The answer of last kNN selection is {`São Carlos, Analândia, Itirapina, Ribeirão Bonito, Ibaté`}. This proof validates the Property 9. □

This property is valid if query elements s_{q1} and s_{q2} are distinct. For the same query elements, there exists a special case where kNN selection operation becomes commutative under its composition: if $s_{q1} = s_{q2} = s_q$, the Expression 24 can be rewritten as a conjunction of kNN conditions, then only the kNN selection with the smallest k condition needs to be executed [13]. That is,

Property 9.1. *Special case where* $s_{q1} = s_{q2} = s_q$.

$$\ddot{\sigma}_{\left(S\ \ddot{\theta}(d,k_1)\ s_q\right)}\left(\ddot{\sigma}_{\left(S\ \ddot{\theta}(d,k_2)\ s_q\right)} T\right) = \ddot{\sigma}_{\left(S\ \ddot{\theta}(d,k_2)\ s_q\right)}\left(\ddot{\sigma}_{\left(S\ \ddot{\theta}(d,k_1)\ s_q\right)} T\right) = \ddot{\sigma}_{\left(S\ \ddot{\theta}(d,k_1)\ s_q\right) \wedge \left(S\ \ddot{\theta}(d,k_2)\ s_q\right)} T\ =\ \ddot{\sigma}_{\left(S\ \ddot{\theta}(d,\min(k_1,k_2))\ s_q\right)} T. \tag{25}$$

Property 10. *The kNN selection operation and traditional selection operation do not commute under their composition, i.e.,*

$$\ddot{\sigma}_{\left(S\ \ddot{\theta}(d,k)\ s_q\right)}\left(\sigma_{(A\ \theta\ a)} T\right) \neq \sigma_{(A\ \theta\ a)}\left(\ddot{\sigma}_{\left(S\ \ddot{\theta}(d,k)\ s_q\right)} T\right). \tag{26}$$

Proof. We use the following counterexample to prove this property.

Example 3. *"Select 5 nearest cities of "São Carlos-SP", which latitude* $= -22.02$ *and longitude* $= 47.89$*, and their latitude is smaller than the latitude of "São Carlos-SP", considering Euclidean distance* L_2*".*

$$\ddot{\sigma}_{\left(\texttt{coordenada}\ \ddot{\theta}(L_2,5)\ (-22.02,47.89)\right)}\left(\sigma_{(\texttt{lat}<-22.02)}\ \texttt{CitySaoCarlos}\right) = \sigma_{(\texttt{lat}<-22.02)}\left(\ddot{\sigma}_{\left(\texttt{coordenada}\ \ddot{\theta}(L_2,5)\ (-22.02,47.89)\right)}\ \texttt{CitySaoCarlos}\right).$$

However, when algebraic expressions are executed, the results are not the same. This can be observed in the Figure 5.

In the 5a, first it is executed kNN selection in the relation `CitySaoCarlos` and, then, traditional selection. The answer of this query is {`Itirapina, Analândia, Ribeirão Bonito`}. In the Figure 5b, it is first executed traditional selection in the relation `CitySaoCarlos` and, then, kNN selection. The answer of last selection is {`Corumbataí, Analândia, Ribeirão Bonito, Itirapina, Brotas`}. This proof validates the Property 10. □

Property 11. *The kNN selection operation and range selection operation do not commute under their composition, i.e.,*

$$\ddot{\sigma}_{\left(S_1\ \ddot{\theta}(d_1,k)\ s_{q1}\right)}\left(\hat{\sigma}_{\left(S_2\ \hat{\theta}(d_2,\xi)\ s_{q2}\right)} T\right) \neq\ \hat{\sigma}_{\left(S_2\ \hat{\theta}(d_2,\xi)\ s_{q2}\right)}\left(\ddot{\sigma}_{\left(S_1\ \ddot{\theta}(d_1,k)\ s_{q1}\right)} T\right). \tag{27}$$

Proof. We use the following counterexample to prove this property.

Example 4. *"Select 5 nearest cities of São Carlos-SP, which latitude* $= -22.02$ *and longitude* $= 47.89$*, and their distance from "Araraquara", which latitude* $= -21.79$ *and longitude* $= 48.18$*, are not farther then* 0.3 *distance units, considering Euclidean distance* L_2*".*

$$\ddot{\sigma}_{\left(\texttt{coordenada}\ \ddot{\theta}(L_2,5)\ (-22.02,47.89)\right)}\left(\hat{\sigma}_{\left(\texttt{coordenada}\ \hat{\theta}(L_2,0.3)\ (-21.79,48.18)\right)}\texttt{CitySaoCarlos}\right) = \hat{\sigma}_{\left(\texttt{coordenada}\ \hat{\theta}(L_2,0.3)\ (-21.79,48.18)\right)}\left(\ddot{\sigma}_{\left(\texttt{coordenada}\ \ddot{\theta}(L_2,5)\ (-22.02,47.89)\right)}\texttt{CitySaoCarlos}\right).$$

Algebraic expressions are executed and their results are not the same. This can be observed in the Figure 6.

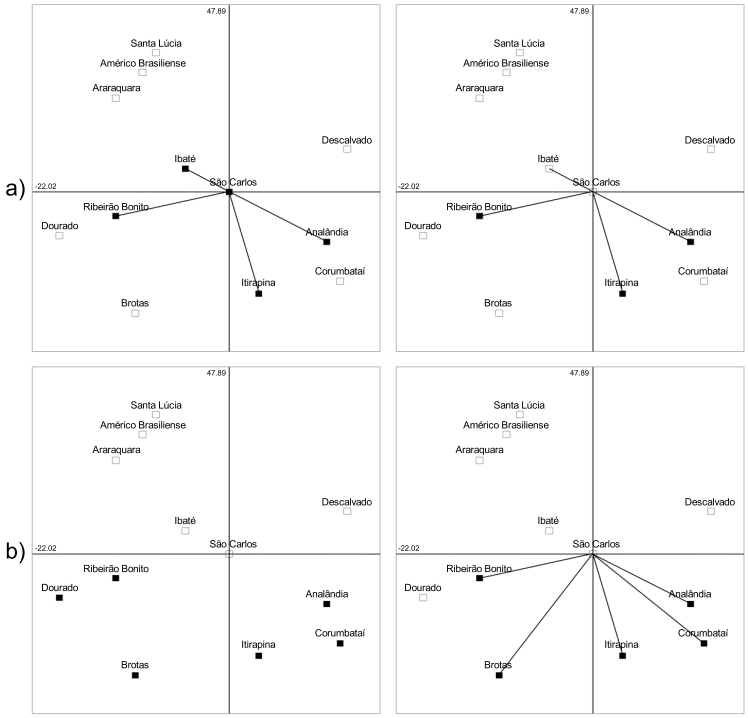

Fig. 5. Non-commutativity of kNN selection operation with traditional selection operation. Query presents in Example 3: a) Query first executed kNN selection and, then, traditional selection in the relation `CitySaoCarlos`. b) Query executed first traditional selection and, then, kNN selection in the relation `CitySaoCarlos`.

In the Figure 6a, first it is executed *range* selection with $s_{q2}=$ "Araraquara" in the relation `CitySaoCarlos` and, then, kNN selection with $s_{q1}=$ "São Carlos". The answer of this query is {`Ribeirão Bonito, Ibaté, Araraquara, Américo Brasiliense, Santa Lúcia`}. In the Figure 6b, it is first executed kNN selection with $s_{q1}=$ "São Carlos" in the relation `CitySaoCarlos` and, then, *range* selection with $s_{q2}=$ "Araraquara" over the result of first selection. The answer of last selection is {`Ribeirão Bonito, Ibaté`}. This proof validates the Property 11. □

This property is valid if query elements s_{q1} and s_{q2} are distinct. For the same query elements, there exists a special case where kNN selection operation becomes commutative with range selection operation: if $s_{q1} = s_{q2} = s_q$, the conjunctions of range and kNN conditions can be rewritten as a intersection of the results from both basic operators, as follow [13]. This is also equivalent to the execution of the algorithm $kAndRange(\theta, s_q, k, \xi)$.

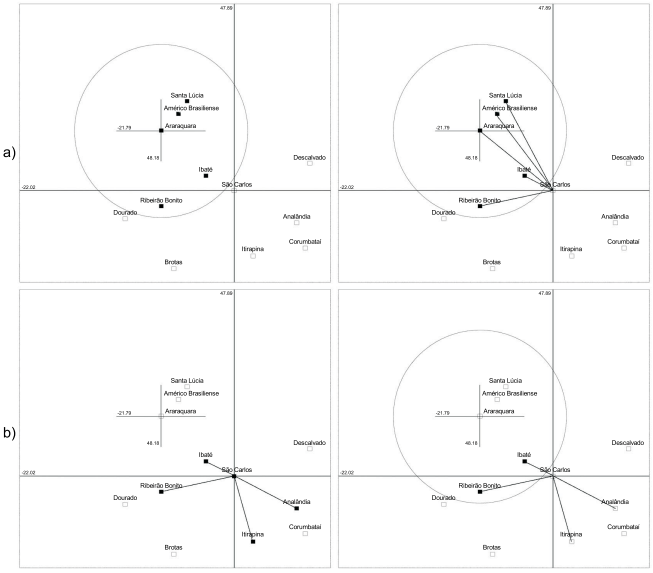

Fig. 6. Non-commutativity of kNN selection operation with range selection operation (query elements distinct). Query presents in Example 4: a) Query first executed *range* selection with $s_{q2} =$ "Araraquara" and, then, kNN selection with $s_{q1} =$ "São Carlos" in the relation `CitySaoCarlos`. b) Query executed first kNN selection with $s_{q1} =$ "São Carlos" and, then, *range* selection with $s_{q2} =$ "Araraquara" in the relation `CitySaoCarlos`.

Definition 11.1. *Special case where* $s_{q1} = s_{q2} = s_q$.

$$\hat{\sigma}_{\left(S\ \hat{\theta}(d,\xi)\ s_q\right)}\left(\ddot{\sigma}_{\left(S\ \ddot{\theta}(d,k)\ s_q\right)}T\right) = \ddot{\sigma}_{\left(S\ \hat{\theta}(d,\xi)\ s_q\right)\wedge\left(S\ \ddot{\theta}(d,k)\ s_q\right)}\ T\ =$$
$$\left(\hat{\sigma}_{\left(S\ \hat{\theta}(d,\xi)\ s_q\right)}T\right)\cap\left(\ddot{\sigma}_{\left(S\ \ddot{\theta}(d,k)\ s_q\right)}T\right)\ =\ kAndRange(\theta, s_q, k, \xi); \tag{28}$$

and the disjunction of range and kNN conditions can be rewritten as a union of the results from both basic operators, as follow [13]. This is also equivalent to the execution of the algorithm $kOrRange(\theta, s_q, k, \xi)$.

Definition 11.2. *Special case where* $s_{q1} = s_{q2} = s_q$.

$$\hat{\sigma}_{\left(S\ \hat{\theta}(d,\xi)\ s_q\right)}\left(T\right)\ \cup\ \ddot{\sigma}_{\left(S\ \ddot{\theta}(d,k)\ s_q\right)}\left(T\right) \Leftrightarrow kOrRange\left(\theta, s_q, k, \xi\right)\ \ ; \tag{29}$$

where θ denotes if it is a range and kNN condition or range^{-1} and kFN condition. This special case is important in the similarity query optimization because it allows reading the dataset only once, while the intersection expression requires reading the dataset twice to answer the query. Then, the I/O and CPU costs can be reduced.

In summary, k-NN-based selections do not commute with any other selection conditions including other kNN conditions. Therefore, distinct execution orders of kNN

conditions result in different answers. Moreover, for this kind of query, it is necessary to execute each selection condition separately, and then return the intersection or the union of results to answer the query. Algebraically, non-commutativity of kNN selection operation can be expressed in two ways depending on how the conditions are connected. Comparing commutativity equivalences of kNN selections with those presented by traditional selections, the following properties occurs, regarding conjunctive conditions ($\wedge$):

$$\begin{array}{ccc}
\ddot{\sigma}_{\left(S_1\ \ddot{\theta}(d_1,k_1)\ s_{q1}\right)\wedge\left(S_2\ \ddot{\theta}(d_2,k_2)\ s_{q2}\right)}T & \overset{1}{\Leftrightarrow} & \left(\ddot{\sigma}_{\left(S_1\ \ddot{\theta}(d_1,k_1)\ s_{q1}\right)}T\right)\cap\left(\ddot{\sigma}_{\left(S_2\ \ddot{\theta}(d_2,k_2)\ s_{q2}\right)}T\right) \overset{2}{\Leftrightarrow}\dots \\
\Updownarrow 3 & & \Updownarrow 4 \\
\ddot{\sigma}_{\left(S_2\ \ddot{\theta}(d_2,k_2)\ s_{q2}\right)\wedge\left(S_1\ \ddot{\theta}(d_1,k_1)\ s_{q1}\right)}T & \overset{6}{\Leftrightarrow} & \left(\ddot{\sigma}_{\left(S_2\ \ddot{\theta}(d_2,k_2)\ s_{q2}\right)}T\right)\cap\left(\ddot{\sigma}_{\left(S_1\ \ddot{\theta}(d_1,k_1)\ s_{q1}\right)}T\right) \overset{7}{\Leftrightarrow}\dots
\end{array}$$

$$\begin{array}{c}
\dots \overset{2}{\Leftrightarrow} \ddot{\sigma}_{\left(S_1\ \ddot{\theta}(d_1,k_1)\ s_{q1}\right)}\left(\ddot{\sigma}_{\left(S_2\ \ddot{\theta}(d_2,k_2)\ s_{q2}\right)}T\right) \\
\Updownarrow 3 \\
\dots \overset{7}{\Leftrightarrow} \ddot{\sigma}_{\left(S_2\ \ddot{\theta}(d_2,k_2)\ s_{q2}\right)}\left(\ddot{\sigma}_{\left(S_1\ \ddot{\theta}(d_1,k_1)\ s_{q1}\right)}T\right),
\end{array}$$

and regarding disjunctive conditions ($\vee$):

$$\begin{array}{ccc}
\ddot{\sigma}_{\left(S_1\ \ddot{\theta}(d_1,k_1)\ s_{q1}\right)\vee\left(S_2\ \ddot{\theta}(d_2,k_2)\ s_{q2}\right)}T & \overset{1}{\Leftrightarrow} & \left(\ddot{\sigma}_{\left(S_1\ \ddot{\theta}(d_1,k_1)\ s_{q1}\right)}T\right)\cup\left(\ddot{\sigma}_{\left(S_2\ \ddot{\theta}(d_2,k_2)\ s_{q2}\right)}T\right) \\
\Updownarrow 2 & & \Updownarrow 3 \\
\ddot{\sigma}_{\left(S_2\ \ddot{\theta}(d_2,k_2)\ s_{q2}\right)\vee\left(S_1\ \ddot{\theta}(d_1,k_1)\ s_{q1}\right)}T & \overset{4}{\Leftrightarrow} & \left(\ddot{\sigma}_{\left(S_2\ \ddot{\theta}(d_2,k_2)\ s_{q2}\right)}T\right)\cup\left(\ddot{\sigma}_{\left(S_1\ \ddot{\theta}(d_1,k_1)\ s_{q1}\right)}T\right).
\end{array}$$

Thus, Property 12 holds.

Property 12. *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:*

$$\begin{array}{l}
\left(\ddot{\sigma}_{\left(S_1\ \ddot{\theta}(d_1,k_1)\ s_{q1}\right)}T\right)\cap\left(\ddot{\sigma}_{\left(S_2\ \ddot{\theta}(d_2,k_2)\ s_{q2}\right)}T\right) = \\
\left(\ddot{\sigma}_{\left(S_2\ \ddot{\theta}(d_2,k_2)\ s_{q2}\right)}T\right)\cap\left(\ddot{\sigma}_{\left(S_1\ \ddot{\theta}(d_1,k_1)\ s_{q1}\right)}T\right);
\end{array} \tag{30}$$

and for disjunctive conditions:

$$\begin{array}{l}
\left(\ddot{\sigma}_{\left(S_1\ \ddot{\theta}(d_1,k_1)\ s_{q1}\right)}T\right)\cup\left(\ddot{\sigma}_{\left(S_2\ \ddot{\theta}(d_2,k_2)\ s_{q2}\right)}T\right) = \\
\left(\ddot{\sigma}_{\left(S_2\ \ddot{\theta}(d_2,k_2)\ s_{q2}\right)}T\right)\cup\left(\ddot{\sigma}_{\left(S_1\ \ddot{\theta}(d_1,k_1)\ s_{q1}\right)}T\right).
\end{array} \tag{31}$$

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

Proof. We prove this property using the Definition 3:

$$
\begin{aligned}
&\left(\ddot{\sigma}_{\left(S_1\ \ddot{\theta}(d_1,k_1)\ s_{q1}\right)}T\right) \cap \left(\ddot{\sigma}_{\left(S_2\ \ddot{\theta}(d_2,k_2)\ s_{q2}\right)}T\right) = \\
&\{t_i \in T |\ \forall\, t_{i1} \in [T - T_1], T_1 = t_{i=1,\ldots,k_1}, d\left(t_i\left(S\right), s_{q1}\right) \leq d\left(t_{i1}\left(S\right), s_{q1}\right)\} \cap \\
&\{t_i \in T |\ \forall\, t_{i2} \in [T - T_2], T_2 = t_{i=1,\ldots,k_2}, d\left(t_i\left(S\right), s_{q2}\right) \leq d\left(t_{i2}\left(S\right), s_{q2}\right)\} = \\
&\{t_i \in T |\ \forall\, t_{i1} \in [T - T_1], T_1 = t_{i=1,\ldots,k_1}, d\left(t_i\left(S\right), s_{q1}\right) \leq d\left(t_{i1}\left(S\right), s_{q1}\right) \wedge \\
&\qquad \forall\, t_{i2} \in [T - T_2], T_2 = t_{i=1,\ldots,k_2}, d\left(t_i\left(S\right), s_{q2}\right) \leq d\left(t_{i2}\left(S\right), s_{q2}\right)\} = \\
&\{t_i \in T |\ \forall\, t_{i2} \in [T - T_2], T_2 = t_{i=1,\ldots,k_2}, d\left(t_i\left(S\right), s_{q2}\right) \leq d\left(t_{i2}\left(S\right), s_{q2}\right) \wedge \\
&\qquad \forall\, t_{i1} \in [T - T_1], T_1 = t_{i=1,\ldots,k_1}, d\left(t_i\left(S\right), s_{q1}\right) \leq d\left(t_{i1}\left(S\right), s_{q1}\right)\} = \\
&\{t_i \in T \mid \forall\, t_{i2} \in [T - T_2], T_2 = t_{i=1,\ldots,k_2}, d\left(t_i\left(S\right), s_{q2}\right) \leq d\left(t_{i2}\left(S\right), s_{q2}\right)\} \cap \\
&\{t_i \in T \mid \forall\, t_{i1} \in [T - T_1], T_1 = t_{i=1,\ldots,k_1}, d\left(t_i\left(S\right), s_{q1}\right) \leq d\left(t_{i1}\left(S\right), s_{q1}\right)\} = \\
&\left(\ddot{\sigma}_{\left(S_2\ \ddot{\theta}(d_2,k_2)\ s_{q2}\right)}T\right) \cap \left(\ddot{\sigma}_{\left(S_1\ \ddot{\theta}(d_1,k_1)\ s_{q1}\right)}T\right).
\end{aligned}
$$

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

The next set of properties involves traditional binary operators. In relational algebra, it is possible to distribute σ over binary operators in different relations of the expression. Therefore, the following expression is valid for union ($\cup$) operator since the relations T_1 e T_2 are union compatible:

$$
\sigma_{(A\ \theta\ a)}\left(T_1\ \cup\ T_2\right) = \left(\sigma_{(A\ \theta\ a)}T_1\right) \cup \left(\sigma_{(A\ \theta\ a)}T_2\right). \tag{32}
$$

Though, if σ is changed by $\ddot{\sigma}$, the Expression 32 becomes invalid, generating the Property 13.

Property 13. *The $\ddot{\sigma}$ is not distributed over union ($\cup$).*

$$
\ddot{\sigma}_{\left(S\ \ddot{\theta}(d,k)\ s_q\right)}\left(T_1\ \cup\ T_2\right) \neq \left(\ddot{\sigma}_{\left(S\ \ddot{\theta}(d,k)\ s_q\right)}T_1\right) \cup \left(\ddot{\sigma}_{\left(S\ \ddot{\theta}(d,k)\ s_q\right)}T_2\right). \tag{33}
$$

Proof. We use the following counterexample to prove this property.

Example 5. *"Select 5 nearest cities of São Carlos-SP, which latitude* $= -22.02$ *and longitude* $= 47.89$*, that belong to relation "CitySaoCarlos" or relation "CityAraraquara", considering Euclidean distance* L_2*".*

$$
\begin{aligned}
&\ddot{\sigma}_{\left(\texttt{coordenada}\ \ddot{\theta}(L_2,5)\ (-22.02,47.89)\right)}\left(\texttt{CityAraraquara} \cup \texttt{CitySaoCarlos}\right) = \\
&\left(\ddot{\sigma}_{\left(\texttt{coordenada}\ \ddot{\theta}(L_2,5)\ (-22.02,47.89)\right)}\texttt{CityAraraquara}\right) \cup \\
&\left(\ddot{\sigma}_{\left(\texttt{coordenada}\ \ddot{\theta}(L_2,5)\ (-22.02,47.89)\right)}\texttt{CitySaoCarlos}\right).
\end{aligned} \tag{34}
$$

Figure 7 shows that the results of two algebraic expressions are not the same.

In the Figure 7a, first it is executed the union of relations `CitySaoCarlos` and `CityAraraquara` and, then, kNN selection. The answer of this query is {`São Carlos, Ibaté, Itirapina, Analândia, Ribeirão Bonito`}. In the Figure 7b, first it is executed kNN selection in the relations `CitySaoCarlos` and `CityAraraquara` and, then, the union of the results. The answer of last query is {`São Carlos, Analândia, Ribeirão Bonito, Ibaté, Itirapina, Araraquara, Américo Brasiliense`}. This proof validates the Property 13. □

In the difference $(-)$, σ can be distributed to the first relation and, optionally, to both relations of the expression. This is present in the following expression. The relations T_1 e T_2 must be union compatible.

$$\sigma_{(A\,\theta\,a)}\left(T_1\ -\ T_2\right) = \left(\sigma_{(A\,\theta\,a)}T_1\right) -\ T_2 = \left(\sigma_{(A\,\theta\,a)}T_1\right) - \left(\sigma_{(A\,\theta\,a)}T_2\right). \tag{35}$$

However, if σ is altered by $\ddot{\sigma}$, Expression 35 becomes invalid and the Property 14 is generated.

Property 14. *The $\ddot{\sigma}$ is not distributed over difference $(-)$, i.e., the expressions*

$$\ddot{\sigma}_{\left(S\,\ddot{\theta}(d,k)\,s_q\right)}\left(T_1\ -\ T_2\right), \tag{36}$$

$$\left(\ddot{\sigma}_{\left(S\,\ddot{\theta}(d,k)\,s_q\right)}T_1\right) -\ T_2, \tag{37}$$

$$\left(\ddot{\sigma}_{\left(S\,\ddot{\theta}(d,k)\,s_q\right)}T_1\right) - \left(\ddot{\sigma}_{\left(S\,\ddot{\theta}(d,k)\,s_q\right)}T_2\right) \tag{38}$$

are distinct pairwise.

Proof. We use the following counterexample to prove this property.

Example 6. *"Select 5 nearest cities of São Carlos-SP, which latitude $= -22.02$ and longitude $=47.89$, that belong to relation "CityAraraquara" but do not belong to relation "CitySaoCarlos", considering Euclidean distance L_2".*

$$\begin{aligned}
&\ddot{\sigma}_{\left(\texttt{coordenada}\,\ddot{\theta}(L_2,5)\,(-22.02,47.89)\right)}\left(\texttt{CityAraraquara}\ -\ \texttt{CitySaoCarlos}\right) = \\
&\left(\ddot{\sigma}_{\left(\texttt{coordenada}\,\ddot{\theta}(L_2,5)\,(-22.02,47.89)\right)}\texttt{CityAraraquara}\right) - \texttt{CitySaoCarlos} = \\
&\left(\ddot{\sigma}_{\left(\texttt{coordenada}\,\ddot{\theta}(L_2,5)\,(-22.02,47.89)\right)}\texttt{CityAraraquara}\right) - \\
&\left(\ddot{\sigma}_{\left(\texttt{coordenada}\,\ddot{\theta}(L_2,5)\,(-22.02,47.89)\right)}\texttt{CitySaoCarlos}\right).
\end{aligned} \tag{39}$$

The results of algebraic expression are not the same. This can be observed in the Figure 8.

In the Figure 8a, first it is executed the difference between the relations `CityAraraquara` and `CitySaoCarlos` and, then, kNN selection. The answer of this query is {`Boa Esperança do Sul, Guatapará, Motuca, Rincão, Trabiju`}. In the Figure 8b, first it is executed kNN selection in the relation `CityAraraquara` and, then, the difference of its result with the relation `CitySaoCarlos` is realized. The answer of

this is empty ($\varnothing$), i.e., all tuples returned in the kNN selection belong to the relation `CitySaoCarlos`. In the Figure 8c, first it is executed kNN selection in the relations `CityAraraquara` and `CitySaoCarlos` and, then, the difference of their results. The answer of last query is {`Américo Brasiliense, Araraquara`}. This proof validates the Property 14. □

In the intersection ($\cap$), traditional selection can be distributed over to the first relation, to the second relation or, optionally, to both relations of the expression, depending on the attributes mentioned in condition. Then, Expression 40 shows this property. The relations T_1 e T_2 must be union compatible.

$$\begin{aligned}\sigma_{(A\,\theta\,a)}\left(T_1 \cap T_2\right) &= \underbrace{\left(\sigma_{(A\,\theta\,a)}T_1\right) \cap T_2}_{1} = \underbrace{T_1 \cap \left(\sigma_{(A\,\theta\,a)}T_2\right)}_{2} \\ &= \left(\sigma_{(A\,\theta\,a)}T_1\right) \cap \left(\sigma_{(A\,\theta\,a)}T_2\right). \end{aligned} \tag{40}$$

If all attributes mentioned in the condition belong to T_1 then the Equivalence 1 of Expression 40 is valid; but, if all attributes mentioned in the condition belong to T_2 then the Equivalence 2 is valid.

Nevertheless, if σ is switched to $\ddot{\sigma}$, this expression get invalid and the Property 15 is generated.

Property 15. *The $\ddot{\sigma}$ is not distributed over intersection ($\cap$), that is, the expressions*

$$\ddot{\sigma}_{\left(S\,\ddot{\theta}(d,k)\,s_q\right)}\left(T_1 \cap T_2\right), \tag{41}$$

$$\left(\ddot{\sigma}_{\left(S\,\ddot{\theta}(d,k)\,s_q\right)}T_1\right) \cap T_2, \tag{42}$$

$$T_1 \cap \left(\ddot{\sigma}_{\left(S\,\ddot{\theta}(d,k)\,s_q\right)}T_2\right), \tag{43}$$

$$\left(\ddot{\sigma}_{\left(S\,\ddot{\theta}(d,k)\,s_q\right)}T_1\right) \cap \left(\ddot{\sigma}_{\left(S\,\ddot{\theta}(d,k)\,s_q\right)}T_2\right), \tag{44}$$

are distinct pairwise.

Proof. We use the following counterexample to prove this property.

Example 7. *"Select 5 nearest cities of São Carlos-SP, which latitude $= -22.02$ and longitude $= 47.89$, that belong to the relation "CityAraraquara" and the relation "CitySaoCarlos", considering Euclidean distance L_2".*

$$\begin{aligned}&\ddot{\sigma}_{\left(\texttt{coordenada}\;\ddot{\theta}(L_2,5)\;(-22.02,47.89)\right)}\left(\texttt{CityAraraquara} \cap \texttt{CitySaoCarlos}\right) = \\ &\left(\ddot{\sigma}_{\left(\texttt{coordenada}\;\ddot{\theta}(L_2,5)\;(-22.02,47.89)\right)}\texttt{CityAraraquara}\right) \cap \\ &\left(\ddot{\sigma}_{\left(\texttt{coordenada}\;\ddot{\theta}(L_2,5)\;(-22.02,47.89)\right)}\texttt{CitySaoCarlos}\right). \end{aligned} \tag{45}$$

The results of algebraic expression are presented in the Figure 9.

In the Figure 9a, first it is executed the intersection between the relations `CityAraraquara` and `CitySaoCarlos` and, then, kNN selection. The answer of this query is {`São Carlos, Ribeirão Bonito, Ibaté, Araraquara, Américo Brasiliense`}. In the Figure 9b, first it is executed kNN selection in the relations `CityAraraquara` and `CitySaoCarlos` and, then, the intersection of their results. The answer of last query is {`Ribeirão Bonito, São Carlos, Ibaté`}. This proof validates the Property 15. □

For join ($\bowtie$) and cross product ($\times$) binary operators, σ should be distributed to a relation that has all the attributes mentioned in the condition, if one exists. Traditional selection can only be distributed to a relation if all attributes in the condition belong to this relation. That is, the following expression is valid for cross product binary operator, in relational algebra.

$$\sigma_{(A\ \theta\ a)}\left(T_1\ \times\ T_2\right) = \underbrace{\left(\sigma_{(A\ \theta\ a)}T_1\right)\ \times\ T_2}_{1} = \underbrace{T_1\ \times\left(\sigma_{(A\ \theta\ a)}T_2\right)}_{2}. \tag{46}$$

If all attributes mentioned in the condition belong to T_1 then the Equivalence 1 of Expression 46 is valid; but, if all attributes mentioned in the condition belong to T_2 then the Equivalence 2 is valid.

Nonetheless, if σ is substituted by $\ddot{\sigma}$, Expression 46 becomes invalid generating the Property 16.

Property 16. *The $\ddot{\sigma}$ is not distributed over cross product ($\times$), i.e., the expressions*

$$\ddot{\sigma}_{\left(S\ \ddot{\theta}(d,k)\ s_q\right)}\left(T_1\ \times\ T_2\right), \tag{47}$$

$$\left(\ddot{\sigma}_{\left(S\ \ddot{\theta}(d,k)\ s_q\right)}T_1\right)\ \times\ T_2, \tag{48}$$

$$T_1\ \times\left(\ddot{\sigma}_{\left(S\ \ddot{\theta}(d,k)\ s_q\right)}T_2\right) \tag{49}$$

are distinct pairwise.

Proof. We use the following counterexample to prove this property.

Example 8. *"Select 5 pairs of nearest cities of São Carlos-SP, which latitude $=-22.02$ and longitude $=47.89$, belong to the relations "CityAraraquara" and "CitySaoCarlos", considering Euclidean distance L_2".*

$$\underbrace{\ddot{\sigma}_{\left(\texttt{coordenada}\ \ddot{\theta}(L_2,5)\ (-22.02,47.89)\right)}\left(\texttt{CityAraraquara}\ \times\ \texttt{CitySaoCarlos}\right)}_{1} =$$

$$\underbrace{\left(\ddot{\sigma}_{\left(\texttt{coordenada}\ \ddot{\theta}(L_2,5)\ (-22.02,47.89)\right)}\texttt{CityAraraquara}\right)\ \times\ \texttt{CitySaoCarlos}}_{2} =$$

$$\underbrace{\texttt{CityAraraquara}\ \times\left(\ddot{\sigma}_{\left(\texttt{coordenada}\ \ddot{\theta}(L_2,5)\ (-22.02,47.89)\right)}\texttt{CitySaoCarlos}\right)}_{3}. \tag{50}$$

Expression 1 returns 204 tuples ($|17| * |12|$); Expression 2 returns 60 tuples ($|5| * |12|$); and Expression 3 returns 85 tuples ($|17| * |5|$). This proof validates the Property 16. □

In relational algebra, if $\times$ is replaced by $\bowtie$, the Expression 46 continues valid. It is shown in the following expression.

$$\sigma_{(A\,\theta\,a)}\left(T_1 \bowtie T_2\right) = \underbrace{\left(\sigma_{(A\theta\,a)}T_1\right) \bowtie T_2}_{1} = \underbrace{T_1 \bowtie \left(\sigma_{(A\,\theta\,a)}T_2\right)}_{2}. \tag{51}$$

If all attributes mentioned in the condition belong to T_1 then the Equivalence 1 of Expression 51 is valid; but, if all attributes mentioned in the condition belong to T_2 then the Equivalence 2 is valid.

However, if σ is exchanged by $\ddot{\sigma}$ in the Expression 51, this expression becomes invalid and the Property 17 is generated.

Property 17. *The $\ddot{\sigma}$ is not distributed over join ($\bowtie$), that is, the expressions*

$$\ddot{\sigma}_{\left(S\,\ddot{\theta}(d,k)\,s_q\right)}\left(T_1 \bowtie T_2\right), \tag{52}$$

$$\left(\ddot{\sigma}_{\left(S\,\ddot{\theta}(d,k)\,s_q\right)}T_1\right) \bowtie T_2, \tag{53}$$

$$T_1 \bowtie \left(\ddot{\sigma}_{\left(S\,\ddot{\theta}(d,k)\,s_q\right)}T_2\right) \tag{54}$$

are distinct pairwise.

Proof. We use the following counterexample to prove this property.

Example 9. *"Select 5 nearest cities of São Carlos-SP, which latitude $= -22.02$ and longitude $= 47.89$, belong to the relations "CityAraraquara" and "CitySaoCarlos", considering Euclidean distance L_2".*

$$\underbrace{\ddot{\sigma}_{\left(\texttt{coordenada}\;\ddot{\theta}(L_2,5)\;(-22.02,47.89)\right)}\left(\texttt{CityAraraquara} \bowtie \texttt{CitySaoCarlos}\right)}_{1} =$$

$$\underbrace{\texttt{CityAraraquara} \bowtie \left(\ddot{\sigma}_{\left(\texttt{coordenada}\;\ddot{\theta}(L_2,5)\;(-22.02,47.89)\right)}\texttt{CitySaoCarlos}\right)}_{2}. \tag{55}$$

Expression 1 returns 5 tuples: {São Carlos, Ribeirão Bonito, Ibaté, Araraquara, Américo Brasiliense}; and Expression 2 returns 3 tuples: {São Carlos, Ribeirão Bonito, Ibaté}. This proof validates the Property 17. □

Alternatively, if the condition is conjunctive and can be written as $A_1\theta_1 a_1 \wedge A_2\theta_2 a_2$, where condition $A_1\,\theta_1\,a_1$ involves only attributes of T_1 and condition $A_2\,\theta_2\,a_2$ involves only attributes of T_2, the following expressions are valid.

$$\sigma_{(A_1\,\theta_1\,a_1\,\wedge\,A_2\,\theta_2\,a_2)}\left(T_1\,\theta\,T_2\right) = \left(\sigma_{(A_1\,\theta_1\,a_1)}T_1\right)\;\theta\;\left(\sigma_{(A_2\,\theta_2\,a_2)}T_2\right), \tag{56}$$

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

Notwithstanding, if σ is modified by $\ddot{\sigma}$, Expression 56 becomes invalid.

$$\ddot{\sigma}_{\left(S_1\,\ddot{\theta}(d_1,k_1)\,s_{q1}\right)\,\wedge\,\left(S_2\,\ddot{\theta}(d_2,k_2)\,s_{q2}\right)}\left(T_1\,\theta\,T_2\right) \neq$$
$$\left(\ddot{\sigma}_{\left(S_1\,\ddot{\theta}(d_1,k_1)\,s_{q1}\right)}T_1\right)\,\theta\,\left(\ddot{\sigma}_{\left(S_2\,\ddot{\theta}(d_2,k_2)\,s_{q2}\right)}T_2\right), \tag{57}$$

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

Proof. This inequality is proved using the properties 7, 16 and 17. □

Then, Inequality 57 completes the Property 17. Therefore, 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 accept only three properties (7, 8 and 12) and five special cases (over the same query elements), 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 [13] are examples specifically created to handle the commutativity property of range query and kNN query over the same query element.

Other conclusion is that there are not properties applied with kNN selection operation using traditional binary operators. Therefore, when kNN selection operation involves traditional binary operators, optimization can not be applied to $\ddot{\sigma}$.

Tables 1 and 2 show a summary of algebraic properties for range / reversed range selection ($\hat{\sigma}$) and k-Nearest / k-Farthest Neighbor selection ($\ddot{\sigma}$) to traditional unary and traditional binary operators, respectively. In Table 1, for traditional selection (σ), $c_1 = (A_1\, \theta_1\, a_1)$, $c_2 = (A_2\, \theta_2\, a_2)$ and $\theta =$ '$<$', '$\leq$', '$>$', '$\geq$', '$=$' or '$\neq$'; and for similarity selection ($\hat{\sigma}$ or $\ddot{\sigma}$), $c_1 = (S_1\, \theta_{c1}(d_1, \lim_1)\, s_{q1})$ and $c_2 = (S_2\, \theta_{c2}(d_2, \lim_2)\, s_{q2})$. In the case of $\hat{\sigma}$, $\theta_c =$ '$\hat{\theta}$' or '$\hat{\theta}^{-1}$', $\lim_1 = \xi_1$ and $\lim_2 = \xi_2$; and in the case of $\ddot{\sigma}$, $\theta_c =$ '$\ddot{\theta}$' or '$\ddot{\theta}_F$', $\lim_1 = k_1$ and $\lim_2 = k_2$. In Table 2, for traditional selection (σ), $c = (A\, \theta\, a)$ and $\theta =$ '$<$', '$\leq$', '$>$', '$\geq$', '$=$' or '$\neq$'; for similarity selection ($\hat{\sigma}$ or $\ddot{\sigma}$), $c = (S\, \theta_c(d, \lim)\, s_q)$. In the case of $\hat{\sigma}$, $\theta_c =$ '$\hat{\theta}$' or '$\hat{\theta}^{-1}$' and $\lim = \xi$; and in the case of $\ddot{\sigma}$, $\theta_c =$ '$\ddot{\theta}$' or '$\ddot{\theta}_F$' and $\lim = k$.

Table 1. Traditional unary operators - summary of the algebraic properties of σ, $\hat{\sigma}$ and $\ddot{\sigma}$.

Unary operators				
Properties σ	$\hat{\sigma}$	$\ddot{\sigma}$	$s_{q1} = s_{q2}$	**Exception** ($s_{q1} = s_{q2}$)
$\sigma_{(c_1 \wedge c_2)}T = (\sigma_{c_1}T) \cap (\sigma_{c_2}T) = \sigma_{c_1}(\sigma_{c_2}T)$	✔	✔	✔	$\hat{\sigma}/\ddot{\sigma}_{(min(c_1,c_2))}T$
$\sigma_{(c_1 \vee c_2)}T = (\sigma_{c_1}T) \cup (\sigma_{c_2}T)$	✔	✔	✔	$\hat{\sigma}/\ddot{\sigma}_{(max(c_1,c_2))}T$
$\sigma_{c_1}(\sigma_{c_2}T) = \sigma_{c_2}(\sigma_{c_1}T)$	✔	✗	✔	$\ddot{\sigma}_{(min(c_1,c_2))}T$
Commutativity σ	✔	✗	✗	–
Commutativity $\hat{\sigma}$	✔	✗	✔	$(\hat{\sigma}_{c_1}T) \cap (\ddot{\sigma}_{c_2}T)/(\hat{\sigma}_{c_1}T) \cup (\ddot{\sigma}_{c_2}T)$
Commutativity $\ddot{\sigma}$	✗	✗	✔	$\ddot{\sigma}_{c_1}(\ddot{\sigma}_{c_2}T) = \ddot{\sigma}_{c_2}(\ddot{\sigma}_{c_1}T)$

4 Conclusion

Nowadays, storing and retrieving multimedia data is a requirement that must be provided by RDBMS, so the challenge of enabling them has drawn researcher's attentions.

Table 2. Traditional binary operators - summary of the algebraic properties of σ, $\hat{\sigma}$ and $\ddot{\sigma}$.

Binary operators		
Properties σ	$\hat{\sigma}$	$\ddot{\sigma}$
$\sigma_c\,(T_1 \cup T_2) = (\sigma_c\,T_1) \cup (\sigma_c\,T_2)$	✔	✗
$\sigma_c\,(T_1 - T_2) = (\sigma_c\,T_1) - T_2 = (\sigma_c\,T_1) - (\sigma_c\,T_2)$	✔	✗
$\sigma_c\,(T_1 \cap T_2) = (\sigma_c\,T_1) \cap T_2 = (\sigma_c\,T_1) \cap (\sigma_c\,T_2) = T_1 \cap (\sigma_c\,T_2)$	✔	✗
$\sigma_c\,(T_1 \times T_2) = (\sigma_c\,T_1) \times T_2$	✔	✗
$\sigma_c\,(T_1 \bowtie T_2) = (\sigma_c\,T_1) \bowtie T_2$	✔	✗
$\sigma_c\,(T_1 \times T_2) = T_1 \times (\sigma_c\,T_2)$	✔	✗
$\sigma_c\,(T_1 \bowtie T_2) = T_1 \bowtie (\sigma_c\,T_2)$	✔	✗
$\sigma_{c_1 \wedge c_2}\,(T_1 \times T_2) = (\sigma_{c_1}\,T_1) \times (\sigma_{c_2}\,T_2)$	✔	✗
$\sigma_{c_1 \wedge c_2}\,(T_1 \bowtie T_2) = (\sigma_{c_1}\,T_1) \bowtie (\sigma_{c_2}\,T_2)$	✔	✗

In order to allow query processor to execute similarity queries, the algebraic properties of similarity operators must be stated. In this technical report, we presented the Similarity Algebra with the properties holding for the unary similarity operators, that is, for range / reversed range selection and for k-nearest / k-farthest neighbor selection. Then, the incorporation of similarity queries in RDBMS becomes possible.

Another contribution is that as range / reversed range selections have the same properties of traditional selections, the RDBMS query optimizer can treat these selections as traditional one, and traditional optimization algorithms can be used to optimize them. Distinctly from range / reversed range and traditional selections, kNN / kFN selections have only three properties and five special cases (over the same query elements) to rewrite algebraic expressions. Then, they do not allow optimization algorithms equivalent to traditional and range / reversed range selections. Specific optimization algorithms should be implemented in the RDBMS query optimizer to optimize these kind of selections.

References

1. Codd, E.F.: A relational model of data for large shared data banks. Communications of the ACM (CACM) **13**(6) (1970) 377–387
2. Gaede, V., Günther, O.: Multidimensional access methods. ACM Computing Surveys (CSUR) **30**(2) (1998) 170–231
3. Faloutsos, C.: Indexing of multimedia data. In: Multimedia Databases in Perspective. Springer (1997) 219–245
4. Barioni, M.C.N., Razente, H.L., Traina, A.J.M., Traina-Jr., C.: SIREN: A similarity retrieval engine for complex data. In Dayal, U., Whang, K.Y., Lomet, D.B., Alonso, G., Lohman, G.M., Kersten, M.L., Cha, S.K., Kim, Y.K., eds.: VLDB'06: Proceedings of the 32nd International Conference on Very Large Data Bases, Seoul, Korea, ACM (2006) 1155–1158

5. Barioni, M.C.N., Razente, H.L., Traina, A.J.M., Traina-Jr., C.: Seamlessly integrating similarity queries in SQL. Technical Report 252, Institute of Mathematics and Computer Sciences (ICMC) / University of São Paulo (USP) (2006)
6. Ferreira, M.R.P., Traina-Jr., C., Traina, A.J.M.: An efficient framework for similarity query optimization. In Samet, H., Schneider, M., Shahabi, C., eds.: ACM GIS'07: Proceedings of the 15th ACM International Symposium on Advances in Geographic Information Systems, November 7-9, 2007, Seattle, WA, USA, ACM Press (2007) 396–399
7. Adali, S., Bonatti, P., Sapino, M., Subrahmanian, V.: A multi-similarity algebra. In Haas, L.M., Tiwary, A., eds.: SIGMOD'98: Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data, Seattle, WA, USA, ACM Press (1998) 402–413
8. Atnafu, S., Chbeir, R., Coquil, D., Brunie, L.: Integrating similarity-based queries in image dbmss. In: SAC'04: Proceedings of the 2004 ACM Symposium on Applied Computing, ACM Press (2004) 735–739
9. Belohlávek, R., Opichal, S., Vychodil., V.: Relational algebra for ranked tables with similarity: properties and implementation. In Berthold, M.R., Shawe-Taylor, J., Lavrac, N., eds.: IDA'07: Proceedings of the 7th International Symposium on Intelligent Data Analysis. Volume 4723 of Lecture Notes in Computer Science., Springer Verlag (2007) 140–151
10. Penzo, W.: Rewriting rules to permeate complex similarity and fuzzy queries within a relational database system. IEEE Transactions on Knowledge and Data Engineering (TKDE) **17**(2) (2005) 255–270
11. Adali, S., Bufi, C., Sapino, M.L.: Ranked relations: query languages and query processing methods for multimedia. Multimedia Tools and applications Journal (MTAJ) **24**(3) (2004) 197–214
12. Li, C., Chang, K.C.C., Ilyas, I.F., Song, S.: RankSQL: query algebra and optimization for relational top-k queries. In Vijayaraman, T.M., Buchmann, A.P., Mohan, C., Sarda, N.L., eds.: SIGMOD'05: Proceedings of the ACM International Conference on Management of Data, June 14–16, 2005, Baltimore, Maryland, USA, ACM Press (2005) 131–142
13. Traina-Jr., C., Traina, A.J.M., Vieira, M.R., Arantes, A.S., Faloutsos, C.: Efficient processing of complex similarity queries in RDBMS through query rewriting. In: CIKM'06: Proceedings of the 15th ACM International Conference on Information and Knowledge Management, Arlington, Virginia, USA, ACM Press (2006) 4–13

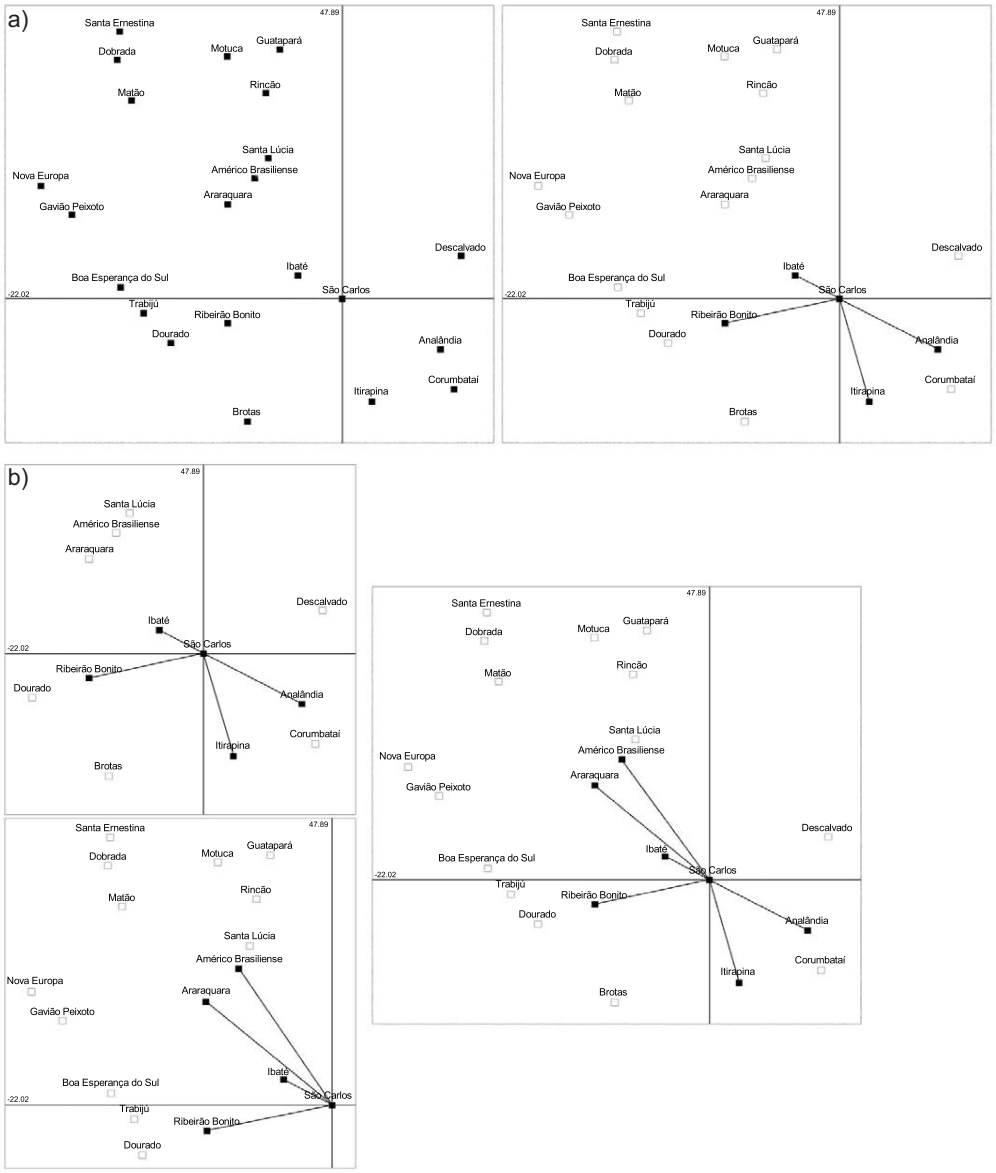

Fig. 7. Non-distributivity of kNN selection operation over union binary operator. Query presents in Example 5: a) Query first executed the union of relations `CitySaoCarlos` and `CityAraraquara` and, then, kNN selection over its result. b) Query executed first kNN selection in the relations `CitySaoCarlos` and `CityAraraquara` and, then, the union of their results.

Fig. 8. Non-distributivity of kNN selection operation over difference binary operator. Query presents in Example 6: a) Query first executed the difference of relations `CityAraraquara` and `CitySaoCarlos` and, then, kNN selection over its result. b) Query executed first kNN selection in the relation `CityAraraquara` and, then, the difference of its result with the relation `CitySaoCarlos`. c) Query executed first kNN selection in the relations `CityAraraquara` and `CitySaoCarlos` and, then, the difference of their results.

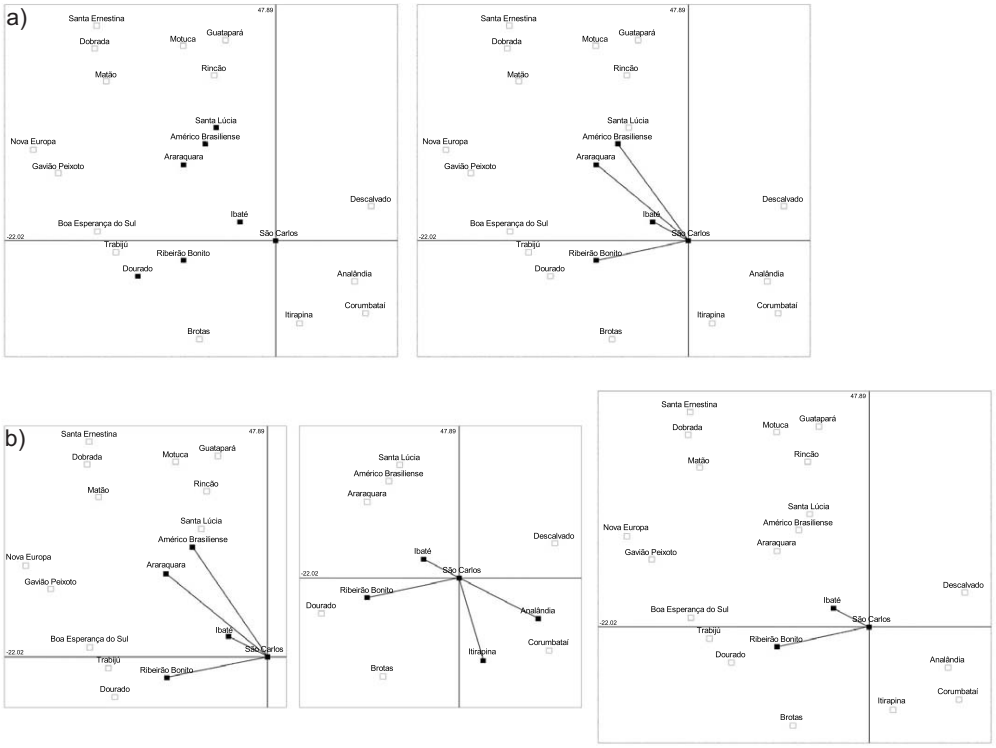

Fig. 9. Non-distributivity of kNN selection operation over intersection binary operator. Query presents in Example 7: a) Query first executed the intersection of relations `CityAraraquara` and `CitySaoCarlos` and, then, kNN selection over its result. b) Query executed first kNN selection in the relations `CityAraraquara` and `CitySaoCarlos` and, then, the intersection of their results.