Algebraic Properties to Optimize kNN Queries

Mônica R. P. Ferreira[1,3], Lucio F. D. Santos[1], Agma J. M. Traina[1],
Ires Dias[2], Richard Chbeir[3], Caetano Traina Jr.[1]

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

Abstract. New applications that are being required to employ Database Management Systems (DBMSs), such as storing and retrieving complex data (images, sound, temporal series, genetic data, etc.) and analytical data processing (data mining, social networks analysis, etc.), increasingly impose the need for new ways of expressing predicates. Among the new most studied predicates are the similarity-based ones, where the two commonest are the similarity range and the k-nearest neighbor predicates. The k-nearest neighbor predicate is surely the most interesting for several applications, including Content-Based Image Retrieval (CBIR) and Data Mining (DM) tasks, yet it is also the most expensive to be evaluated. A strong motivation to include operators to execute the k-nearest neighbor predicate inside a DBMS is to employ the powerful resource of query rewriting following algebraic properties to optimize query execution. Unfortunately, too few properties of the k-nearest neighbor operator have been identified so far that allow query rewriting rules leading to effectively more efficient query execution. In fact, a k-nearest neighbor operator does not even commute with either other k-nearest neighbor operator or any other attribute comparison operators (similarity range or any other of the traditional attribute comparison operator). In this paper, we investigate a new class of properties for the k-nearest neighbor operator based not on expression equivalence, but on result set inclusion. We develop a complete set of properties based on set inclusion, which can be successfully employed to rewrite query expressions involving k-nearest neighbor operators combined to any of the traditional attribute comparison operators or to other k-nearest neighbor and similarity range operators. We also give examples of how applying those properties to rewrite queries improve retrieval efficiency.

Categories and Subject Descriptors: H.2.4 [**Database management**]: Systems—*Query processing*

Keywords: algebraic properties, query optimization, similarity algebra, unary similarity queries

1. INTRODUCTION

The relational database management systems (RDBMSs) were initially conceived to handle data composed of numbers and small character strings. Those traditional data can be compared using both exact matching ($=$ and $\neq$) and relational operators ($<$, $\leq$, $>$ and $\geq$), which we call traditional operators. Almost every component of a RDBMS is able to process only data that meet the exact matching and relational operators, including the indexing structures, query optimizers and selectivity estimators. Nowadays, new applications that are being required to employ Database Management Systems (DBMSs) increasingly requires new kinds of comparison operators. Examples of those applications include storing and retrieving complex data (images, sound, multi-dimensional measurements, temporal series, genetic data, etc.) and analytical data processing (data mining, social networks evaluation, etc.). The relational operators do not apply over complex data, and they are not useful to perform analytical processing over them either. The exact matching operators have also few uses, as few complex data are identical to others. For those applications, the notion of similarity emerges naturally

This work has been supported by FAPESP, CAPES, CNPq, CNRS and Microsoft Research.

as the way to compare pairs of elements in complex domains, pushing the degree of similarity among data elements as the most important factor [Faloutsos 1997].

The two most well-known similarity-based comparison operators are the similarity range and the k-nearest neighbor ones. A query using the similarity range operator – usually known as a range query – returns the elements closer than or at the same distance to a threshold from a query element (also called the query center). A query using the k-nearest neighbor operator (kNN) – usually known as a k-nearest neighbor query – returns the k elements nearest to the query center. The kNN predicate is the most interesting for several similarity search applications, including retrieving of complex elements by content such as Content-Based Image Retrieval (CBIR) and Data Mining (DM) tasks [Böhm et al. 2000; Berchtold et al. 2001; Bartolini 2002; Falchi et al. 2008].

To be able to perform similarity comparisons, a function must be defined to evaluate how similar two elements are. Assuming that more similar elements can be seen as closer in a given data space, this function can be defined as a distance function d, which is the basis to create a metric space. Formally, a metric space is a pair $M = \langle \mathbb{S}, d \rangle$, where $\mathbb{S}$ denotes the universe of valid elements, and d is a function $d : \mathbb{S} \times \mathbb{S} \rightarrow \mathbb{R}^+$ that expresses the distance between elements of $\mathbb{S}$, and which satisfies the following properties: symmetry: $d(s_1, s_2) = d(s_2, s_1)$; non-negativity: $0 < d(s_1, s_2) < \infty$, if $s_1 \neq s_2$ and $d(s_1, s_1) = 0$; and triangular inequality: $d(s_1, s_2) \leq d(s_1, s_3) + d(s_3, s_2), \forall s_1, s_2, s_3 \in \mathbb{S}$.

Nowadays, RDBMSs provide very little support for similarity queries, leading the developers of applications that handle complex elements or that execute data mining processes to code the similarity based algorithms inside the applications. However, a much more elegant and flexible way would be providing support for similarity queries inside the RDBMS. A great motivation should be to employ the powerful resource of query rewriting following algebraic properties to optimize the execution of non-trivial, compound queries. That is, queries that involve several predicates, and in particular, queries in which at least one of the predicates are based on similarity. An example of a query that meets this expectation over a database of medical exams is: "*Select the five radiographies taken the last month that are the most similar to this one from my current patient and whose medical report was written by Dr. House and include at least three of the words {`calcification, cyst, fibroadenoma, fibrocyst`}*". This query includes four predicates:

—An exact matching predicate: select the exams whose report was `written_by='Dr. House'`;

—A relational comparison predicate: select the exams having `date>=Today-30`;

—A similarity range predicate: select the exams whose report is in the range of at most 1 word excluding the specified ones (using a distance function that counts how many words of those given one are not in the medical report);

—A similarity k-nearest neighbor predicate: select the five images similar to the given one (using image features and a distance function that are not specified here).

Choosing the best way to execute compound queries depends on the existence of two or more equivalent ways to represent it. For example, it is a well-known property that exact matching and relational comparisons can be performed in any order – they meet the commutative property. Thus, it is possible to elect the most selective predicate among those that refer to indexed attributes to be executed first. In fact, the rich set of algebraic properties existing over the traditional operators allows the optimizer of a RDBMS to rewrite a compound query into several equivalent representations for execution, attaining speedups of several magnitude orders.

Unfortunately, too few equivalence properties exists for the k-nearest neighbor operator. In fact, it does not even commute, neither with other k-nearest neighbor or similarity range operator nor with any of the traditional attribute comparison operators. In a more formal description, $\sigma_{(\theta)} \left(\sigma_{(\ddot{\theta})} T \right) \neq \sigma_{(\ddot{\theta})} \left(\sigma_{(\theta)} T \right)$ where $\ddot{\theta}$ is a k-nearest neighbor operator and θ is either a k-nearest neighbor or any other attribute comparison operator (traditional operators or the similarity range) [Ferreira et al. 2009].

This article proposes a new class of properties for the kNN operator that is not based on expression equivalence, but on inclusion expression – for example, acknowledging that $\sigma_{(\theta)}\left(\sigma_{(\ddot{\theta})}T\right) \subseteq \sigma_{(\ddot{\theta})}\left(\sigma_{(\theta)}T\right)$. We develop a complete set of properties based on inclusion that can be successfully employed to rewrite query expressions involving k-nearest neighbor operators combined to any of the traditional attribute comparison operators or to other k-nearest neighbor and similarity range operators.

The remainder of this article is structured as follows. Section 2 discusses previous works and shows properties and algebraic manipulations already studied for the k-nearest neighbor operator. Section 3 presents concepts existing in the literature that are important to help understanding the current article. Section 4 presents a broad collection of set inclusion properties for kNN predicates that enlarges the resources that the query DBMS rewriter module can count on to choose good query execution plans. Section 5 shows experiments performed over real datasets that include traditional and complex attributes, evaluating compound queries mixing similarity, relational and exact match predicates, and shows that our properties can indeed improve query execution efficiency. In Section 6 we digest the results of the experiments aiming at using the proposed properties in a DBMS interpreter/optimizer. Finally, Section 7 concludes this article.

2. RELATED WORK

There are several extensions to the relational algebra in the literature aimed at including similarity-based functionality in RDBMS from various perspectives. Usually, multimedia information systems treat similarity using four different approaches: 1) Rank: aims at providing only the top-k results, according to a user-specified ranking criterion; 2) Fuzzy: associates the similarity to an uncertainty or imprecise grade and provides fuzzy logic-based methods to solve queries; 3) Hybrid: mixes the rank and the fuzzy approaches; 4) Exact: evaluates each element according to how well it fits similarity criteria given for the query.

In 1998, [Adali et al. 1998] presented the *Multi-Similarity Algebra* (MSA). Although this work was the first one to consider a similarity extension to relational algebra, it is not fully consistent with the relational model because the MSA cannot be used for modeling, optimizing and processing queries that combine traditional and similarity-based operators [Atnafu et al. 2004].

Further developing the "rank approach", Adali et al. [2004] introduced an algebra for querying ranked relations and proved various coherence preservation properties for this algebra, which shows when it is possible to guarantee that distinct rank columns induce the same ordering among tuples. Li et al. [2005] proposed the "rank-relational" algebra, which supports ranking as a first-class construct. In Adali et al. [2007], the authors presented an algebra that supports mining and fusion tasks using the reuse and combination of ranked elements.

Regarding the "fuzzy approach", the first work presenting an extension to the relational algebra was Montesi and Trombetta [1999], where the authors included two new operators (the top and ϵ-similar ones) to formulate queries about the similarity of elements represented in the fuzzy relational model. The *Similarity Algebra for Multimedia Extended with Weights* (SAME^W), proposed by Ciaccia et al. [2000], generalized the relational algebra to allow the formulation of compound similarity queries over multimedia databases. SAME^W also introduced two new operators (the cut and top ones) that are useful for range and kNN queries. Picariello and Sapino [2002] developed an algebra for dealing with fuzziness related to the semantic descriptors of an image. Montesi et al. [2003] extended the relational algebra to correctly capture the imprecise information related to several kinds of multimedia data. Schmitt and Schulz [2004] introduced the *Similarity Algebra* (SA), raising vagueness and weighting into relational algebra.

The work of Belohlavek et at. [2007] targeted a "hybrid approach", in which both rank and fuzzy

approaches were combined to extend the relational algebra over domains enabled to be evaluated by similarity. Also, the concept of a ranked table based on the calculus of fuzzy relations was introduced.

Pursuing the "exact approach", Atnafu et al. [2001] initially defined a similarity-based algebra able to identify equivalence properties in an expression involving similarity-based operators. However, this first work only targeted similarity comparisons over images stored in a database, aiming at formalizing similarity queries over images that are indirectly evaluated over feature vectors extracted from each image, instead of directly comparing the images. The first work to target equivalence properties of similarity-based operators that hold on general metric spaces were proposed by Traina Jr. et al. [2006]. It proposed an extension for the relational algebra to allow combining two or more similarity predicates over the same query reference connected by Boolean operators. Ferreira et al. [2009] further improved that work, evaluating the properties holding over any combination of traditional predicates with similarity-based ones. The authors identified the fundamental properties that allow integrating both traditional and similarity-based unary operators into the Relational Algebra, providing a complete set of equivalence rules to optimize queries combining at least one similarity-based with any other operators through query rewriting. The analyzed properties are able to treat queries centered either at the same or at distinct query elements. In another work, Chbeir and Laurent [2009] also studied the inclusion properties of multimedia operators (range and KNN) in order to identify the minterms in a query sets. The aim of this study was to adapt multimedia fragmentation database. Silva and Aref [2009] analyzed relationships among similarity join and similarity group-by operators.

However, all of these works were elaborated using only equivalence properties that, as shown in Ferreira et al. [2009], provide too few properties for the kNN predicate. In this work, we explore another class of properties that also holds on similarity predicates and their combination with the traditional ones, namely the set inclusion properties. Those properties also allow choosing the best definition of what interpretation the DBMS must assume for a query expression involving kNN predicates, due to the lack of conformance to the commutativity property of those predicates, and we argue that the performance-guided consensus existing in the literature is not the best choice.

3. CONCEPTS

In this article, we employ the basic notation introduced in Ferreira et al. [2009] to represent database relations storing at least one attribute that can be referenced in similarity predicates. We also follow the same conceptual definitions of that work, which are summarized as follows. The term 'simple attribute' refers to an attribute of a traditional, scalar data type that is compared by the traditional relational or identity comparison operators, and the term 'simple domain' refers to the traditional scalar data types. In the same way, we use the term 'complex attribute' to refer to an attribute that can be compared by similarity, that is, it is drawn from a metric domain where a distance function was defined.

Let $A_h \subset \mathbb{A}_h$ be a simple attribute in a domain $\mathbb{A}_h$ that allows comparisons using traditional (θ) operators. Let $S_j \subset \mathbb{S}_j$ be a complex attribute in a domain $\mathbb{S}_j$ in a metric space that allows comparisons using similarity (either $\ddot{\theta}$ or $\hat{\theta}$) operators; and let T be a relation with any number of both simple and complex attributes. In this way, $\mathbb{T} = \{\mathbb{A}_1, \ldots, \mathbb{A}_m, \mathbb{S}_1, \ldots, \mathbb{S}_p\}$ represents a schema relation, and a relation $T \subset \mathbb{T}$ is a set of elements represented as tuples of the format $T = (A_1, \ldots, A_m, S_1, \ldots, S_p)$. Each tuple $t = \langle a_1, \ldots, a_m, s_1, \ldots, s_p \rangle$ has values a_h ($1 \leq h \leq m$) obtained in the the corresponding domain $\mathbb{A}_h$ and values s_j ($1 \leq j \leq p$) obtained in the corresponding domain $\mathbb{S}_j$. Thus, let $t_i(S_j)$ ($1 \leq i \leq n$, where n is the cardinality of the relation) be the value of the complex attribute S_j of the i^{th} tuple in the relation, and correspondingly let $t_i(A_h)$ be the value of the simple attribute A_h. To alleviate the notation, we also use S and $\mathbb{S}$ to refer to complex attribute S_j and its respective domain $\mathbb{S}_j$ (and correspondingly A and $\mathbb{A}$ to refer to a simple attribute A_h and its respective domain $\mathbb{A}_h$) whenever only one attribute is involved in the text.

Table I. Table of symbols.

Symbol	Description
$\mathbb{S}$	Complex domain over which similarity conditions can be expressed.
$\ddot{\theta}$	kNN operator.
$\hat{\theta}$	Similarity range operator.
θ	Exact match or relational comparison operator.
S, S_1, S_2	Complex attributes taken in domain $\mathbb{S}$.
d, d_1, d_2	Distance functions in $\mathbb{S}$.
k, $k_1, k_2 \in \mathbb{N}^*$	Quantities of elements to be retrieved, used as query stop thresholds.
s_q, s_{q_1}, s_{q_2}	Query references.

Traditional selections follow the format $\sigma_{(A\ \theta\ a)}\ T$, where θ is a traditional 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. Likewise, 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. Obviously, in this formulation a domain $\mathbb{S}$ where similarity queries can be applied is a metric domain. The two most common similarity operators θ_c are the range and the k-nearest neighbor operators. As their properties can be different from the relational and exact match counterparts, we use the symbols $\hat{\theta}$ and $\ddot{\theta}$ to represent the range and the k-nearest neighbor operators, and the symbols $\hat{\sigma}$ and $\ddot{\sigma}$ to represent the range and the k-nearest neighbor selection queries, respectively.

A range query – R_q – returns every element that differs from the query reference in at most the similarity threshold. Its formal definition and properties are defined in Ferreira et at. [2009]. A k-nearest neighbor query – kNN_q – returns the k elements nearest to the query reference. The goal of this article is to define algebraic properties aimed at optimizing kNN queries. In the sequence we summarize the equivalence properties defined in Ferreira et al. [2009].

Applying similarity queries over a metric domain $\mathbb{S}$ implies that a distance function exists over the domain. Therefore, for a relational expression to be able to refer to a complex attribute by similarity, the domain's metric must be known. We assume here that the assignment of a metric to a complex attribute is stated as a constraint over the attribute. Considering an implementation of those concepts, we assume that the SQL language is extended to include the ability of defining the attribute metric either as a row or as a table constraint in the table definition command, in the same way as it was done in SIREN [Barioni et al. 2006; Barioni et al. 2009].

There are three equivalence-based properties for kNN queries in general and two special cases that hold only for kNN queries sharing the same query reference and distance functions. Table II summarizes those properties, where the employed symbols are defined in Table I.

Property 1 expresses that a sequence of complex predicates (i.e. conjunctions of $\ddot{\theta}$ operators), can be rewritten into a sequence of intersection operations. Notice that the query centers and the distance functions employed can be different at each predicate. A special case exists for this property: if the query centers and the distance functions are the same for all the predicates, then only the kNN selection with the smallest number of elements needs to be executed [Traina-Jr. et al. 2006].

Property 2 expresses that a disjunction of complex predicates (i.e., disjunctions of $\ddot{\theta}$ operators), can be rewritten into a sequence of union operations. Notice that if relation T is a set, this property warrants that duplications will be correctly eliminated. A special case exists for this property too: if the query centers and the distance functions are the same for all the predicates then only the kNN selection with the largest number of elements needs to be executed [Traina-Jr. et al. 2006].

Strictly speaking, Property 3 does not refer to a kNN predicate, as it just states that the datasets that are the results of two kNN selections are commutative. However, as the kNN operator by itself

Table II. The three equivalence-based properties for kNN queries in general and the two special cases that holds for kNN queries sharing the same query reference.

1	$\ddot{\sigma}_{(S_1\ \ddot{\theta}(d_1,k_1)\ s_{q_1})\wedge(S_2\ \ddot{\theta}(d_2,k_2)\ s_{q_2})}\ T = \left(\ddot{\sigma}_{(S_1\ \ddot{\theta}(d_1,k_1)\ s_{q_1})}\ T\right) \cap \left(\ddot{\sigma}_{(S_2\ \ddot{\theta}(d_2,k_2)\ s_{q_2})}\ T\right)$
1.1	$\left(\ddot{\sigma}_{(S\ \ddot{\theta}(d,k_1)\ s_q)}\ T\right) \cap \left(\ddot{\sigma}_{(S\ \ddot{\theta}(d,k_2)\ s_q)}\ T\right) = \ddot{\sigma}_{(S\ \ddot{\theta}(d,k_1)\ s_q)\wedge(S\ \ddot{\theta}(d,k_2)\ s_q)}\ T = \ddot{\sigma}_{(S\ \ddot{\theta}(d,min(k_1,k_2))\ s_q)}\ T$
2	$\ddot{\sigma}_{(S_1\ \ddot{\theta}(d_1,k_1)\ s_{q_1})\vee(S_2\ \ddot{\theta}(d_2,k_2)\ s_{q_2})}\ T = \left(\ddot{\sigma}_{(S_1\ \ddot{\theta}(d_1,k_1)\ s_{q_1})}\ T\right) \cup \left(\ddot{\sigma}_{(S_2\ \ddot{\theta}(d_2,k_2)\ s_{q_2})}\ T\right)$
2.1	$\left(\ddot{\sigma}_{(S\ \ddot{\theta}(d,k_1)\ s_q)}\ T\right) \cup \left(\ddot{\sigma}_{(S\ \ddot{\theta}(d,k_2)\ s_q)}\ T\right) = \ddot{\sigma}_{(S\ \ddot{\theta}(d,k_1)\ s_q)\vee(S\ \ddot{\theta}(d,k_2)\ s_q)}\ T = \ddot{\sigma}_{(S\ \ddot{\theta}(d,max(k_1,k_2))\ s_q)}\ T$
3	$\left(\ddot{\sigma}_{(S_1\ \ddot{\theta}(d_1,k_1)\ s_{q_1})}\ T\right) \cap \left(\ddot{\sigma}_{(S_2\ \ddot{\theta}(d_2,k_2)\ s_{q_2})}\ T\right) = \left(\ddot{\sigma}_{(S_2\ \ddot{\theta}(d_2,k_2)\ s_{q_2})}\ T\right) \cap \left(\ddot{\sigma}_{(S_1\ \ddot{\theta}(d_1,k_1)\ s_{q_1})}\ T\right)$, $\left(\ddot{\sigma}_{(S_1\ \ddot{\theta}(d_1,k_1)\ s_{q_1})}\ T\right) \cup \left(\ddot{\sigma}_{(S_2\ \ddot{\theta}(d_2,k_2)\ s_{q_2})}\ T\right) = \left(\ddot{\sigma}_{(S_2\ \ddot{\theta}(d_2,k_2)\ s_{q_2})}\ T\right) \cup \left(\ddot{\sigma}_{(S_1\ \ddot{\theta}(d_1,k_1)\ s_{q_1})}\ T\right)$

is not commutative either with other selection operators or with itself, each selection must be executed separately and the intersection (for conjunctive conditions) or the union (for disjunctive conditions) of their results can be manipulated. Thus, Property 3 is useful to handle kNN predicates.

Besides those three properties involving only kNN predicates, equivalent ones hold when combining kNN predicates with either similarity range or traditional predicates. Thus, Properties 1 to 3 hold for any combination of "$\ddot{\sigma}[\cap,\cup]\hat{\sigma}$" and "$\ddot{\sigma}[\cap,\cup]\sigma$", even when the query centers, distance functions and query stop thresholds are distinct.

4. KNN PROPERTIES

In this section we describe the properties that hold for kNN predicates taking into account the set inclusion properties of querying by similarity datasets of complex elements. For illustration purposes, we first describe a dataset used as a running example for the remainder of this article. Although the presented concepts hold for any dataset of complex elements that can be represented in a metric space, without loss of generality we use a dataset of geographical coordinates under the Euclidean distance function, as it is both a metric space and makes it intuitively easy to understand the presented examples. The running example is a relation of cities

`USCities={Id, Name, State, Lat, Long, Coordinate, PerCapita, PctPovertyFam}`,

where the attribute `Coordinate` is the complex one, that is, the Euclidean distance function is defined over the domain of `Coordinate`, so the similarity predicates can be answered over it. In order to make easier to understand the examples, we assume that there is a function `Coord(USCities.Name)` that returns the `Coordinate` of the city named `Name`.

4.1 Formal Definition

A kNN query returns the k elements most similar to the query reference based on a given distance function. Usually, it is defined as in Definition 1.

Definition 1. **k-Nearest Neighbor query - kNN_q - Conventional**: Given that $\mathbb{S}$, $\ddot{\theta}$, S, d, $k \in \mathbb{N}^*$ and $s_q \in \mathbb{S}$ as defined in Table I, the query $\ddot{\sigma}_{(S\ \ddot{\theta}(d,k)\ s_q)}\ T$ returns the tuples $\{t_1,\ldots,t_k\} \subseteq T$ such that, for each $i = 1,\ldots,k$, the value of the attribute S in the tuple t_i - $t_i(S)$ - is one of the k elements in S nearest to the query element s_q based on the distance function d. That is,

$$\ddot{\sigma}_{(S\ \ddot{\theta}(d,k)\ s_q)}\ T\ = \{t_i \in T \mid \forall\, t \in T - T', d\,(t_i(S), s_q) \leq d\,(t(S), s_q)\}\ , \tag{1}$$

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

Defining the kNN_q in this way is easier to understand and it is the most commonly used in the database literature to explain the kNN operator. However, to prove the inclusion-based properties, it is convenient to express the kNN_q query following algebraic rules, where the concept being defined cannot itself be employed in its definition. Therefore, we express its formal definition as follows.

Definition 2. **k-Nearest Neighbor query - kNN_q - Formal**: Given that $\mathbb{S}$, $\ddot{\theta}$, S, d, $k \in \mathbb{N}^*$ and $s_q \in \mathbb{S}$ as defined in Table I, the query $\ddot{\sigma}_{(S\ \ddot{\theta}(d,k)\ s_q)}\ T$ returns the tuples

$$\ddot{\sigma}_{(S\ \ddot{\theta}(d,k)\ s_q)}\ T = \{t_1, \ldots, t_k\}\ , \tag{2}$$

where

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

In other words, since T is a finite set, we can write:

$$\ddot{\sigma}_{(S\ \ddot{\theta}(d,k)\ s_q)}\ T = \{t_1, \ldots, t_k\}\ , \tag{3}$$

where

$$\begin{aligned}
t_1 =& \{t_1\ \in\ T |\ d\left(t_1(S), s_q\right) = \min\left\{d\left(t(S), s_q\right);\ t\ \in\ T\right\}\}\ ,\\
t_2 =& \{t_2\ \in\ T - \{t_1\} |\ d\left(t_2(S), s_q\right) = \min\left\{d\left(t(S), s_q\right);\ t\ \in\ T - \{t_1\}\right\}\}\ ,\\
&\vdots\\
t_k =& \{t_k\ \in\ T - \{t_1, t_2, \ldots, t_{k-1}\} |\ d\left(t_k(S), s_q\right) = \min\left\{d\left(t(S), s_q\right);\ t\ \in\ T - \{t_1, \ldots, t_{k-1}\}\right\}\}\ .
\end{aligned}$$

Thus, by definition, $d(t_1(S), s_q) \leq d(t_2(S), s_q) \leq \ldots \leq d(t_k(S), s_q)$.

4.2 kNN Properties

Now we are ready to show the set inclusion-based properties valid for kNN selection predicates.

4.2.1 *Idempotent property.* An operation is said to be idempotent when repeating this operation multiple times has exactly the same result as applying it once. Property 4.1 shows that a kNN selection operation meets this property.

PROPERTY 4.1. *Let T be a relation, S in T be a complex attribute taken in domain $\mathbb{S}$ over which the similarity condition is expressed, and $\ddot{\theta}$, d, k and s_q as defined in Table I. Then the kNN idempotent property is expressed as*

$$\ddot{\sigma}_{(S\ \ddot{\theta}(d,k)\ s_q)}\left(\ddot{\sigma}_{(S\ \ddot{\theta}(d,k)\ s_q)}\ T\right) = \ddot{\sigma}_{(S\ \ddot{\theta}(d,k)\ s_q)}\ T\ .\ \square \tag{4}$$

Example 4.1 illustrates this property. In this example, we want to find the 5 most similar cities of "New York city-NY".

EXAMPLE 4.1. *"Select the 5 nearest cities of "New York city-NY", considering the Euclidean distance L_2".*

Algebraically, this query is expressed as $\ddot{\sigma}_{(\texttt{Coordinate}\ \ddot{\theta}(L_2,5)\ Coord(\text{New York}))}\ \texttt{USCities}$. The execution of this query returns 5 tuples `{New York city-NY, Guttenberg town-NJ, Hoboken city-NJ, Union City city-NJ, West New York town-NJ}`. If we repeat the query in multiple selection as in:

$$\ddot{\sigma}_{(\texttt{Coordinate}\ \ddot{\theta}(L_2,5)\ Coord(\text{New York}))}\left(\ddot{\sigma}_{(\texttt{Coordinate}\ \ddot{\theta}(L_2,5)\ Coord(\text{New York}))}\ \texttt{USCities}\right)\ ,$$

the results are always the same, due to the idempotent property of kNN_q. Although this property is barely useful for query rewriting purposes, it is nevertheless required to have a well-defined algebra.

4.2.2 *Inclusion properties.* The following properties show the inclusion properties when executing kNN in sequence with itself and with other traditional and similarity operations.

Property 4.2 treats the relationship between conjunctions of $\ddot{\theta}$ operators and the composition of kNN selection operations. Following this property, the result of a kNN executed over a conjunction of $\ddot{\theta}$ operators is included in the result of the conjunction of kNN selection operations.

PROPERTY 4.2. *Let T be a relation, S_1, S_2 in T be a complex attribute taken in domain $\mathbb{S}$ over which the similarity condition is expressed, and $\ddot{\theta}$, d_1, d_2, k_1, k_2, s_{q_1} and s_{q_2} as defined in Table I. Then*

$$\underbrace{\ddot{\sigma}_{(S_1\ \ddot{\theta}(d_1,k_1)\ s_{q_1})\wedge(S_2\ \ddot{\theta}(d_2,k_2)\ s_{q_2})}\ T}_{(i)} \subseteq \underbrace{\ddot{\sigma}_{(S_1\ \ddot{\theta}(d_1,k_1)\ s_{q_1})}\left(\ddot{\sigma}_{(S_2\ \ddot{\theta}(d_2,k_2)\ s_{q_2})}T\right)}_{(ii)} \quad . \quad \square \tag{5}$$

Due to space limitations, we do not prove every property derived in this article. We do however prove this property, noting that the others follow suit.

PROOF. By Definition 1, we have:

$$\ddot{\sigma}_{(S_2\ \ddot{\theta}(d_2,k_2)\ s_{q_2})}\ T = \{t_i \in T | \forall t \in T - T_2', d_2\left(t_i(S_2), s_{q_2}\right) \leq d_2\left(t(S_2), s_{q_2}\right)\} = D \subseteq T,$$

where $T_2' = \emptyset$ if $i = 1$ and $T_2' = \{t_1, \ldots, t_{i-1}\}$ if $1 < i \leq k_2$.

Replacing in (ii), we have:

$$(ii) = \ddot{\sigma}_{(S_1\ \ddot{\theta}(d_1,k_1)\ s_{q_1})}\ D = \{t_i \in D | \forall t \in D - T_1', d_1\left(t_i(S_1), s_{q_1}\right) \leq d_1\left(t(S_1), s_{q_1}\right)\},$$

where $T_1' = \emptyset$ if $i = 1$ and $T_1' = \{t_1, \ldots, t_{i-1}\}$ if $1 < i \leq k_1$.

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

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

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

To exemplify this property, suppose that we are looking for the 5 cities most similar to "New York city-NY" and the 3 cities most similar to "Jersey City city-NJ", as presented in Example 4.2.

EXAMPLE 4.2. *"Select the 5 nearest cities of "New York city-NY" and the 3 nearest cities of "Jersey City city-NJ" considering the Euclidean distance function L_2".*

Algebraically, this query can be represented as:

$$\ddot{\sigma}_{(\texttt{Coordinate}\ \ddot{\theta}(L_2,5)\ Coord(\text{New York}))\wedge(\texttt{Coordinate}\ \ddot{\theta}(L_2,3)\ Coord(\text{New Jersey}))}\ \texttt{USCities}\ . \tag{6}$$

It returns 2 tuples corresponding to `{Hoboken city-NJ, Union City city-NJ}`. Applying algebraic properties, we can transform this algebraic expression into a composition of kNN/kNN selection operation, such as:

$$\ddot{\sigma}_{(\texttt{Coordinate}\ \ddot{\theta}(L_2,3)\ Coord(\text{New Jersey}))}\left(\ddot{\sigma}_{(\texttt{Coordinate}\ \ddot{\theta}(L_2,5)\ Coord(\text{New York}))}\texttt{USCities}\right)\ , \tag{7}$$

which returns the 3 tuples `{Hoboken city-NJ, Union City city-NJ, Guttenberg town-NJ}`. Notice that the result of Equation 6 is included in the result of Equation 7.

Property 4.3 treats the relation between compositions of kNN and traditional selection operation, showing that the result of a traditional selection performed over a kNN selection is included in the result of a kNN selection performed over a traditional selection.

PROPERTY 4.3. *Let T be a relation, S in T be a complex attribute taken in domain $\mathbb{S}$ over which the similarity condition is expressed, and $\ddot{\theta}$, d, k and s_q as defined in Table I. Let also A in T be a traditional attribute taken in domain $\mathbb{A}$ over which the traditional condition is expressed, θ as defined in Table I, and a be either a constant taken in a domain of A or the value of another attribute from the same domain of A in the same tuple. Then*

$$\sigma_{(A\ \theta\ a)}\left(\ddot{\sigma}_{(S\ \ddot{\theta}(d,k)\ s_q)}\ T\right) \subseteq \ddot{\sigma}_{(S\ \ddot{\theta}(d,k)\ s_q)}\left(\sigma_{(A\ \theta\ a)}\ T\right) \quad . \quad \square \tag{8}$$

To illustrate this property, suppose that we want to find the 5 cities most similar to "New York City-NY" that have the per capita income greater than 22,400 (the per capita income of "New York City-NY") in Census 2000, according to Example 4.3.

EXAMPLE 4.3. *"Select the 5 cities nearest to "New York city-NY" considering the Euclidean distance L_2, and the per capita income greater than 22,400".*

Expressing Example 4.3 algebraically, we have:

$$\sigma_{(\texttt{PerCapita} > 22400)} \left(\ddot{\sigma}_{\left(\texttt{Coordinate}\ \ddot{\theta}(L_2,5)\ Coord(\text{New York})\right)} \texttt{USCities} \right) \quad , \tag{9}$$

which returns 2 tuples {Hoboken city-NJ, Guttenberg town-NJ}. If we apply the commutative property, executing the traditional selection before the kNN one, this algebraic expression is expressed as:

$$\ddot{\sigma}_{\left(\texttt{Coordinate}\ \ddot{\theta}(L_2,5)\ Coord(\text{New York})\right)} \left(\sigma_{(\texttt{PerCapita} > 22400)} \texttt{USCities} \right) \quad , \tag{10}$$

which returns the 5 tuples {Ridgefield borough-NJ,Guttenberg town-NJ,Edgewater borough-NJ, Cliffside Park borough-NJ, Hoboken city-NJ}. Notice that the result of Equation 9 is included in the result of Equation 10.

Analogously, Property 4.4 considers the relationship between *range* and kNN selection operations. According to this property, the result of the *range* selection executed over a kNN selection operation is included in the result of the kNN selection performed over the *range* selection operation.

PROPERTY 4.4. *Let T be a relation, S_1, S_2 in T be a complex attribute taken in domain $\mathbb{S}$ over which the similarity condition is expressed, ξ be the threshold and $\hat{\theta}$, $\ddot{\theta}$, d, k and s_q as defined in Table I. Then*

$$\hat{\sigma}_{\left(S_2\ \hat{\theta}(d_2,\xi)\ s_{q_2}\right)} \left(\ddot{\sigma}_{\left(S_1\ \ddot{\theta}(d_1,k)\ s_{q_1}\right)} T \right) \subseteq \ddot{\sigma}_{\left(S_1\ \ddot{\theta}(d_1,k)\ s_{q_1}\right)} \left(\hat{\sigma}_{\left(S_2\ \hat{\theta}(d_2,\xi)\ s_{q_2}\right)} T \right) \quad . \quad \square \tag{11}$$

The following example illustrates this property. Suppose that we want to show the 5 cities most similar to "New York city-NY" that are not farther than 210 km from "Albany city-NY" (the capital of New York State), as shown in Example 4.4.

EXAMPLE 4.4. *"Select the 5 cities nearest to "New York city-NY" and whose distances from "Albany city-NY" are not farther than 210 km, considering the Euclidean distance L_2".*

Using similarity algebra, this query is expressed as:

$$\hat{\sigma}_{\left(\texttt{Coordinate}\ \hat{\theta}(L_2,1.9)\ Coord(\text{Albany})\right)} \left(\ddot{\sigma}_{\left(\texttt{Coordinate}\ \ddot{\theta}(L_2,5)\ Coord(\text{New York})\right)} \texttt{USCities} \right) \quad . \tag{12}$$

It returns the two tuples {Guttenberg town-NJ, West New York town-NJ}. Applying the commutative property and executing the range selection before the kNN one, this algebraic expression is posed as:

$$\ddot{\sigma}_{\left(\texttt{Coordinate}\ \ddot{\theta}(L_2,5)\ Coord(\text{New York})\right)} \left(\hat{\sigma}_{\left(\texttt{Coordinate}\ \hat{\theta}(L_2,1.9)\ Coord(\text{Albany})\right)} \texttt{USCities} \right) \quad , \tag{13}$$

which returns the 5 tuples {West New York town-NJ,Fairview borough-NJ,Guttenberg town-NJ, Edgewater borough-NJ, Cliffside Park borough-NJ}. Notice that the result of Equation 12 is included in the result of Equation 13.

Property 4.5 describes the effect of applying the kNN selection over the traditional union binary operator. It states that the result of the kNN executed over the union of relations is included in the result of the union between kNN selection operations executed in both relations.

PROPERTY 4.5. *Let T_1 and T_2 be two relations, S be a complex attribute taken in domain $\mathbb{S}$ over which the similarity condition is expressed, and $\ddot{\theta}$, d, k and s_q as defined in Table I. Then*

$$\ddot{\sigma}_{\left(S\ \ddot{\theta}(d,k)\ s_q\right)} \left(T_1 \bigcup T_2 \right) \subseteq \left(\ddot{\sigma}_{\left(S\ \ddot{\theta}(d,k)\ s_q\right)} T_1 \right) \bigcup \left(\ddot{\sigma}_{\left(S\ \ddot{\theta}(d,k)\ s_q\right)} T_2 \right) . \quad \square \tag{14}$$

For instance, suppose that we want to find, in either New York State or New Jersey State, the 5 cities most similar to "New York city-NY". Example 4.5 expresses this query.

EXAMPLE 4.5. *"Select the 5 cities nearest to "New York city-NY" that belong either to the New York State or to the New Jersey State, considering the Euclidean distance L_2".*

This query, algebraically expressed as

$$\ddot{\sigma}_{(\texttt{Coordinate}\ \ddot{\theta}(L_2,5)\ Coord(\text{New York}))}\left(\sigma_{(State=NY)}\texttt{USCities} \bigcup \sigma_{(State=NJ)}\texttt{USCities}\right) \ , \qquad (15)$$

returns 5 tuples {Guttenberg town-NJ,New York city-NY,Union City city-NJ,Hoboken city-NJ, West New York town-NJ}. Applying the distributive property and executing the union over the kNN selection of each relation, this expression is rewritten as:

$$\left(\ddot{\sigma}_{(\texttt{Coordinate}\ \ddot{\theta}(L_2,5)\ Coord(\text{New York}))}\ \sigma_{(State=NY)}\texttt{USCities}\right) \ \bigcup$$
$$\left(\ddot{\sigma}_{(\texttt{Coordinate}\ \ddot{\theta}(L_2,5)\ Coord(\text{New York}))}\ \sigma_{(State=NJ)}\texttt{USCities}\right) \qquad (16)$$

which returns the 10 tuples {New York city-NY,Inwood CDP-NY,Harbor Hills CDP-NY,Bellerose Terrace CDP-NY, Saddle Rock village-NY,Cliffside Park borough-NJ,Guttenberg town-NJ, West New York town-NJ, Union City city-NJ, Hoboken city-NJ}. We can see that the result of Equation 15 is included in the result of Equation 16.

For the traditional set difference binary operator, there are two ways to relate it with the kNN selection operation. First, the result of the difference between the kNN selection in the first relation and the second relation is included in the result of the kNN executed over the difference of relations. In the second way, the result of the difference between the kNN selection in the first relation and the second relation is included in the result of the difference executed over kNN selection operations in both relations. These relationships are presented in Property 4.6.

PROPERTY 4.6. *Let T_1 and T_2 be two relations, S be a complex attribute taken in domain $\mathbb{S}$ over which the similarity condition is expressed, and $\ddot{\theta}$, d, k and s_q as defined in Table I. Then*

$$\left(\ddot{\sigma}_{(S\ \ddot{\theta}(d,k)\ s_q)}\ T_1\right) \ - \ T_2 \ \subseteq \ \ddot{\sigma}_{(S\ \ddot{\theta}(d,k)\ s_q)}\ (T_1 \ - \ T_2) \ \ ; \qquad (17)$$

$$\left(\ddot{\sigma}_{(S\ \ddot{\theta}(d,k)\ s_q)}\ T_1\right) \ - \ T_2 \ \subseteq \ \left(\ddot{\sigma}_{(S\ \ddot{\theta}(d,k)\ s_q)}\ T_1\right) \ - \ \left(\ddot{\sigma}_{(S\ \ddot{\theta}(d,k)\ s_q)}\ T_2\right). \quad \square \qquad (18)$$

In order to exemplify this property, suppose that we want to find the 5 cities nearest to "New York city-NY" that belong to the relation of cities that have the percentage of families in poverty level percentage smaller than or equal to 18.5 (that is, the poverty level percentage from the "New York city-NY" – stored in the PctPovertyFam attribute) and that are not cities of the New York state. Example 4.6 shows this query.

EXAMPLE 4.6. *"Select the 5 cities nearest to "New York city-NY" that have the percentage of families in poverty level PctPovertyFam$\leq$ 18.5 but are not cities of the New York state, considering the Euclidean distance L_2".*

Transforming this query into a similarity algebra expression, we have:

$$\ddot{\sigma}_{(\texttt{Coordinate}\ \ddot{\theta}(L_2,5)\ Coord(\text{New York}))}\left(\sigma_{(PctPovertyFam\leq 18.5)}\texttt{USCities} \ - \ \sigma_{(State=NY)}\texttt{USCities}\right), \qquad (19)$$

returns 5 tuples {Asbury Park city-NJ, Wood Ridge borough-NJ, Newark city-NJ, Union City city-NJ, Paterson city-NJ}. Applying the kNN selection over the first relation and executing its difference with the second relation, we have:

$$\left(\ddot{\sigma}_{(\texttt{Coordinate}\ \ddot{\theta}(L_2,5)\ Coord(\text{New York}))}\sigma_{(PctPovertyFam\leq 18.5)}\texttt{USCities}\right) \ - \ \sigma_{(State=NY)}\texttt{USCities}, \qquad (20)$$

which also returns 4 tuples: {Newark city-NJ, Paterson city-NJ, Union City city-NJ, Wood Ridge borough-NJ}. Therefore, we see that the result of Equation 20 is included in the result of Equation 19. Moreover, applying the distributive property and executing the difference over the kNN selection of each relation, the new expression:

$$\left(\ddot{\sigma}_{(\texttt{Coordinate}\ \ddot{\theta}(L_2,5)\ Coord(\text{New York}))}\ \sigma_{(PctPovertyFam\leq 18.5)}\ \texttt{USCities}\right) \ - \\ \left(\ddot{\sigma}_{(\texttt{Coordinate}\ \ddot{\theta}(L_2,5)\ Coord(\text{New York}))}\sigma_{(State=NY)}\ \texttt{USCities}\right) \ , \tag{21}$$

returns the 5 tuples {Kaser village-NY, Wood Ridge borough-NJ, Newark city-NJ, Paterson city-NJ, Union City city-NJ}. In this way, we see that the result of Equation 20 is included in the result of Equation 21 also.

Property 4.7 treats the relation of a kNN selection operation and the traditional intersection binary operator. The result of the intersection between the kNN selection executed in both relations is included in the result of the kNN executed over the intersection of both relations. Also, the result of the intersection between the kNN selection on the first relation and the kNN selection on the second relation is properly included in the result of the kNN executed over the intersection of both relations.

PROPERTY 4.7. *Let T_1 and T_2 be two relations, S be a complex attribute taken in domain $\mathbb{S}$ over which the similarity condition is expressed, and $\ddot{\theta}$, d, k and s_q as defined in Table I. Then*

$$\left(\ddot{\sigma}_{(S\ \ddot{\theta}(d,k)\ s_q)}\ T_1\right) \bigcap T_2 \ \subseteq \ \ddot{\sigma}_{(S\ \ddot{\theta}(d,k)\ s_q)}\ (T_1 \cap T_2) \ ; \tag{22}$$

$$\left(\ddot{\sigma}_{(S\ \ddot{\theta}(d,k)\ s_q)}\ T_1\right) \bigcap \left(\ddot{\sigma}_{(S\ \ddot{\theta}(d,k)\ s_q)}\ T_2\right) \ \subseteq \ \ddot{\sigma}_{(S\ \ddot{\theta}(d,k)\ s_q)}\ (T_1 \cap T_2)\,. \quad \square \tag{23}$$

For example, suppose that we want to show the 5 cities nearest to "New York city-NY", which belongs to the New York state and whose percentage of families in the poverty level percentage are smaller than 18.5. Example 4.7 summarizes this query.

EXAMPLE 4.7. *"Select the 5 cities nearest to "New York city-NY" that belong to the New York state and has the percentage of families in poverty level* PctPovertyFam$\leq$ 18.5*, considering Euclidean distance L_2".*

In similarity algebra, this query is written as:

$$\ddot{\sigma}_{(\texttt{Coordinate}\ \ddot{\theta}(L_2,5)\ Coord(\text{New York}))}\left(\sigma_{(State=NY)}\texttt{USCities} \bigcap \sigma_{(PctPovertyFam\leq 18.5)}\texttt{USCities}\right), \tag{24}$$

and returns 5 tuples {Kaser village-NY, Monsey CDP-NY, New Square village-NY, Verplanck CDP-NY, New York city-NY}. Applying the kNN selection over the first relation and executing its intersection with the second relation, we have:

$$\left(\ddot{\sigma}_{(\texttt{Coordinate}\ \ddot{\theta}(L_2,5)\ Coord(\text{New York}))}\sigma_{(State=NY)}\texttt{USCities}\right) \bigcap \sigma_{(PctPovertyFam\leq 18.5)}\texttt{USCities}, \tag{25}$$

which returns only the tuple {New York city-NY}. In this way, we see that the result of Equation 25 is included in the result of Equation 24. Moreover, applying the distributive property and executing the intersection over the kNN selection of each relation, the expression is rewritten as:

$$\left(\ddot{\sigma}_{(\texttt{Coordinate}\ \ddot{\theta}(L_2,5)\ Coord(\text{New York}))}\ \sigma_{(State=NY)}\ \texttt{USCities}\right) \ \bigcap \\ \left(\ddot{\sigma}_{(\texttt{Coordinate}\ \ddot{\theta}(L_2,5)\ Coord(\text{New York}))}\sigma_{(PctPovertyFam\leq 18.5)}\ \texttt{USCities}\right) \ , \tag{26}$$

which also returns only tuple {New York city-NY}. Thus, we see that the result of Equation 26 is also included in the result of Equation 24.

The relationship between kNN and the traditional cross product operator is presented in Property 4.8. For these operators, the result of the kNN executed over the cross product of relations is

included in the result of the cross product between the kNN selection executed in one relation and the other relation.

PROPERTY 4.8. *Let T_1 and T_2 be two relations, S in T_1 be a complex attribute taken in domain $\mathbb{S}$ over which the similarity condition is expressed, and $\ddot{\theta}$, d, k and s_q as defined in Table I. Then*

$$\ddot{\sigma}_{\left(S\ \ddot{\theta}(d,k)\ s_q\right)}\left(T_1\ \times\ T_2\right)\ \subseteq\ \left(\ddot{\sigma}_{\left(S\ \ddot{\theta}(d,k)\ s_q\right)}\ T_1\right)\ \times\ T_2\ .\ \square \tag{27}$$

There is a special case of cross product operator to treat the conjunction of selection predicates. If the condition is conjunctive and it can be rewritten as $S_1\ \ddot{\theta}\ s_{q_1}\ \wedge\ S_2\ \ddot{\theta}\ s_{q_2}$ where $S_i \in T_i$, then the result of the conjunctions of a kNN executed over the cross product of two relations is included in the result of the cross product between the kNN selection executed over the first relation and the kNN selection executed over the second relation.

PROPERTY 4.8.1. *Special case for the cross product: Let T_1 and T_2 be two relations, $S_1 \in T_1$ and $S_2 \in T_2$ be complex attributes taken in domain $\mathbb{S}$ over which the similarity condition is expressed, and $\ddot{\theta}$, d_1, d_2, k_1, k_2, s_{q_1} and s_{q_2} as defined in Table I. Then*

$$\ddot{\sigma}_{\left(S_1\ \ddot{\theta}(d_1,k_1)\ s_{q_1}\right)\wedge\left(S_2\ \ddot{\theta}(d_2,k_2)\ s_{q_2}\right)}\left(T_1 \times T_2\right)\ \subseteq\ \left(\ddot{\sigma}_{\left(S_1\ \ddot{\theta}(d_1,k_1)\ s_{q_1}\right)}\ T_1\right) \times \left(\ddot{\sigma}_{\left(S_2\ \ddot{\theta}(d_2,k_2)\ s_{q_2}\right)}\ T_2\right).\ \square \tag{28}$$

As Cartesian products are conceptually a component of the Join operators and usually have few intuitive applications on user queries, we postpone an example of this property for the next property.

Considering the kNN selection operation and the traditional join operator, the result of the join between the result of kNN selection operations executed in each relation is included in the result of the kNN executed over the join of both relations, as shown in Property 4.9.

PROPERTY 4.9. *Let T_1 and T_2 be two relations, $S\ \in\ T_1$ be a complex attribute taken in domain $\mathbb{S}$ over which the similarity condition is expressed, and $\ddot{\theta}$, d, k and s_q as defined in Table I. Then*

$$\left(\ddot{\sigma}_{\left(S\ \ddot{\theta}(d,k)\ s_q\right)}\ T_1\right)\ \bowtie\ T_2\ \subseteq\ \ddot{\sigma}_{\left(S\ \ddot{\theta}(d,k)\ s_q\right)}\left(T_1\ \bowtie\ T_2\right)\ .\ \square \tag{29}$$

To illustrate this property, suppose that we want to retrieve from the cities of the New York state, the 5 cities nearest to the "New York city-NY" whose percentage of families poverty level is smaller than or equal to 18.5. This query is summarized in Example 4.8.

EXAMPLE 4.8. *"Select the 5 cities nearest to "New York city-NY", which belong to the New York state and have the percentage of families in poverty level smaller than or equal to 18.5, considering Euclidean distance L_2".*

Algebraically, this query is expressed as:

$$\ddot{\sigma}_{\left(\texttt{Coordinate}\ \ddot{\theta}(L_2,5)\ Coord(\text{New York})\right)}\left(\sigma_{(State=NY)}\texttt{USCities} \bowtie \sigma_{(PctPovertyFam\leq 18.5)}\texttt{USCities}\right). \tag{30}$$

It returns the 5 tuples {`Kaser village-NY, Monsey CDP-NY, New Square village-NY, Verplanck CDP-NY, New York city-NY`}. Applying the kNN selection over the first relation and executing its join with the second one, we have:

$$\left(\ddot{\sigma}_{\left(\texttt{Coordinate}\ \ddot{\theta}(L_2,5)\ Coord(\text{New York})\right)}\sigma_{(State=NY)}\texttt{USCities}\right) \bowtie \sigma_{(PctPovertyFam\leq 18.5)}\texttt{USCities}, \tag{31}$$

which returns only the tuple {`New York city-NY`}. Thus we can see that the result of Equation 31 is included in the result of Equation 30.

There is also a special case of the join operator to treat the conjunction of selection predicates. If the condition is conjunctive and can be rewritten as $S_1\ \ddot{\theta}\ s_{q_1}\ \wedge\ S_2\ \ddot{\theta}\ s_{q_2}$, where $S_i \in T_i$, then the

result of the join between the kNN selection executed over the first relation and the kNN selection executed over the second relation is included in the result of the conjunction of the kNN executed over the join of relations.

PROPERTY 4.9.1. *Special case for the join operator: Let T_1 and T_2 be two relations, $S_1 \in T_1$ and $S_2 \in T_2$ be complex attributes taken in domain $\mathbb{S}$ over which the similarity condition is expressed, and $\ddot{\theta}$, d_1, d_2, k_1, k_2, s_{q_1} and s_{q_2} as defined in Table I. Then*

$$\left(\ddot{\sigma}_{(S_1\ \ddot{\theta}(d_1,k_1)\ s_{q_1})}\ T_1\right) \bowtie \left(\ddot{\sigma}_{(S_2\ \ddot{\theta}(d_2,k_2)\ s_{q_2})}\ T_2\right) \subseteq \ddot{\sigma}_{(S_1\ \ddot{\theta}(d_1,k_1)\ s_{q_1}) \wedge (S_2\ \ddot{\theta}(d_2,k_2)\ s_{q_2})}\ (T_1 \bowtie T_2)\,. \quad \square \quad (32)$$

5. EXPERIMENTAL EVALUATION

Our experiments were performed over SIREN, as it supports an extension of SQL to handle similarity queries [Barioni et al. 2006]. In this section we present experiments comparing SIREN executing queries rewritten using our proposed algebraic properties. SIREN is implemented in C++, and the experiments were evaluated using an Intel Core 2 Quad 2.83GHz processor with 4GB of main memory, under the Windows XP operating system. SIREN was configured to process the traditional part of the queries in Oracle 9i. Due to space limitations, we only report here the performance regarding total time (in milliseconds), as it summarizes the whole computational cost. As our objective here is also to show the effect of using the set inclusion-based properties on the intuitive query results and not only its contribution to improve query performance, we do not perform tests on synthetic, controlled databases. Rather, we employ two real world publicly available data sets, as follows:

—*USCities*: a set of 25,374 American cities and their economic characteristics in Census 2000, obtained from U.S. Census Bureau website[1]. They were compared using the Euclidean distance function.

—*DDSM*: a set of 4,612 mammography images, obtained between 1993 and 1999 from the Digital Database for Screening Mammography (DDSM) website[2] [Heath et al. 1998][Heath et al. 2000]. They were compared using the texture distance function [Haralick et al. 1973].

In our experiments, we apply the canonical and the alternative plans to execute the following queries:

Q1: In a Geographic Information Systems: "*Find the 5 cities nearest to 'New York city-NY', whose distances from 'Albany city-NY' are not farther than 210 km, considering the Euclidean distance L_2, having the per capita income greater than 22,400 and the percentage of families in poverty level smaller than or equal to 18.5*".

Q2: In a health-care information system: "*Select the 3 mammographies taken in 1993 that are the most similar to this one from my current patient (Patient X), whose patient is less then 45 years old and the exam was done in Massachusetts General Hospital (MGH)*".

Query **Q**1 involves traditional, similarity *range* and kNN selections. It can be algebraically expressed as:

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

In Figures 1 and 2, we present the canonical query plan resulting from the original query and the alternative execution plan that results from query rewriting, for Queries **Q**1 and **Q**2, respectively. Both figures also show the number of expected (k) and returned results for each plan, and their

[1]U.S. Census Bureau Homepage. Last access in: 2011 May 15. Available at: `http://www.census.gov/`

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

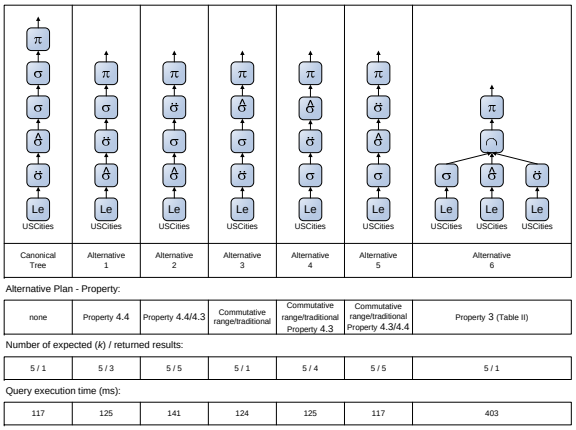

Fig. 1. Canonical, alternative plans and execution time of Query **Q**1.

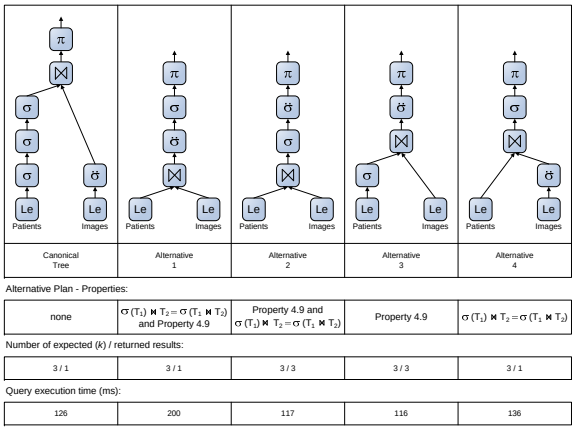

Fig. 2. Canonical, alternative plans and execution time of Query **Q**2.

evaluation time in milliseconds (ms). The time reported corresponds to the average execution of 10 queries like **Q**1 and **Q**2, respectively.

Considering Query **Q**1, Figure 1 shows the canonical and six alternative execution plans for this query. Although the evaluation time of the Canonical and the Alternative 5 plan is the same, the canonical plan can return less than k tuples, as further operations are applied over the first k tuples selected, pruning more results. When the evaluation of the predicates of the other selections return at least k tuples, executing the kNN as the last operation (Alternative 5) warrants that the asked amount k of tuples are returned. Moreover, these those k tuples are always returned by the canonical plan. In Alternative 5 of Figure 1, we rewrote the canonical plan applying the commutative property between range and traditional operators, Property 4.3, and Property 4.4, respectively.

Query **Q**2 involves both traditional and kNN selection, as well as a traditional join. It can be expressed as:

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

The canonical and four alternative plans for this query are presented in Figure 2. Alternative plans 3 and 2 have a gain about 8% comparing with the canonical tree. Analogous to Query **Q**1, alternative plans 3 and 2 both return k tuples. In the Alternative plan 2 and 3, we apply the $\sigma(T_1) \bowtie T_2 = \sigma(T_1 \bowtie T_2)$ property and Property 4.9 over canonical plan to query rewrite, as shown in Figure 2.

6. DISCUSSION

It is expected that the canonical query plan generated by the DBMS interpreter has the sequence of select operators in the same original query plan sequence. As the query optimizer module further rearranges this ordering based on the commutative property of the traditional predicates, this is not a issue at all. However, kNN predicates are not commutative. Thus, when SQL is extended to support similarity queries, the corresponding interpreter must have a rule about how to position the kNN selections regarding the other operators. It has been well-accepted that the kNN predicates should be the first to be executed, on behalf that in this way a similarity index existing on the query predicate attributes could be employed, thus helping to speed up the most costly part of the query execution. This choice is a pragmatic one, as the lack of the commutative property denies that alternative choices result in the same answer, and it is supposed that it can obtain the fastest execution. The

experiments we showed in the previous section highlight that this choice does not stand when our new set inclusion-based properties are taken into account.

In fact, the current choice of executing first the kNN predicates usually leads to fewer tuples in the result than specified in the kNN arguments. This is due to the fact that executing the other predicates after the kNN filters out part of the k tuples selected (or even all the tuples). To allow returning the same number of tuples asked (at least when just one kNN predicate exists in the query), the kNN predicate should be executed last. Our experiments show that the alleged lack of performance induced by this choice does not occur in practice. In fact, empowering the query optimizer with the set inclusion-based properties presented in this article, it is possible to find query execution plans that are at least as fast as those that execute first the kNN predicate, yet returning the desired number of tuples – provided they exist in the database.

A final remark about including operations in the DBMS execution core. Although similarity indexes are not used if a kNN selection is not the first operator executed, this does not preclude executing kNN indexed access: it remains useful to answer kNN only queries, including simple queries, when there is no way to employ traditional attribute indexes.

7. CONCLUSION

One of the most interesting features of Relational Database Management Systems is their ability to handle the algebraic expression of a query, in order to evaluate the costs of several access plans to choose the one that is probably the fastest. This feature requires that the DBMS optimizer module knows the algebraic properties of the query operators, in order to derive the alternative plans. The kNN is one of the more costly predicates, so it should be one that better took advantage of this DBMS feature, but unfortunately the kNN predicate has too few equivalence properties to allow good query optimizations. It even does not meet the commutative property.

In this article we presented a complete set of properties for the kNN predicates based not on equivalence, but in set inclusion, which greatly enlarges the collection of algebraic properties that the DBMS can count on to optimize compound queries involving at least one kNN predicate. We also presented several examples of how queries can be rewritten using those properties, and evaluated compound queries involving the kNN predicate placed over two databases having relations that are composed of attributes able to be queried by similarity. For both databases, we show that the optimizer can in fact choose interesting alternative plans.

The lack of the commutative property for the kNN predicate has implications on how the DBMS interpreter handles query commands that include kNN predicates. It is well accepted by several researchers in the database community that the kNN predicates should be executed first, allowing to employ existing similarity-based indexes, thus helping to speed up the most costly part of the query execution. However, this predicate ordering almost always retrieves less tuples than asked in the query. Our experiments using the set inclusion-based properties show that it is possible to execute the kNN as the last predicate in a conjunction of select operators, and yet obtain the answer in the same time, or even faster, than executing first the kNN predicate, allowing to obtain the asked number of tuples.

REFERENCES

Adali, S., Bonatti, P., Sapino, M., and Subrahmanian, V. A multi-similarity algebra. In *Proceedings of the ACM SIGMOD International Conference on Management of Data Conference*. Seattle, USA, pp. 402–413, 1998.

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

Adali, S., Sapino, M. L., and Marshall, B. A rank algebra to support multimedia mining applications. In *Proceedings of the International Workshop on Multimedia Data Mining*. San Jose, USA, pp. 1–9, 2007.

Atnafu, S., Brunie, L., and Kosch, H. Similarity-based algebra for multimedia database systems. In *Proceedings of the Australasian Database Conference*. Gold Coast, Australia, pp. 115–122, 2001.

Atnafu, S., Chbeir, R., Coquil, D., and Brunie, L. Integrating similarity-based queries in image DBMSs. In *Proceedings of the ACM Symposium on Applied Computing*. Nicosia, Cyprus, pp. 735–739, 2004.

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

Barioni, M. C. N., Razente, H. L., Traina, A. J. M., and Traina-Jr., C. SIREN: A similarity retrieval engine for complex data. In *Proceedings of the International Conference on Very Large Data Bases*. Seoul, South Korea, pp. 1155–1158, 2006.

Bartolini, I. *Efficient and Effective Similarity Search in Image Databases*. Ph.D. thesis, University of Bologna, 2002.

Belohlávek, R., Opichal, S., and Vychodil., V. Relational algebra for ranked tables with similarity: properties and implementation. In *Proceedings of the International Symposium on Intelligent Data Analysis*. Ljubljana, Slovenia, pp. 140–151, 2007.

Berchtold, S., Böhm, C., Keim, D. A., Krebs, F., and Kriegel, H.-P. On optimizing nearest neighbor queries in high-dimensional data spaces. In *Proceedings of the International Conference on Database Theory*. London, UK, pp. 435–449, 2001.

Böhm, C., Braunmüller, B., and Kriegel, H.-P. The pruning power: A theory of scheduling strategies for multiple k-nearest neighbor queries. In *Proceedings of the Data Warehousing and Knowledge Discovery*. Greenwich, UK, pp. 372–381, 2000.

Chbeir, R. and Laurent, D. Towards a novel approach to multimedia data mixed fragmentation. In *Proceedings of the International ACM Conference on Management of Emergent Digital EcoSystems*. Lyon, France, pp. 200–204, 2009.

Ciaccia, P., Montesi, D., Penzo, W., and Trombetta, A. Imprecision and user preferences in multimedia queries: a generic algebraic approach. In *Proceedings of the International Symposium on Foundations of Information and Knowledge Systems*. Burg, Germany, pp. 50–71, 2000.

Falchi, F., Lucchese, C., Orlando, S., Perego, R., and Rabitti, F. A metric cache for similarity search. In *Proceedings of the ACM Workshop on Large-Scale Distributed Systems for Information Retrieval*. Napa Valley, USA, pp. 43–50, 2008.

Faloutsos, C. Indexing of multimedia data. In *Multimedia Databases in Perspective*. Lecture Notes in Computer Science. Indexing of Multimedia Data, pp. 219–245, 1997.

Ferreira, M. R. P., Traina, A. J. M., Dias, I., Chbeir, R., and Traina Jr., C. Identifying algebraic properties to support optimization of unary similarity queries. In *Proceedings of the Alberto Mendelzon International Workshop on Foundations of Data Management*. Vol. 450. Arequipa, Peru, pp. 1–10, 2009.

Haralick, R. M., Shanmugam, K., and Dinstein, I. Textural features for image classification. *IEEE Transactions on Systems, Man, and Cybernetics* 3 (6): 610–621, 1973.

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

Heath, M., Bowyer, K., Kopans, D., Moore, R., and Kegelmeyer-Jr., P. The digital database for screening mammography. In *Proceedings of the International Workshop on Digital Mammography*. Toronto, Canada, pp. 212–218, 2000.

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

Montesi, D. and Trombetta, A. Similarity search through fuzzy relational algebra. In *Proceedings of the International Conference on Database and Expert Systems Applications*. Florence, Italy, pp. 235–239, 1999.

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

Picariello, A. and Sapino, M. L. A fuzzy algebra for image data bases. In *Proceedings of International Workshop on Multimedia Information Systems*. Tempe, USA, pp. 86–95, 2002.

Schmitt, I. and Schulz, N. Similarity relational calculus and its reduction to a similarity algebra. In *Proceedings of the International Symposium on Foundations of Information and Knowledge Systems*. Wilhelminenburg Castle, Austria, pp. 252–272, 2004.

Silva, Y. N. and Aref, W. G. Similarity-aware query processing and optimization. In *Proceedings of the International Conference on Very Large Data Bases PhD Workshop*. Lyon, France, pp. 1–6, 2009.

Traina-Jr., C., Traina, A. J. M., Vieira, M. R., Arantes, A. S., and Faloutsos, C. Efficient processing of complex similarity queries in RDBMS through query rewriting. In *Proceedings of the International Conference on Information and Knowledge Engineering*. Arlington, USA, pp. 4–13, 2006.