Embedding k-Nearest Neighbor Queries into Relational Database Management Systems

Gabriel V. De Pierro, Mônica R.P. Ferreira, Lucio F.D. Santos, Willian D. Oliveira, Agma J. M. Traina, Caetano Traina Jr.

Computer Science Dept., ICMC-Univ of São Paulo, São Carlos-SP, Brazil
{gdpierro,monika,luciodb,willian,agma,caetano}@icmc.usp.br

Abstract.

The ability to use, in a single query, both similarity and traditional select operators inside a Relational Database Management System (RDBMS) has proven to be a valuable tool, which provides a strong capability for better contextualizing queries involving complex data. However, the integration of both query operator classes is still in its early stages, as most works on similarity queries focus on the data structures and operations required to execute just the similarity retrieval operation, but few of them target problems on the integration itself. Among them, the k-Nearest Neighbor (k-NN) comparison operator gives ground to conflicting interpretations when composed with others. In this article we present a novel approach to remove dubiety when similarity-selections based on k-NN are embedded into a RDBMS, formalize the possible interpretations, and show how to resolve implementation issues that occur when executing them on database relations that include attributes queried by similarity. We also implemented a prototype for a RDBMS that also supports queries employing k-NN predicates and evaluated the proposed interpretations, measuring the performance of their respective operators, as well as the semantic implications. The proposed solution is versatile, being able to handle queries combining similarity-based and traditional predicates.

Categories and Subject Descriptors: H.2.4 [**Database Management**]: Systems; H.3.3 [**Information Systems**]: Information Search and Retrieval

Keywords: k-Nearest neighbors, Relational model, Similarity algebra, Similarity search

1. INTRODUCTION

The ever increasing computational processing power and broadband capability has consequently brought forth an ongoing rise in data volume and complexity. Complex data types, such as multimedia, time-series, rich text and geo-referenced information, have seamlessly blended into the common every-day life.

However, in spite of the new paradigms provided by the increasing data complexity, the queries that involve traditional data (i.e. numeric values, small strings of characters and dates) are still of utmost importance, as the complex data only increase the field of possibilities, but do not replace queries over traditional data. Furthermore, there is plenty of evidence in the literature that it is possible to achieve better results through the "contextualization" of information [Figueiredo et al. 2013; Serra et al. 2011; Koopman et al. 2012], allowing not only for more meaningful results by combining traditional and complex data into a single query, but also augmenting query precision by allowing end-users to create more expressive queries using both data types [Ferreira et al. 2010].

In this context, the relational database management systems (RDBMS) are still lagging behind as, while being exceptional tools to deal with traditional data, there has not been a significant break-

This work was supported by FAPESP, CAPES and CNPq.

through able to unleash these systems to properly and homogeneously handle complex data types. Part of the problem is that the research efforts focus on the similarity retrieval side, but few target its integration with identity and Total Order Relationship (TOR) operators into the RDBMS execution environment and with the relational model itself. However, both from a semantic and a performance point of view, it is important to treat the two operator types as equally meaningful during all stages of the query processing, including providing conceptual tools for the optimization of the query execution plans.

As we show in this article, there are several challenges that this integration brings to light:
—from the semantic perspective, how to express the meaning expected from melting similarity within and between tuples;
—from the syntactic perspective, how to express the precise intention when integration must occur, in the relational model; and
—from a more pragmatic view, how complex queries involving identity, TOR and similarity operators can be executed efficiently within the RDBMS environment, and how they provide efficient models to retrieve complex data.
These challenges arise from the fact that, in contrast to the identity- and TOR-based traditional operators, which evaluates to `true` or `false` based solely on the compared attribute values in the involved tuples, it exists similarity operators that depend upon all the involved attribute's active domains.

In the relational model, a database relation (or relation state) is a set of tuples whose schema is denoted by $(\mathsf{A}_1, \ldots \mathsf{A}_n)$, where $\mathsf{A}_1, \ldots \mathsf{A}_n$ are the relation attributes and each attribute A_i is said to be the role played by its domain $\mathbb{A}_i$ in the relation, that is, $Dom(\mathsf{A}_i) = \mathbb{A}_i$. The attribute's active domain $A = D^{*}om\,(\mathsf{A})$ is the subset of all the values $t[\mathsf{A}] \in \mathbb{A}$ that occur in at least one tuple of relation T.

In this article we assume that the attributes can have both scalar or complex domains. To distinguish attributes traditionally supported in RDBMS from those that can be compared by similarity, we call the latter complex attributes. A relation T can have any number of either scalar or complex attributes.

We reserve the symbol E to represent the scalar attributes, compared by identity and TOR, and the symbol S to refer to complex attributes compared by similarity. The symbol A is employed to mean either complex or scalar attributes. Corresponding to attribute A, whose domain $Dom(\mathsf{A}) = \mathbb{A}$ and active domain $D^{*}om\,(\mathsf{A}) = A$, we use the symbols $\mathsf{E}, \mathbb{E}$ and E for scalar attributes, and the symbols $\mathsf{S}, \mathbb{S}$ and S for complex attributes. Hence, the relation schema can be denoted as $(\mathsf{E}_1, \ldots \mathsf{E}_q, \mathsf{S}_1, \ldots \mathsf{S}_p)$. Finally, considering that T is a set of tuples, a tuple $t = \langle e_1, ..., e_q, s_1, ..., s_p \rangle \in \mathsf{T}$ has each value $e_{ih} = t_i[\mathsf{E}_h], (1 \leqslant h \leqslant q)$ obtained in domain $\mathbb{E}_h$ and each value $s_{ig} = t_i[\mathsf{S}_g], (1 \leqslant g \leqslant p)$ obtained in the domain $\mathbb{S}_j$.

We define that an attribute S is complex if it is associated to a distance function $d : \mathbb{S} \times \mathbb{S} \to \mathbb{R}^{+}$ defined on the attribute domain $\mathbb{S}$ that measures the similarity (or distance) among any pair of elements in $\mathbb{S}$. Therefore, an attribute A_i of a scalar type that is associated to a distance function is also complex and can be employed either in similarity or in TOR-based comparison operators. In this article we are interested in studying k-NN selection on complex attributes S, regardless of whether it is able to be queried also by identity and TOR.

The two most common similarity-based comparison operators are the similarity range (Rng) and the k-nearest neighbor (k-NN) comparison. Both are applied on a complex attribute S sampling from the active domain $S \in \mathbb{S}$, which meets a similarity criteria defined with respect to a data element $s_q \in \mathbb{S}$, called the "query center" [Hu and Lee 2006]. A query using the similarity range comparison Rng, usually known as a range query (Rq), returns the elements closer than or at the same distance d from the query center. Likewise, a query using the k-nearest neighbor operator k-NN, known as a k-nearest neighbor query (k-NNq), returns the k elements nearest to the query center in the active domain $S = D^{*}om\,(\mathsf{S})$ of attribute S. To be able to execute those operators with a precise meaning in

an efficient manner, it is necessary to move away from the purely tuple-based comparison operators and acknowledge the requirements of domain-based comparison operators. Furthermore, it is also required to allow representing the complex data into metric spaces, where similarity comparisons replace the traditional TOR paradigm.

In this article, we propose to solve some of the problems originating from the inclusion of similarity queries and the retrieval of complex data types under the relational model, aiming at taking a first step to treat queries involving complex and traditional data types homogeneously inside the RDBMS. We are particularly interested in the k-NN comparison operator, as it brings the majority of the integration problems, and because easing the integration problems for k-NN naturally solves the Rng operator too. In order to achieve this objective, it is necessary to effectively define new algebraic operators for the relational model, extending the algebraic rules and corresponding algorithms to support similarity search, as most of its components (indexing structures, query optimizers, selectivity estimators) can only process data that meet the TOR properties. We target specifically the following problem:

—*When dealing with a relational tool that accepts complex attributes as tuple values, the k-NN operator leads to two distinct interpretations. This occurs because whereas, in a conceptual basis, a complex attribute domain forms a metric space, in a practical application several tuples in the database can assume the same complex attribute value. Therefore, a comparison operator whose answer depends on the whole active domain will provide distinct answers for distinct input relations, even if the tuple values do not change. As a consequence, a k-NN query can return a number $n \geqslant k$ of tuples even when only k different attribute values are retrieved. This dubiety leads to the question of what result should the query actually return, i.e. "what does the user expect as the result?" The k-NN query must be well defined, in order to express whether k tuples or k distinct complex attribute values are expected.*

To illustrate this problem of interpretation, take as an example an art gallery that sells copies of paintings and logs every sale on a RBDMS in the following relation:

`ItemSold={Item, SaleDate, ClientName, PaintingName, ArtistName, PaintingPicture}`

where `PaintingPicture` is a complex attribute storing a picture of the painting sold, and therefore is liable to be searched by similarity, such as by a k-NNq selection. Consider the following query thrown over this relation:

—"*$Q1$ - Which are the items sold that have their picture among the 5 most similar to picture s_q?*"

$Q1$ exemplifies employing a traditional k-NN operator, contextualized into a relational environment. Complex data are stored as attribute values of the relations, just like traditional attributes. Therefore, `PaintingPicture` is an attribute (of a complex type) in relation `ItemSold` and the query requests the tuples whose picture is among the 5 closest to the query center s_q. Notice that despite being a k-NNq selection with $k = 5$, the actual result set may have $n \geqslant k$ tuples, with n depending on the amount of repeated items sold, which is irrelevant for the query.

However, when the user wants the k tuples that have the most similar pictures to his/her query center s_q, he would ask:

—"*$Q2$ - Which are the 5 items sold whose pictures are the most similar ones to picture s_q ?*"

The query is still an application of the k-NN comparison, but now within a tuple-sided perspective, i.e. the operator should consider k as the number of resulting tuples, and not of complex attribute values. This second perspective is complementary to the one corresponding to query $Q1$. Notice that

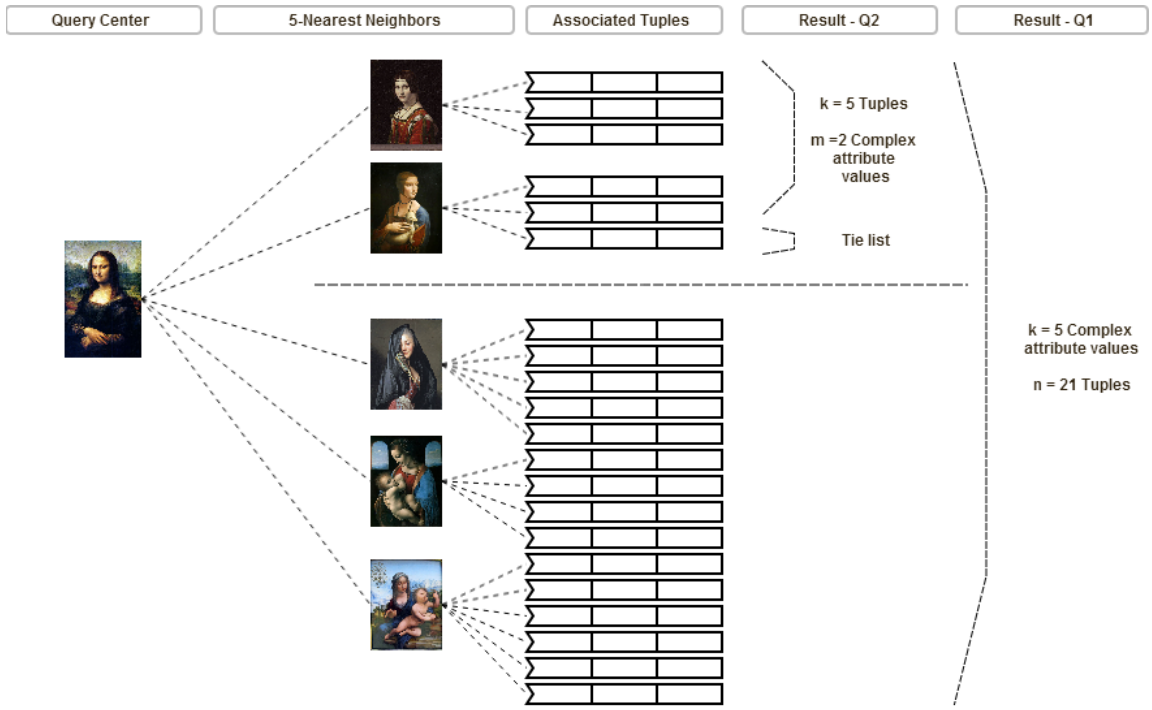

Fig. 1. A visual representation of the two proposed interpretations of the k-NN operator using an hypothetical art gallery database and having the query center s_q as a picture of the "Mona Lisa". The labels on top indicate what is represented in each step.

the two perspectives only bear out as a consequence of contextualizing the k-NN operator into the relational model, although up to now this point did not have a proper study or representation.

To further illustrate both interpretations, Figure 1 shows a visual representation of a possible answer to $Q1$ and $Q2$ with the query center s_q being a picture of the "Mona Lisa". As it can be seen, both interpretations can have highly different cardinalities depending on the nature of the dataset. Notice also that $Q1$'s answer contains $Q2$'s.

With the objective of fully including similarity search operations in a RDBMS engine, giving them the same status as the identity and TOR comparisons and homogenizing the processing of both, we present two retrieval operators and the corresponding interpretations for the k-NN comparison operator when embedded into a RDBMS, as well as an extension for the SQL language, able to represent queries on either interpretation. Our experimental results show that both interpretations have close execution times while keeping apart the cardinality of the answer set and the comparison parameters.

To include both interpretations in an extended SQL language that supports similarity operators, we start assuming the syntax of the SIREN engine presented in Barioni et al. [2009], that already provides a basic representation for similarity operators. Also, as suggested by Ferreira et al. [2011], we adopt the symbol $\ddot{\sigma}$, instead of the regular σ, to represent the selection operator using the k-NN similarity criteria, as the k-NNq -based selection operator does not meet the same properties as the regular σ selection operator. The algebraic properties of this operator were thoroughly studied in [Ferreira et al. 2009] and [Ferreira et al. 2011]. Another variation for the k-NNq -based selection operator also had its properties already studied in the literature [Silva et al. 2013].

The remainder of this article is structured as follows: Section 2 describes related work and fundamental concepts. Section 3 defines an extended relational model aiming at allowing similarity-based

queries and operators in a homogeneous way. Section 4 describes our novel interpretation for similarity operators integrated into the extended relational model. Section 5 describes supporting experimental results. Finally, Section 6 concludes the article.

2. FUNDAMENTAL CONCEPTS AND RELATED WORK

2.1 Fundamental Concepts

When dealing with complex data, TOR and identity comparisons usually do not hold a significant semantic value. In videos or images, for example, identity would almost never happen and TOR comparisons have very little significance [Faloutsos 1996]. In fact, comparisons in complex data domains should contemplate *similarity*. This can be achieved by extracting a numeric *"feature vector"* to represent some characteristic of the objects, thereafter comparing those features instead of the raw data directly, yielding more significant results [Faloutsos 1997]. Comparisons of feature vectors are performed through a *"distance function"*, i.e. the two feature vectors compared and the resulting value represents their similarity - or, more accurately, dissimilarity - as distances measuring how far apart two elements are. The pair $\langle set\_of\_feature\_vectors, distance\_function \rangle$ forms a metric space.

The SQL language, despite being at the same time simple and powerful, allowing a broad spectrum of query formulations, so far does not meet the requirements for handling complex data properly, as it does for traditional ones, being highly entangled with the TOR and identity paradigms of data comparisons. In fact, it assumes that the data values have an inherent order and thus the comparison operations, the index structures and, much by consequence, the query syntax and processing are straightforwardly bonded to identity and TOR.

In contrast, when dealing with complex data, to perform comparisons, it is first necessary to establish a metric by which two objects are being evaluated and the SQL needs to properly handle these comparisons, as well as the association of metrics to each object. Moreover, traditional selection predicates on complex elements are seldom useful. Thus, it is necessary for SQL to represent the useful predicates for complex data types, the most common being the range query (Rq) and the k-nearest neighbor query (k-NNq), as presented in Section 1.

Aiming at filling the gap between scalar and complex data in SQL, Barioni et al. [2005] developed an SQL extension and respective interpreter, the similarity retrieval engine (SIREN) [Barioni et al. 2006]. It was implemented as a middleware to enhance the capabilities of regular RDBMS with similarity-based resources. SIREN stands between the application software and the RDBMS, interpreting and processing the complex data- and similarity-related parts of each query command, and relaying to the underlying RDBMS the execution of the traditional parts of the query. It uses a metric access method (MAM), more specifically the slim-tree [Traina Jr. et al. 2002], to execute the similarity-based predicates. SIREN's syntactical query structure follows the same pattern of traditional queries, only enlarged with the following similarity-based predicate:

$$\langle attr \rangle NEAR \langle val \rangle \texttt{ [STOP AFTER } < k > \texttt{] [RANGE } < \xi > \texttt{]}$$

where `NEAR` indicates a similarity comparison operator, $\langle attr \rangle$ is a complex attribute and $\langle val \rangle$ is an element from the same complex domain of the attribute. The two optional arguments `[ STOP AFTER` $< k >$ `]` and `[ RANGE` $< \xi >$ `]` specify if the query is respectively a Rq or a k-NNq , as well as their corresponding threshold.

As an example, Query ***Q1*** presented in Section 1 is written in SIREN SQL as:

```
SELECT Item
   FROM ItemSold
   WHERE PaintingPicture NEAR s_q STOP AFTER 5;
```

where `NEAR` s_q `STOP AFTER 5` indicates the k-NNq query using the k-NN comparison operator with $k = 5$ and s_q is the target image, indicated either as a file in the computer system or as the result of a subselect to retrieve an image already stored in the database.

SIREN allows comparison predicates on complex data only when the attribute storing them is assigned to a metric. This is performed when the attribute is declared in the `CREATE TABLE` command as an attribute of complex type, which requires it to be associated to a metric. For example, the following command creates a table with a scalar attribute `Item` of type `INT` and a complex attribute `PaintingPicture` of type `STILL_IMAGE` whose similarity is measured by metric `Color`:

```
CREATE TABLE ItemSold (
 Item INT,
 PaintingPicture STILL_IMAGE METRIC(Color))
```

Metrics must be previously created, using the SIREN's `CREATE METRIC`, and can be further manipulated with the `ALTER METRIC` and `DROP METRIC` commands.

SIREN and its extended SQL adequately fulfill implementation requirements to process complex data. More detailed information about SIREN and the extended SQL can be found at Barioni et al. [2009]. However, having a RDBMS which allows representing similarity and traditional comparison operators in a single query highlights that several semantic and modeling issues are still open. Specifically, it is always assumed that the `[ STOP AFTER` $< k >$ `]` construction means that k tuples must be returned from the corresponding select operator, and there is no way to specify whether k distinct values or k tuples is the expected answer. Moreover, SIREN enables optimization for queries that use similarity and traditional predicates, as its syntax provides room for both to be intermixed in a single command, and the execution does not consider each predicate as being isolate, but the query as a whole. However, SIREN does not, initially, perform the optimizations, leaving open questions about how to perform them.

2.2 Related Work

The literature presents works regarding the integration of similarity-based and traditional predicates, which we refer to here as hybrid queries. The works have a twofold focus, emphasizing either integration as an integral part of the relational model and the RDBMSs with their underlying algebra, or from the algorithm point of view.

Also, some works take a more practical approach, seeking to optimize how hybrid queries are executed and offering new paradigms on how queries are processed, while others take a more semantic-oriented approach. The former approach focuses on the implementation of RDBMSs that combine both types of predicates, whereas the later aims at answering questions such as: "How should similarity be represented in the relational model?". Often, both sides of the problem are looked into, but usually weighing more towards one side. The idea of extending the relational algebra with similarity was pioneered by Adali et al. [1998], which proposed the multi-similarity algebra (MSA). From there, several propositions were made both at embedding similarity into the relational algebra or taking more algorithmic approaches.

Some works specifically intend to optimize similarity query processing inside RDBMS, identifying algebraic properties of similarity operators, and creating techniques and algebraic rules to rewrite queries [Silva et al. 2013; Ferreira et al. 2009; Traina Jr. et al. 2006]. From those approaches, a notable point to be highlighted is that, while the Rq selection operator behaves like the traditional selection, the k-NNq selection operator does not. As a consequence, it is necessary to review its interaction with every operator of a RDBMS. In fact, the k-NNq has few equivalence properties, and it does not even meet the commutative property. Therefore, independently of how costly it is, handling them inside a RDBMS becomes much more challenging, as the order of the predicates change

the query results.

Silva et al. [2010] presents a similarity-aware database management system, extending the PostgreSQL RDBMS and leveraging its capabilities, coupled with transformation rules, to support the implementation and query-plan optimization of queries with multiple similarity-aware operators and traditional operators. In [Silva et al. 2013] the authors extend this idea, elaborating on how similarity-aware database operators should be treated in the context of query processing where multiple operators are involved, aimed at allowing for proper evaluation and optimization of complex similarity queries that involve multiple similarity operators such as selection, join and group-by.

In Budíková et al. [2012], the authors provide an extension to the SQL language that complements the MESSIF framework [Batko et al. 2007], presenting the necessary interface for complex queries with both similarity and traditional predicates. Moreover, the authors raise several important points regarding the possible and necessary requirements for similarity queries. However, they do not investigate the representation of similarity operations into the relational model or RDBMS, taking the middleware approach and focusing more on similarity searching on metric spaces and not on the possible outcomes of the integration with the concepts of tuples and relations. In fact, MESSIF is a self-contained framework for similarity search, and does not specifically target the RDBMS integration. The proposed SQL extension, albeit providing an interface for queries composed of both complex and traditional data, is presented as a theoretical tool waiting for a future implementation.

More specific solutions focus on resolving a specific problem, or handling certain datatypes and applications like content-based image retrieval (CBIR). In this context there are works such as PostgreSQL-IE [Guliato et al. 2009], which extends PostgreSQL to enable the treatment of images and CBIR. Another one is FMI-SiR [Kaster et al. 2010], which leverages the Oracle database and its interMedia package for media manipulating capabilities, improving on the existing capabilities of Oracle and extending them. FMI-SiR is also specialized into MedFMI-SiR [Kaster et al. 2011], to specifically handle medical images and their particularities.

Other popular approaches are rank queries, fuzzy-based queries, and the combination of both. Regarding fuzzy techniques, the work of Penzo [2005] expands on the similarity algebra $SAME^W$ [Ciaccia et al. 2000], allowing for a mapping from complex to SQL queries to be done applying rewriting rules. Li et al. [2005] propose RankSQL, a rank-based extension of the relational algebra, aiming at treating *top-k* as "first-class" queries within a RDBMS. Belohlávek et al. [2007] propose a rank-based extension to the relational model, also combining to added uncertainty and imprecision (i.e. fuzziness) to address imprecise queries within RDBMS. This idea is further expanded in [Belohlávek and Vychodil 2013], where the authors propose a multi-ranked model, assigning ranks to tuple values, instead of the tuples as a whole.

All of these works deal only with datasets without duplicated values. This issue is treated by our new technique presented here.

3. SIMILARITY QUERIES IN AN EXTENDED RELATIONAL MODEL

To include complex attributes among those handled by the relational algebra requires the revision of the similarity operators that have been traditionally developed to query them, as the queries now need to focus on the tuples, instead of on just a single complex attribute at a time. Although this seems a natural step towards integrating them into RDBMS, it was not done before.

In Section 1 we presented the two types of similarity queries most frequently studied in the literature: the range query Rq and the k-nearest neighbor query $k\text{-}NNq$. A range query can be expressed using a select operator $\boxed{\sigma_{(\mathsf{S}\ Rng(d,\xi)\ s_q)}\mathsf{T}}$, where S is a complex attribute and Rng is the range comparison operator, which returns `true` for every tuple $t \in \mathsf{T}$ such that $d(t[\mathsf{S}], s_q) \leqslant \xi$. A range query does not lead to ambiguity in its interpretation, because obtaining "every element closer than the given range

to s_q and selecting all the tuples that have any of these elements as values of attribute S" generates a set equal to the one composed of "all tuples whose S value is closer than the given range to s_q". Moreover, all the properties of the algebraic select operator σ hold when the comparison operator employed in the selection is the Rq comparison operator. For instance, the very useful property of commutativity holds, thus

$$\sigma_{(\mathsf{S}_a\ Rng(d_a,\xi_a)\ s_{qa})}\left(\sigma_{(\mathsf{S}_b\ Rng(d_b,\xi_b)\ s_{qb})}\mathsf{T}\right) \equiv \sigma_{(\mathsf{S}_b\ Rng(d_b,\xi_b)\ s_{qb})}\left(\sigma_{(\mathsf{S}_a\ Rng(d_a,\xi_a)\ s_{qa})}\mathsf{T}\right),$$

also holds for any complex attribute S_a and S_b, distance function d_a and d_b, query radius ξ_a and ξ_b and query center s_a and s_b.

The k-nearest query can not be expressed using only a select operator. In fact, if one try to use the k-nearest neighbor comparison operator in a selection, as in $\ddot{\sigma}_{(\mathsf{S}\ k\text{-}NN(d,k)\ s_q)}\mathsf{T}$, the resulting operator does not hold many properties of a selection operator. For instance, it does not hold the commutativity property, thus

$$\ddot{\sigma}_{(\mathsf{S}_a\ k\text{-}NN(d_a,k_a)\ s_{qa})}\left(\ddot{\sigma}_{(\mathsf{S}_b\ k\text{-}NN(d_b,k_b)\ s_{qb})}\mathsf{T}\right) \not\equiv \ddot{\sigma}_{(\mathsf{S}_b\ k\text{-}NN(d_b,k_b)\ s_{qb})}\left(\ddot{\sigma}_{(\mathsf{S}_a\ k\text{-}NN(d_a,k_a)\ s_{qa})}\mathsf{T}\right).$$

Therefore, given that the selection of k-nearest neighbors does not possess the same properties as the select operator, a variant must be introduced to the algebra as a new derived operator, in order to adequately represent it and its distinct set of properties. Furthermore, the k-nearest neighbors variation also leads to an ambiguity in the interpretation of the queries, because obtaining "the k attribute values closer to s_q and selecting all tuples that have any of these elements as values of attribute S" generates a set, which is different from the set composed of "all k tuples whose S value are the closest to s_q". Therefore, we propose that the k-nearest neighbor comparison operator k-NN generates in fact two select variants: it may return every tuple $t \in \mathsf{T}$ such that value $t[\mathsf{S}]$ is one of the k nearest to s_q; or it may return the k tuples $t \in \mathsf{T}$ such that value $t[\mathsf{S}]$ is one of the nearest to s_q.

As the two operators are derived from the fundamental algebraic properties, they do not change the Relational Algebra's expressive power, but rather they remove ambiguities in the query interpretation and make it easier to express the queries.

Moreover, they share the same algebraic properties already studied in the literature for the selection operators based on $\ddot{\sigma}$ comparisons.

Notice that if attribute S is a key in T, then both interpretations are equivalent. The works presented so far in the literature always assume that each tuple has a value distinct for each tuple, that is, S is a key in T. Therefore, the interpretation ambiguity has not been studied yet in the literature, as revised in Section 2. However, in real applications, it is common that more than one tuple have the same value in a complex attribute. In the following section we detail our proposed approach to handle both cases in the relational model and an extension in SQL to represent queries that unambiguously express either interpretations.

4. EMBEDDING THE K-NN COMPARISON OPERATOR INTO THE RELATIONAL MODEL

Under the perspective of handling complex attributes in similarity retrieval operations as presented in Section 3, we establish a novel way to integrate similarity into a RDBMS, but it leads to two distinct interpretation for the k-NN comparison operator. Let us assume that a k-selection returns a result set $\mathsf{T}_R \subset \mathsf{T}$ such that $n = |\mathsf{T}_R|$ is the cardinality of the result, and that $m = \left|\pi_{\{\mathsf{S}\}}\mathsf{T}_R\right|$ is the number of distinct values the complex attribute S obtained.

One interpretation is to obtain a result set T_R composed of tuples restricted to the k distinct values of the complex attribute S that are the closest to the query center s_q. Under this interpretation, it is not the number of tuples that is restricted, but the number of complex attribute values. As a result

Query	Variation	Algebraic Representation	n	m
$Q1$ - Which are the items sold that have their picture amongst the 5 most similar ones to picture s_q ?	Counting values	$\ddot{\sigma}_{(\texttt{PaintingPicture}\ k\text{-}NN(d,5)\ s_q)}\texttt{ItemSold}$	$\geqslant 5$	5
$Q2$ - Which are the 5 items sold whose pictures are the most similar ones to picture s_q ?	Counting tuples	$\dot{\sigma}_{(\texttt{PaintingPicture}\ k\text{-}NN(d,5)\ s_q)}\texttt{ItemSold}$	5	$\leqslant 5$

Table I. Representation of two queries deriving from the two possible interpretations of the k-NN operator.

the number of recovered tuples is $n \geqslant k$, if at least k tuples exist in T, and the number of distinct values of the complex attribute S is $m = k$. We call this interpretation, that limits the number of complex attribute values in the result set, *counting values*. We call the corresponding select variation as the "k-select" operator and note it as $\ddot{\sigma}_{(\mathsf{S}\ k\text{-}NN(d,\xi)\ s_q)}\mathsf{T}$.

The second interpretation corresponds to obtaining a result set T_R composed of the k tuples whose complex attribute S are the closest to the query center s_q. Under this interpretation, the number of tuples obtained is restricted to be $n = k$, but the number m of complex attribute values obtained can be any $0 \leqslant m \leqslant k$, including zero. We call this interpretation, that limits the number of tuples in the result set, *counting tuples*. We call the corresponding select variation as the "kt-select" operator and note it as $\dot{\sigma}_{(\mathsf{S}\ k\text{-}NN(d,\xi)\ s_q)}\mathsf{T}$.

With the two possible interpretations, each k-NN-based similarity selection operator can result in 2 different queries, with 2 distinct result set cardinalities. Table I shows the algebraic representation for queries *Q1* and *Q2* from the example given in the Section 1, with their corresponding cardinalities of both the number of tuples (n) and number of complex attribute values retrieved (m).

Taking into account the SIREN SQL extension presented in Section 2.1, the distinction among both interpretations can be summarized into a single optional keyword: `[VALUES | TUPLES]` in the `AFTER` k clause, so $Q1$ and $Q2$ can be expressed as:

```
SELECT Item
    FROM ItemSold
    WHERE PaintingPicture NEAR s_q STOP AFTER 5 [ VALUES | TUPLES ]
```

where the optional keyword `VALUES` indicates the *"counting values"* (i.e. $Q1$) interpretation, while the keyword `TUPLES` indicates the *"counting tuples"* (i.e. $Q2$) option. If the clause is omitted, the default operation is counting values, the one most studied so far.

The k-nearest neighbor operator k-NN returns true for a predicate $(t_i[\mathsf{S}]\ k\text{-}NN(d,k)\ s_q)$ over a tuple $t_i \in \mathsf{T}$ if $t_i[\mathsf{S}]$ is one of the k elements in the search space nearest to the query center. The search space of either $\ddot{\sigma}_{(\mathsf{S}_b\ k\text{-}NN(d_b,k_b)\ s_{qb})}\mathsf{T}$ or $\dot{\sigma}_{(\mathsf{S}_b\ k\text{-}NN(d_b,k_b)\ s_{qb})}\mathsf{T}$ is the active domain S of S. However, if any of the two k-NN-based selection operators are applied over the result T_{Rtemp} of a previous selection operation (regardless of the previous selection type), the search space is $\pi_{\{\mathsf{T}_{R_{temp}}\}}$. Therefore, both operators are not commutative, disjunctive nor associative. In fact, no one of $\ddot{\sigma}_{(k\text{-}NN\text{Pred})}\mathsf{T} \bigcap \sigma_{(\text{AnyPred})}\mathsf{T}$, $\ddot{\sigma}_{(k\text{-}NN\text{Pred})}\left(\sigma_{(\text{AnyPred})}\mathsf{T}\right)$ or $\sigma_{(\text{AnyPred})}\left(\ddot{\sigma}_{(k\text{-}NN\text{Pred})}\mathsf{T}\right)$ are equivalent to each other.

The only way to univocally process similarity selections embedded in queries with multiple single-attribute selections is to assume that a SQL command like `WHERE` $\langle$AnyPred$\rangle$ `AND` $\langle k\text{-}NN\text{Pred}\rangle$ is translated into the algebraic expression $\sigma_{(\text{AnyPred})}\mathsf{T} \bigcap \ddot{\sigma}_{(k\text{-}NN\text{Pred})}\mathsf{T}$. Thus, all the non-similarity-based selections can be pipelined in the usual way, and the result is intersected with the results of

each similarity-based ones. If the intention is to obtain a pipelined expression, such, for example, as $\boxed{\sigma_{(\text{AnyPred})}\left(\ddot{\sigma}_{(k\text{-}NN\text{Pred})}\mathsf{T}\right)}$, one must express it explicitly in the query command using the `FROM` clause to obtain the search space $\mathsf{T}_{Rtemp} = \pi_{\left\{\ddot{\sigma}_{(k\text{-}NN\text{Pred})}\right\}}\mathsf{T}$.

In this way, all the usual properties of the set theory hold and can be employed to interpret, optimize and execute the queries, provided the respective k-selection and kt-selection operators are implemented in the target DBMS. As follows, Algorithms 1 and 2 describe the basic implementation of each interpretation, to execute k-NN-based selection counting tuples and counting values, respectively. They show how the respective selection operator must be implemented using the k-NN comparison operator, indicated in the algorithms as a call to procedure "i-th-NearestNeighbor" to choose the tuples to be retrieved, which is indicated as a call to procedure "getTuples". Notice that procedures "i-th-NearestNeighbor" and "getTuples", as well as the procedures which implement the access methods that execute the selection operator in the database engine are the existing ones. Hence, every access method able to execute similarity-based selections must be changed to embed the algorithms shown, including *table scan*, *index scan*, *multi-index scan* and *index-only scan*.

Algorithm 1: Counting Tuples

```
Data: k, s_q
Result: resultSet
initialization;
begin
    for i = 1:k do
        complexAtr :=
        i-th-NearestNeighbor(Sq);
        tuples := getTuples( complexAtr );
        resultSet += tuples;
        if resultSet.size() ⩾ k then
            break;
```

Algorithm 2: Counting Values

```
Data: k, s_q
Result: resultSet
initialization;
begin
    for i = 1:k do
        complexAtr[i] :=
        i-th-NearestNeighbor(Sq);
    tuples := getTuples( complexAtr );
    resultSet := tuples;
```

These algorithms were employed in the experiments in Section 5, and to obtain the results shown as the fourth or fifth columns in Figure 1.

5. EXPERIMENTAL RESULTS

In this section we compare both proposed interpretations of the k-NN operator regarding their execution time and the differences in result set cardinality. The operators were implemented in the similarity retrieval engine (SIREN), presented in Section 2.1, and use the Slim-tree implementation available in the Arboretum library [GBDI-ICMC-USP] as the metric access method for the complex data. The experiments were performed on a computer with an Intel Core i7-2720QM 2.20GHz CPU and 8 GB of RAM, under the Ubuntu Linux 12.04 operating system. Two datasets were used to evaluate the results:

— **Painting:** A dataset representing an art gallery, as the example presented on Section 1, where copies of paintings are sold and the information regarding each sale is logged into a RDBMS. This dataset is based on a real world one [1], and contains 1300 distinct paintings. The data about picture purchases (attributes `SaleDate`, `Item` and `ClientName`) where randomly generated following an uniform distribution for the amount of purchases, varying from 1 to 10 for each original painting,

[1] http://commons.wikimedia.org/wiki/Category:Paintings – Last accessed: Apr 25, 2014

resulting in a dataset with 6890 tuples. The distance function was computed using the Euclidean distance (L_2) and the comparisons were made upon Haralick texture descriptors [Haralick et al. 1973] extracted from each image.

— **NSF:** A dataset containing 129,000 abstracts and metadata (year, institution etc.) describing the research awards granted by the U.S. National Science Foundation (NSF) between 1990 and 2003[2]. We employed the Jaccard distance over a complex attribute composed of a bag of words extracted from the abstract to compare **NSF** elements.

We performed queries ***Q1*** (counting values) and ***Q2*** (counting tuples) over the art gallery dataset presented in Section 1 varying k:

—"***Q1*** *(counting values) - Which are the items sold that have their picture amongst the k most similar ones to picture* s_q *?*" executing $\boxed{\ddot{\sigma}_{(\texttt{PaintingPicture}\ k\text{-}NN(L_2(\texttt{Haralick}),\ k)\ s_q)}\texttt{ItemSold}}$.

—"***Q2*** *(counting tuples) - Which are the k items sold whose pictures are the most similar ones to picture* s_q *?*" executing $\boxed{\dot{\sigma}_{(\texttt{PaintingPicture}\ k\text{-}NN(L_2(\texttt{Haralick}),\ k)\ s_q)}\texttt{ItemSold}}$.

where k varies from 1 to 25 in 5 step increments, and s_q is randomly sampled from `ItemSold.PaintingPicture`. The measurements were performed averaging 100 queries with varying query centers s_q. We also evaluated the average number of disk accesses and number of distance calculations required.

Our goal with this experiment was to evaluate the performance of both interpretations, as well as measure the end-result cardinalities for tuples and complex attribute values. As such, three distinct measurements were collected:

—What was the average query time for each value of k?
—How many tuples did the queries retrieve?
—How many distinct pictures (complex attribute values) did the query retrieve?

Figure 2 presents the results for both operators applied over the **Paintings** dataset, showing the three measurements collected and also the average number of disk accesses and distance calculations. Since the dataset has multiple values for each complex attribute `PaintingPicture`, it shows a noticeable difference between both operators regarding the number of tuples and distinct complex attribute values retrieved. As a consequence, executing the counting values interpretation is more costly, as it continues searching for possible results after counting tuples already stopped, being on average 36% slower, but having a 550% larger result set.

To provide a different perspective, we performed the same experiments on the `NSF` dataset. In contrast to the art gallery, the `NSF` dataset is much larger and does not contain repetition, being purely metric. As such, both interpretations of the k-NN operator have the same end result (complex attribute values retrieved and tuple cardinality). The purpose of this experiment is to verify if there is any significant difference in performance when either operator is employed, as it is expected that both retrieve the same result.

The relation containing the `NSF` dataset comprises three attributes: a unique *ID*, the publication *year* and the bag of words `BOWords`. We performed queries ***Q1*** and ***Q2*** as:

—"***Q1*** *(counting values) - Which are the articles whose keywords are amongst the k most similar to article* s_q *ones ?*" executing $\boxed{\ddot{\sigma}_{(\texttt{BOWords}\ k\text{-}NN(\texttt{Jaccard},\ k)\ s_q)}\texttt{NSF}}$.

[2]http://archive.ics.uci.edu/ml/datasets.html – Last accessed: Apr 25, 2014

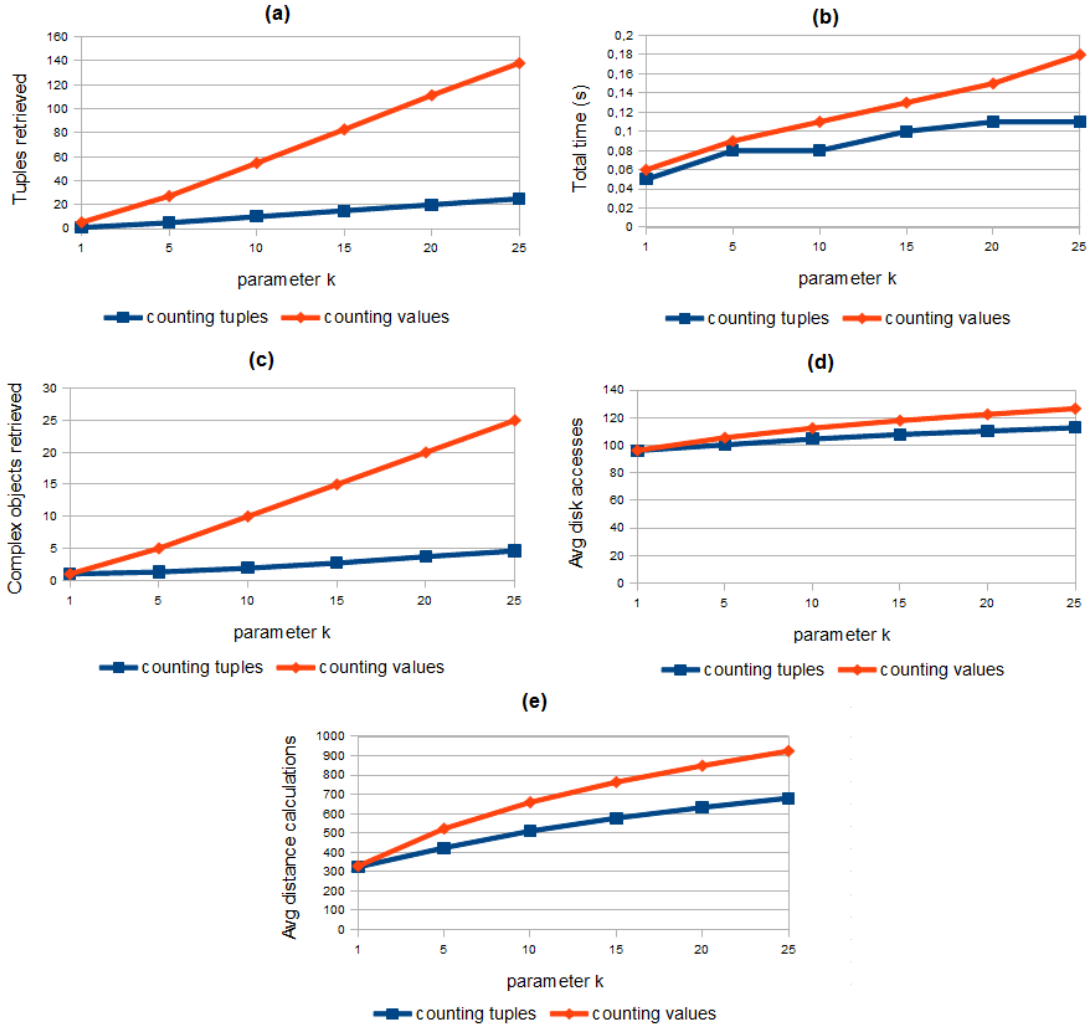

Fig. 2. Measurements for the two interpretations of the k-NN operator performed over the **Paintings** dataset for varying values of k. Each measurement is the average of 100 queries over random query centers. (a) Number of tuples retrieved for each value of k. (b) Total query time elapsed for each value of k. (c) Number of distinct complex objects (i.e. paintings) retrieved for each value of k (d) Average number of disk accesses for each value of k (e) Average number of distance calculations for each value of k.

—"***Q2*** *(counting tuples) - Which are the k articles whose keywords are the most similar to articles* s_q *ones ?*" executing $\boxed{\dot{\sigma}_{(\texttt{BOWords}\ k\text{-}NN(\texttt{Jaccard},\ k)\ s_q)}\texttt{NSF}}$.

Figure 3 presents the results of both operators for the **NSF** dataset. The results were obtained by performing 100 queries with random query centers for each value of k, and averaging the results. The experiments show that, in absence of repetition in the complex attribute value, the two interpretations converge, both in result set cardinality and in performance.

6. CONCLUSION AND FUTURE WORK

In this article we presented two novel ways of interpreting the k-NN operator when integrating similarity queries within a relational database management system and with the relational model. The new interpretations arise from the requirements originated from the inclusion of complex data as

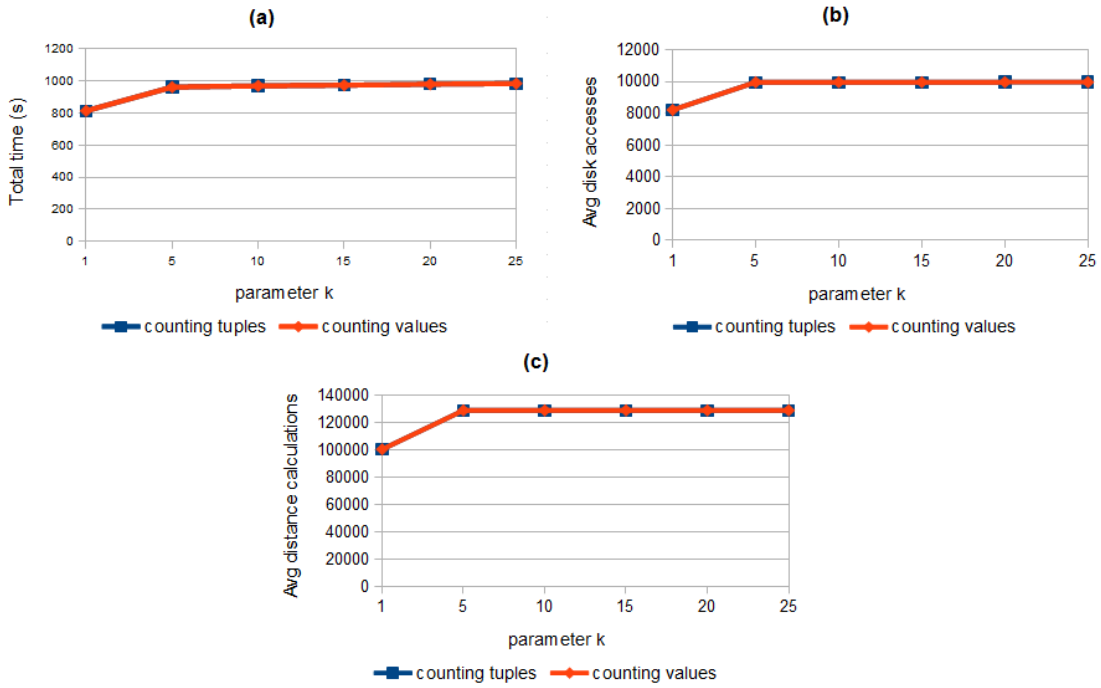

Fig. 3. The two interpretations of the k-NN operator performed over the NSF dataset with various values of k. A hundred queries over random query centers were averaged for the measurements. (a) Total query time elapsed for varying k. (b) Average number of disk accesses for each value of k. (c) Average number of distance calculations for each value of k.

attribute types in the relations and the corresponding similarity-based operators required to search them. The first is the traditional interpretation, which happens when a complex attributes value never occurs in more than one tuple, that is, the complex attribute is a relation key. This interpretation, which we call *counting values*, derives from the metric space paradigm, which imposes that repetition can not occur. However, when integrating similarity queries into the RDBMS, it is necessary to take into account the fact that, in real applications, multiple tuples may often store the same complex attribute value. This fact turns having only the *counting values* interpretation insufficient, and so we proposed a second one: *counting tuples*. The syntax to distinguish both interpretations was added with low impact to SQL, and if the database presents no repetition they are equivalent, both in performance and in result set cardinality.

As future works, we will look to identify the corresponding algebraic properties, aiming to enable query rewriting and optimization during query processing, including the usage of multiple indexes, as it occurs with the scalar data in traditional RDBMS, but allowing for similarity-, identity- and TOR-based ones to answer a query that combines any of those predicate types. We are also working on how the two interpretations of the k-NN-based selection operator affect similarity joins and similarity-based aggregation functions, both from the conceptual perspective and from the algebraic and algorithmic perspectives.

REFERENCES

Adali, S., Bonatti, P., Sapino, M. L., and Subrahmanian, V. S. A multi-similarity algebra. *SIGMOD Record* 27 (2): 402–413, 1998.

Barioni, M. C. N., Razente, H. L., Jr., C. T., and Traina, A. J. M. Querying complex objects by similarity in SQL. In *Proc. of the Brazilian Symposium on Databases*. pp. 130–144, 2005.

Barioni, M. C. N., Razente, H. L., Traina, A. J. M., and Jr., C. T. Siren: A similarity retrieval engine for complex data. In *Proc. of the Intrn. Conf. on Very Large Data Bases*. pp. 1155–1158, 2006.

Barioni, M. C. N., Razente, H. L., Traina, A. J. M., and Jr., C. T. Seamlessly integrating similarity queries in SQL. *Softw., Pract. Exper.* 39 (4): 355–384, 2009.

Batko, M., Novak, D., and Zezula, P. Messif: Metric similarity search implementation framework. In *Proc. of the 1st Intrn. Conf. on Digital Libraries: Research and Development*. pp. 1–10, 2007.

Belohlávek, R., Opichal, S., and Vychodil, V. Relational algebra for ranked tables with similarities: Properties and implementation. In *Proc. of the 7th Intrn. Conf. on Intelligent Data Analysis*. pp. 140–151, 2007.

Belohlávek, R. and Vychodil, V. Relational algebra for multi-ranked similarity-based databases. In *IEEE Symposium on Foundations of Computational Intelligence*. pp. 1–8, 2013.

Budíková, P., Batko, M., and Zezula, P. Query language for complex similarity queries. In *Proc. of the 16th East European Conf. on Advances in Databases and Information Systems*. pp. 85–98, 2012.

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

Faloutsos, C. *Searching Multimedia Databases by Content*. Advances in Database Systems, vol. 3. Kluwer, 1996.

Faloutsos, C. Indexing of multimedia data. In *Multimedia Databases in Perspective*. pp. 219–245, 1997.

Ferreira, M. R. P., da Silva, M. P., Traina, A. J. M., Jr., C. T., de Amo, S. A., Pereira, F. S., and Chbeir, R. Integrating user preference to similarity queries over medical images datasets. In *Proc. of the Intrn. Symposium on Computer-Based Medical Systems (CBMS 2010)*. pp. 1–6, 2010.

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

Ferreira, M. R. P., Traina, A. J. M., Dias, I., Chbeir, R., and Jr., C. T. Identifying algebraic properties to support optimization of unary similarity queries. In *AMW*, 2009.

Figueiredo, F., Pinto, H., Belém, F., Almeida, J., Gonçalves, M., Fernandes, D., and Moura, E. Assessing the quality of textual features in social media. *Information Processing and Management* 49 (1): 222–247, 2013.

GBDI-ICMC-USP. Gbdi arboretum library. `http://www.gbdi.icmc.usp.br/downloads/arboretum/`. Accessed: 2014-04-24.

Guliato, D., de Melo, E. V., Rangayyan, R. M., and Soares, R. C. POSTGRESQL-IE: an image-handling extension for PostgreSQL. *Journal of Digital Imaging* 22 (2): 149–165, 2009.

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

Hu, H. and Lee, D. L. Range nearest-neighbor query. *IEEE Transactions on Knowledge and Data Engineering*. 18 (1): 78–91, 2006.

Kaster, D. S., Bugatti, P. H., Ponciano-Silva, M., Traina, A. J. M., Azevedo-Marques, P. M., Santos, A. C., and Jr., C. T. MedFMI-SiR: a powerful DBMS solution for large-scale medical image retrieval. In *Proc. of the Second Intrn. Conf. on Information Technology in Bio- and Medical Informatics*. pp. 16–30, 2011.

Kaster, D. S., Bugatti, P. H., Traina, A. J. M., and Jr., C. T. FMI-SiR: a flexible and efficient module for similarity searching on oracle database. *Journal of Information and Data Management* 1 (2): 229–244, 2010.

Koopman, B., Zuccon, G., Bruza, P., Sitbon, L., and Lawley, M. An evaluation of corpus-driven measures of medical concept similarity for information retrieval. In *Proc. of the 21st ACM Intrn. Conf. on Information and Knowledge Management*. pp. 2439–2442, 2012.

Li, C., Chang, K. C.-C., Ilyas, I. F., and Song, S. RankSQL: query algebra and optimization for relational Top-k queries. In *Proc. of the 2005 ACM SIGMOD Intrn. Conf. on Management of Data*. pp. 131–142, 2005.

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

Serra, E., Cortez, E., da Silva, A. S., and de Moura, E. S. On using wikipedia to build knowledge bases for information extraction by text segmentation. *Journal of Information and Data Management* 2 (3): 259–272, 2011.

Silva, Y. N., Aly, A. M., Aref, W. G., and Larson, P.-A. SimDB: a similarity-aware database system. In *Proc. of the 2010 ACM SIGMOD Intrn. Conf. on Management of Data*. pp. 1243–1246, 2010.

Silva, Y. N., Aref, W. G., Larson, P.-A., Pearson, S. S., and Ali, M. H. Similarity queries: Their conceptual evaluation, transformations, and processing. *The VLDB Journal* 22 (3): 395–420, 2013.

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

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 *Proc. of the 15th ACM Intrn. Conf. on Information and Knowledge Management*. pp. 4–13, 2006.