A 'Wider' Concept for Similarity Joins

Luiz Olmes Carvalho, Willian Dener de Oliveira, Ives R. V. Pola, Agma J. M. Traina, Caetano Traina Jr.

Institute of Mathematics and Computer Sciences - University of São Paulo, São Carlos - SP, Brazil
{olmes, willian, ives, agma, caetano}@icmc.usp.br

Abstract. Join is one of the most studied and employed retrieval operators made available by the modern relational database management systems (RDBMSs). This binary operator is algebraically defined as a Cartesian product followed by the selection operator that specifies the join condition. In modern RDBMS, the join condition employs comparison operators based both on equality and on the Total Ordering Relationship. The resulting selection and join operators based on them are commutative, so they can be executed in any order, regardless of the expression ordering in the query command, allowing the RDBMS to look for optimized query evaluation strategies. However, more complex data do not follow equality nor TOR properties. In fact, they are better compared by similarity. Join operators based on similarity comparison operators are called "similarity joins". There exist similarity joins that can be executed directly as a similarity selection following a Cartesian product, but others demand extra and/or intermediate operations before they can compute the final result. In this paper, we analyze the existing types of similarity join in an algebraic way and we claim that in fact some of them generate operators that are not a join, but other binary operators. We identify a novel binary operator that embraces them and provide three variations of the new proposed operator. As an important follow up, the new operator allows relationally expressing and answering queries that previously could be processed only in sequential ways. The experiments performed on real data sets support our claim that our new operator is fast and flexible enough to be represented in the relational algebra and included into the RDBMSs, and it meets the demands for similarity queries from modern applications.

Categories and Subject Descriptors: I.1 [**Symbolic and Algebraic Manipulation**]: Miscellaneous

Keywords: database operators, join processing, relational algebra, similarity search

1. INTRODUCTION

In the modern relational database management systems (RDBMSs), the information retrieval is performed by query operators, such as project, union, selection and join [Date 2011]. When the query requires comparing attribute values, it employs the selection and/or join operators. Traditionally, RDBMS are well-equipped to handle scalar data, such as number, dates and small character strings. Therefore, until recently, the most frequent comparison operators were those based either on equality or on Total Order Relationship (TOR). Now, complex data are increasingly being included among the data types handled by RDBMS, which are better compared by similarity-based comparison operators. Employing similarity-based comparison operators in selection or join conditions generates similarity-based instances of those operators.

Join is one of the most useful operators in query processing and has been extensively investigated. The literature often assumes that there exist three distinct similarity join operators [Böhm and Krebs 2004]: the range join, the k-nearest neighbor join and the k-closest neighbor join. The range join relies on a given threshold ξ in order to retrieve all element pairs within the distance ξ. This is the join operator occurring most frequently in the literature, and in fact several studies call it just "similarity join" [Jacox and Samet 2008; Silva et al. 2013] – we call it "(similarity) range join" to avoid misinterpretation, as by "similarity join" we refer to any join operator involving similarity. Direct

This work has been supported by CAPES, STIC-AmSud, FAPESP, CNPq and RESCUER.

application fields of the range join operator include, for example, string matching [Qin et al. 2013; Wang et al. 2014], near duplicate object detection [Xiao et al. 2011] and data cleaning [Chaudhuri et al. 2006]. The other two similarity join operators are based on retrieving a given amount k of pairs. The k-nearest neighbor join retrieves, for each element from the first relation, the k elements from the second closest one. The k-closest neighbor join retrieves the k most similar element pairs among the two relations in general. Those operators can be applied aiming at data mining and data integration [Böhm and Krebs 2004], map-reduce [Zhang et al. 2012] and high dimensional data querying [Yu et al. 2007; Liu and Hao 2007]. In order to choose the overall k most similar pairs, the k-closest neighbor join internally executes sorting and other intermediate operations, which does not comply with the join operation definition, that should be just a Cartesian product followed by a selection operation. Therefore, *we claim that the k-closest neighbor operator is not a join.* We analyze its requirements, and identify that other interesting operators exist in its same class.

In this article, we algebraically analyze the similarity-based variations of the selection and join operators to support the claim that operators such as the k-closest neighbor join in fact belong to a new class of operators. The new class comprises binary operators that retrieve related pairs but require more operations, such as sorting, besides those required by 'real' joins. As the new binary class embraces a basic join and more operations, we refer to them as "**Wide-Joins**", or w-joins. We also investigate employing the similarity range and k-nearest neighbor comparison operators to compose the wide-join predicates, which generates other instances of our novel operator class.

The wide-join variations enable the execution of queries previously answered only through algorithmic processing, i.e., their expression in a relational language required complicated expressions. For instance, consider the following scenarios:

(1) A company needs to pick strategic cities next to Capitals to build new branches. The CEOs are interested in the following query: *Q1 - Which are the 8 closest pairs of Capitals and cities such that the distances between them are at most 10 kilometers?*
(2) In a climate sensor network, some sensors must be turned off to save energy, but only sensors for which there is at least another with similar measures within their covering area that can be powered off. The sensor monitoring team could be interested in knowing: *Q2 - Which are the 50 pairs of sensors whose measurements are the most similar in a global context, such that, each sensor is paired with the 5 similar ones and restricted by a dissimilarity of 0.5?*

Notice that none of the current similarity selection or join operators deal neither with query *Q1* nor *Q2*. However, as we will show, distinct variations of our proposed operator are able to answer them. We performed experiments evaluating speed and scalability of the wide-join operators, using both real and synthetic data sets and showed that they are able to perform similarity retrieval as demanded by the modern information systems.

The main contributions of this article are summarized as follows:

—We analyze the k-closest neighbor join and show that it is not a classic join but a 'wider' operation that includes the join concept;
—We present the definition of a whole new class of binary similarity operators that enlarges the applicability of join for similarity retrieval, helping executing queries targeting data analysis support;
—We identify three derived operators of the new class and express them in relational algebra in a way that enables their seamless integration into RDBMS;
—We present a general algorithm to execute the new proposed class of operators.

The remainder of this article is organized as follows. Section 2 reviews the theoretical concepts required. Section 3 describes the proposed class of operators. Experiments and results are presented in Section 4. Section 5 outlines the work related to ours. Finally, Section 6 presents a general discussion and concludes the article.

2. BACKGROUND

In this section we combine traditional concepts from the relational algebra and query processing with similarity-based ones, which are being developed by several research groups around the world. However, the way we present and integrate them is novel, in the sense that we provide a unified basis to treat both the traditional concepts and the similarity-base developments.

The relational algebra provides six fundamental operators: Projection (π), Selection (σ), Rename (ρ), Cartesian product ($\times$), Union ($\cup$) and Difference ($-$). All the others, called derived operators, such as intersection ($\cap$) and join ($\bowtie$), can be expressed by combining the six primitive operators [Date 2011]. The development of derived operators allows easier but consistent formulation of queries, as well as the implementation of algorithms faster than those obtained by the composition of the primitive ones. For instance, the join operator is defined as:

DEFINITION 1 – JOIN OPERATOR. *The join of two relations* T_1 *and* T_2 *is a binary relational operator* $\bowtie$ *algebraically defined as a composition of two fundamental operators: a Cartesian product followed by the selection that perform the join predicate c:* $\mathsf{T}_1 \overset{(c)}{\bowtie} \mathsf{T}_2 \Leftrightarrow \sigma_{(c)} (\mathsf{T}_1 \times \mathsf{T}_2)$

Besides the fundamental and derived operators, others usually referred to as extended operators are useful and required when implementing DBMS based on the relational model. Those operators are not algebraic ones, but (as the Rename operator) they are required in practical tools embodying the relational model. Two extended operators, presented in [Garcia-Molina et al. 2000], are interesting to our studies:

(1) The extension operator $\psi_{\{\exp \backslash \mathsf{A}\}}\mathsf{T}$: it adds a new attribute A to the relation T, and sets the values $t[\mathsf{A}]$ as the result of the expression exp executed over the attributes of each tuple $t[\mathsf{T}]$;
(2) The sorting operator $\omega_{\{\exp \backslash F\}}\mathsf{T}$: it adds a new attribute F to the relation T, and sets the values $t[F]$ as the result of processing exp executed over the relation T. Usually, exp is a function over a list L of attributes $L = \langle \mathsf{A}_1, \ldots \rangle \mid \{\mathsf{A}_1, \ldots\} \subseteq \mathsf{T}$ that sorts the tuples of T following the values of the attributes indicated in L and assigns $t[F]$ as a function of the resulting order.

Selection and join operators retrieve tuples that meet a predicate, defined as follows:

DEFINITION 2 – RETRIEVAL PREDICATE. *A retrieval predicate is a Boolean expression on terms c of the form* $c = (\mathsf{A}\ \theta\ v)$*, where* A *is an attribute of the involved relation,* v *is the value to which attribute* A *is compared and* θ *is a comparison operator defined for the domain* $\mathbb{A}$ *of* A*.*

For complex data, the comparison operator θ is usually based on similarity. To distinguish among the attributes stored in a relation that should be compared by similarity, we identify them using the symbol S and call them complex attributes, reserving the symbol E for for the scalar attributes. Thus, a relation T is a set of tuples whose schema is denoted by $\mathsf{T} = (\mathsf{E}_1, ..., \mathsf{E}_q, \mathsf{S}_1, ..., \mathsf{S}_p)$, where $\mathsf{E}_1, \ldots \mathsf{E}_q$ are scalar attributes and $\mathsf{S}_1, \ldots \mathsf{S}_p$ are complex attributes. The scalar attributes E can be compared either by identity or by TOR, and the complex attributes S can be compared either by identity or by similarity-based operators. Table I summarizes the symbols employed in this article.

When comparing complex attributes, the condition c is expressed as $c = (\mathsf{S}\ \theta_s\ s_q)$, in which S is a complex attribute defined over a complex domain $\mathbb{S}$, θ_s is a similarity-based comparison operator defined on $\mathbb{S}$ and the value $s_q \in \mathbb{S}$ is the query center. Similarity-based operators are mostly employed over metric spaces, defined as a pair $\langle \mathbb{S}, d \rangle$ in which $d : \mathbb{S} \times \mathbb{S} \mapsto \mathbb{R}_+$ is a metric [Searcóid 2007]. There are two similarity-based operators broadly studied, which are expressed in Definition 3.

DEFINITION 3 – SIMILARITY-BASED COMPARISON OPERATOR. *A similarity term* $c = (\mathsf{S}\ \theta_s\ s_q)$ *compares the attribute* S *with a query center* $s_q \in \mathbb{S}$*. The two similarity comparison operators* θ_s *are:*

—***Similarity range comparison*** - $\theta_s = Rng(d, \xi)$*: returns* `true` *for every tuple* t *such that* $d(t[\mathsf{S}], s_q) \leq \xi$*.*

Table I. Symbols employed in the article

Symbol	Meaning	Symbol	Meaning
A	an attribute (either scalar or complex)	s_q	a complex query center
$\mathbb{A}$	the domain of the attribute A	T	a relation with scalar and complex attributes
c	a retrieval predicate (or condition)	T_R	a result relation
d	a metric (distance function)	t	a tuple
E	a scalar attribute	$t[\mathsf{A}]$	the value of the attribute A in the tuple t
k	constant value	$t[\mathsf{T}]$	a tuple of the relation T
k-NN	k-nearest neighbor comparison operator	ξ	a threshold value
k-CN	k-closest neighbor comparison operator	θ	a comparison operator
Rng	similarity range comparison operator	σ	traditional select operator
S	a complex attribute	$\ddot{\sigma}$	select operator employing k-NN or k-CN
$\mathbb{S}$	a complex domain	$\bowtie$	join operator
s_i	a complex element in $\mathbb{S}$	$\ddot{\bowtie}$	wide-join operator

—***k-nearest neighbor comparison*** - $\theta_s = k\text{-}NN(d,k)$*: returns* `true` *for every tuple t such that $t[\mathsf{S}]$ is one of the k nearest neighbors of s_q.*

The similarity range comparison operator is a tuple-oriented operator, that is, given two elements $t[\mathsf{S}], s_q \in \mathbb{S}$ and a predicate $(\mathsf{S}\ Rng(d,\xi)\ s_q)$ it can evaluate the answer without further knowledge of the remaining elements in the dataset. Moreover, the answer $(s_i\ Rng(d,\xi)\ s_q)$ for a given $s_i \in \mathbb{S}$ is always the same for any $\mathsf{S} \subset \mathbb{S}$. The k-nearest neighbor comparison operators is a search-space-oriented operator, because given two elements $t[\mathsf{S}], s_q \in \mathbb{S}$ and a predicate $(\mathsf{S}\ k\text{-}NN(d,k)\ s_q)$ the answer $(s_i\ k\text{-}NN(d,k)\ s_q)$ for a given $s_i \in \mathbb{S}$ varies depending on the space $S \subset \mathbb{S}$ where the search is performed.

Among the operators of the relational algebra that perform attribute value comparison in the tuples, selection and join are the most general and should evaluate any predicate, either expressing traditional or similarity constructions. With respect to selection, the similarity-based operators lead to two instances of the similarity selection. As the similarity range comparison operator is an tuple-oriented operator, it can be used in the fundamental selection operator σ according to Definition 4.

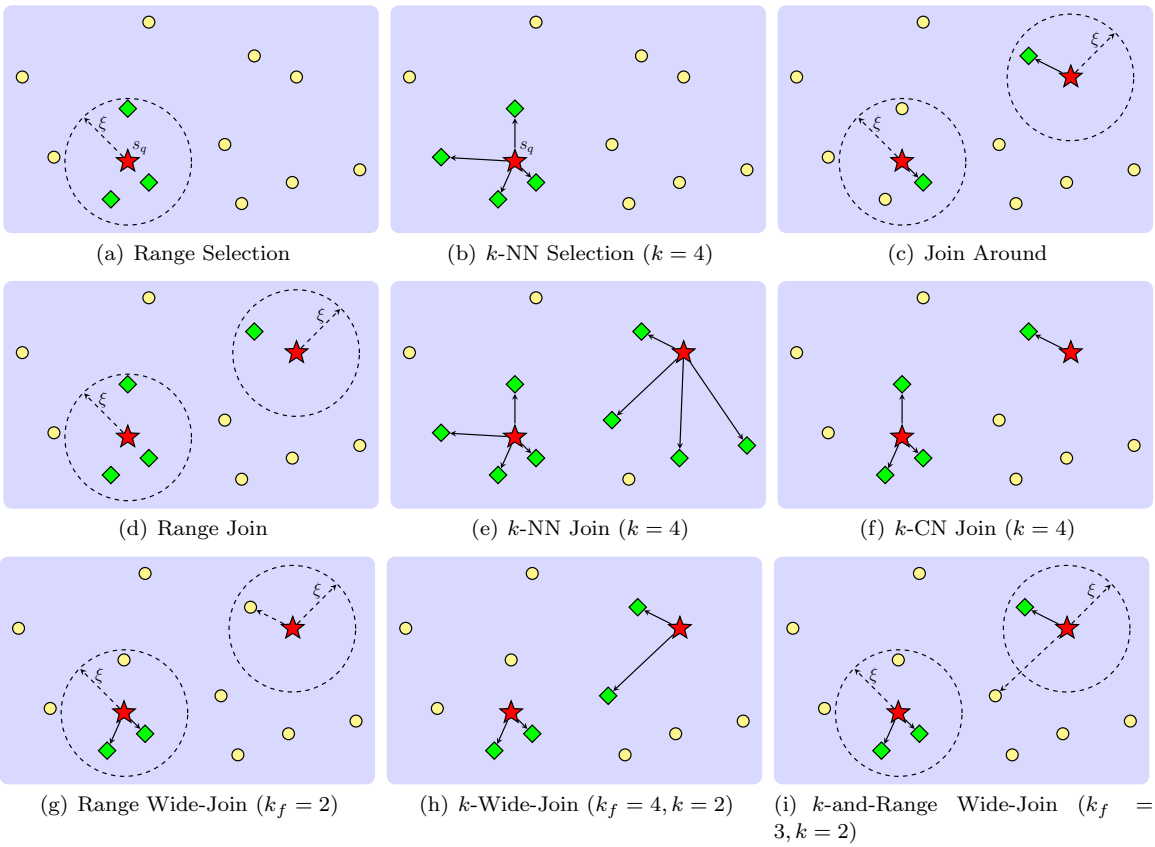
Definition 4 – Similarity range selection. *The similarity range selection $\sigma_{(\mathsf{S}\ Rng(d,\xi)\ s_q)}\mathsf{T}$ (Figure 1(a)) restricts the tuples of T to those where the complex attribute S is similar to s_q considering a similarity threshold of ξ. It retrieves a subset $\mathsf{T}_R \subset \mathsf{T}$ such that $\mathsf{T}_R = \{s_i \in \mathsf{T} \mid d(s_i, s_q) \leq \xi\}$.*

As the k-nearest neighbor comparison operators is not an tuple-oriented operator, it cannot be used in the fundamental selection operator σ, because it is not guaranteed that the algebraic properties of σ hold. Thus, we defined a variation of σ specifically for the k-nearest neighbor comparison, which we call *k-selection* and denote it as $\ddot{\sigma}$, according to Definition 5. Notice that the properties of $\ddot{\sigma}$ are distinct from those of σ. For instance, σ is a commutative operator, whereas $\ddot{\sigma}$ is not.

Definition 5 – k-nearest neighbor k-selection. *The k-nearest neighbor k-selection $\ddot{\sigma}_{(\mathsf{S}\ k\text{-}NN(d,k)\ s_q)}\mathsf{T}$ (Figure 1(b)) restricts the tuples of T to those that are the k nearest to s_q. It retrieves a subset $\mathsf{T}_R \subset \mathsf{T}$ such that $\mathsf{T}_R = \{s_i \in \mathsf{T} \mid \forall s_j \in (\mathsf{T} \setminus \mathsf{T}_R) \Rightarrow |\mathsf{T}_R| = k \wedge d(s_i, s_q) \leq d(s_j, s_q)\}$.*

A more formal definition for k-nearest neighbor queries can be found in [Ferreira et al. 2009]. The join operator combines tuples from two relations T_1 and T_2 such that attributes $\mathsf{S}_1 \in \mathsf{T}_1$ and $\mathsf{S}_2 \in \mathsf{T}_2$ satisfy the join predicate. Both similarity-based comparison operators can be employed in join predicates to generate similarity-based join operators. Following, we present the definition of the instances of similarity joins studied in the literature [Böhm and Krebs 2004; Silva et al. 2013].

Definition 6 – Similarity range join. *The similarity range join $\mathsf{T}_1 \overset{(\mathsf{S}_1\ Rng(d,\xi)\ \mathsf{S}_2)}{\ddot{\bowtie}} \mathsf{T}_2$ (Figure 1(d)) combines the tuples of T_1 and T_2 whose distance between the pair $\langle \mathsf{S}_1, \mathsf{S}_2 \rangle$ of complex attributes is less*



(a) Range Selection (b) k-NN Selection ($k = 4$) (c) Join Around

(d) Range Join (e) k-NN Join ($k = 4$) (f) k-CN Join ($k = 4$)

(g) Range Wide-Join ($k_f = 2$) (h) k-Wide-Join ($k_f = 4, k = 2$) (i) k-and-Range Wide-Join ($k_f = 3, k = 2$)

Fig. 1. Similarity selection and similarity join: for joins, stars represent the elements of the first relation, circles represent the elements of the second and diamonds are the elements of the second relation paired with the elements of the first. For selection, a star is the query center and diamonds are the selected elements.

than or equal to the given threshold ξ. *It retrieves a set of pairs* T_R *such that* $\mathsf{T}_R = \{\langle t[\mathsf{T}_1], t[\mathsf{T}_2]\rangle \in (\mathsf{T}_1 \times \mathsf{T}_2) \mid d(s_i, s_j) \leq \xi, s_i \in \mathsf{S}_1 \subset \mathsf{T}_1, s_j \in \mathsf{S}_2 \subset \mathsf{T}_2\}$.

Definition 7 – k-nearest neighbor k-join. *The k-nearest neighbor k-join* $\mathsf{T}_1 \overset{(\mathsf{S}_1\ k\text{-}NN(d,k)\ \mathsf{S}_2)}{\bowtie} \mathsf{T}_2$ *(Figure 1(e)) combines the tuples of* T_1 *and* T_2 *such that, for each tuple* $t_1 \in \mathsf{T}_1$, *it retrieves the tuples from* T_2 *whose* S_2 *is one of the* k *nearest to* $t_1[\mathsf{S}_1]$. *It retrieves a set of pairs* T_R *such that* $\mathsf{T}_R = \{\langle t[\mathsf{T}_1], t[\mathsf{T}_2]\rangle \in (\mathsf{T}_1 \times \mathsf{T}_2) \mid \forall \langle s_i, s'_j\rangle \in ((\mathsf{T}_1 \times \mathsf{T}_2) \setminus \mathsf{T}_R) \Rightarrow |\mathsf{T}_R| = |\mathsf{T}_1| * k \wedge d(s_i, s_j) \leq d\big(s_i, s'_j\big)\}$.

Definition 8 – k-closest neighbor k-join. *The k-closest neighbor k-join* $\mathsf{T}_1 \overset{(\mathsf{S}_1\ k\text{-}CN(d,k)\ \mathsf{S}_2)}{\bowtie} \mathsf{T}_2$ *(Figure 1(f)) combines the tuples of* T_1 *and* T_2 *such that it retrieves the* k *pairs where* S_1 *are the nearest to* S_2. *It retrieves a set of pairs* T_R *such that* $\mathsf{T}_R = \{\langle t[\mathsf{T}_1], t[\mathsf{T}_2]\rangle \in (\mathsf{T}_1 \times \mathsf{T}_2) \mid \forall \langle s'_i, s'_j\rangle \in ((\mathsf{T}_1 \times \mathsf{T}_2) \setminus \mathsf{T}_R) \Rightarrow |\mathsf{T}_R| = k \wedge d(s_i, s_j) \leq d\big(s'_i, s'_j\big)\}$.

Definition 9 – Similarity join-around. *The similarity join-around* $\mathsf{T}_1 \overset{(\mathsf{S}_1\ A(d,\xi)\ \mathsf{S}_2)}{\bowtie} \mathsf{T}_2$ *(Figure 1(c)) combines the tuples from* T_1 *closest to* T_2 *regarding the complex attribute pair* $\langle \mathsf{S}_1, \mathsf{S}_2\rangle$, *providing they are at most the given threshold* ξ *apart. This operator combines the similarity range with a 1-nearest neighbor comparison operator in a conjunctive join condition and returns a set of pairs* T_R *such that* $\mathsf{T}_R = \{\langle t[\mathsf{T}_1], t[\mathsf{T}_2]\rangle \in (\mathsf{T}_1 \times \mathsf{T}_2) \mid d(s_i, s_j) \leq \xi\} \cap \{\langle t[\mathsf{T}_1], t[\mathsf{T}_2]\rangle \in (\mathsf{T}_1 \times \mathsf{T}_2) \mid \forall \langle s_i, s'_j\rangle \in ((\mathsf{T}_1 \times \mathsf{T}_2) \setminus \mathsf{T}_R) \Rightarrow |\mathsf{T}_R| \leq |\mathsf{T}_1| \wedge d(s_i, s_j) \leq d\big(s_i, s'_j\big)\}$.

3. THE SIMILARITY WIDE-JOIN OPERATOR

Evaluating the four similarity join operators presented in Section 2, notice that the similarity range join $\mathsf{T}_1 \overset{(Rng)}{\bowtie} \mathsf{T}_2$ and the k-nearest neighbor join $\mathsf{T}_1 \overset{(k\text{-}NN)}{\bowtie} \mathsf{T}_2$ are respectively the direct equivalent of the similarity range selection $\sigma_{(Rng)}\mathsf{T}$ and of the k-nearest neighbor selection $\ddot{\sigma}_{(k\text{-}NN)}\mathsf{T}$, employing the corresponding similarity range and k-nearest neighbor comparison operators. In its turn, the similarity join around $\mathsf{T}_1 \overset{(A)}{\bowtie} \mathsf{T}_2$ combines the similarity range and k-nearest neighbor comparison operators in a conjunctive predicate. However, notice that the k-closest neighbor join $\mathsf{T}_1 \overset{(k\text{-}CN)}{\bowtie} \mathsf{T}_2$ does not have a corresponding "k-closest neighbor comparison operator" nor it corresponds to a combination of existing comparison operators. This section explores in an algebraic way the properties of the k-closest neighbor join operator and, based on this analysis, stands that the k-CN join and its generated extensions belong to a new class of operators that extends the join concept. Following the presented definitions, we also identify two new variations of the "k-CN based join". Those derived extensions enable the answering of queries previously processed only in an algorithmic way.

To start our analysis, let us suppose by contradiction that the so called k-closest neighbor join is a join. As its predicate cannot be expressed combining existing comparison operators, a new "closest neighbor" comparison that would enable a k-closest neighbor selection should be defined as follows:

Conjecture 1 – The k-Closest Neighbor Selection Operator. *Let* T *be a relation,* S *a complex attribute in* T*,* $\mathbb{S}$ *a complex attribute domain such that* $Dom(\mathsf{S}) = \mathbb{S}$*,* d *a metric over* $\mathbb{S}$ *and* $s_q \in \mathbb{S}$ *be a query center. The* k*-closest neighbor selection operator* $\ddot{\sigma}_{(\mathsf{S}\ k\text{-}CN(d,k)\ s_q)}\mathsf{T}$ *retrieves from* T *those tuples in which the value of* S *is one of the* k *closest elements to the query center* s_q*, i.e., the tuples* $t_i[\mathsf{T}]$ *whose* $d(t_i[\mathsf{S}], s_q)$ *is one of the* k*-smallest distances.*

According to Conjecture 1, similarity selection query operators could employ three main similarity-based comparison operators in its predicate, which would generate corresponding three distinct implementations of the similarity selection operator: similarity range selection, k-nearest neighbor k-selection and k-closest neighbor k-selection. As the similarity range selection is a tuple-oriented comparison operator, it preserves the commutative, associative and distributive properties. Therefore, it can be combined with traditional or other similarity-based operators in any order in the queries, and the RDBMS should be able to choose the evaluation order that better retrieves the query answers. However, the k-nearest neighbor and the k-closest neighbor are search-space-oriented comparison operators, so they cannot be assumed to be commutative, associative nor distributive to other traditional or similarity-based operators. Thus, a RDBMS must evaluate them in the same order they appear in a query, restricting the possibilities of query optimizations.

The k-nearest neighbor selection retrieves the k tuples "nearest" to the query center, and the k-closest neighbor selection operator retrieves those k "closest". The former returns the tuples whose distance to the query center is one of the k smallest (nearest). According to the Conjecture 1, the latter one also returns the tuples whose distance is one of the k smallest (closest). Thus, with respect to a selection operation, both "nearest" and "closest" criteria have the same meaning, which allows stating the Postulate 1.

Postulate 1. *The* k*-nearest neighbor selection and the* k*-closest neighbor selection employ, as their predicates, similarity-based comparison operators that always produce the same result. Accordingly, they are equivalent operators:* $\ddot{\sigma}_{(\mathsf{S}\ k\text{-}NN(d,k)\ s_q)}\mathsf{T} \Leftrightarrow \ddot{\sigma}_{(\mathsf{S}\ k\text{-}CN(d,k)\ s_q)}\mathsf{T}$.

Regarding selection, Postulate 1 explains the reason for the absence of a "k-closest neighbor selection" and why Conjecture 1 does not hold: since $\ddot{\sigma}_{(k\text{-}NN)}$ and $\ddot{\sigma}_{(k\text{-}CN)}$ are equivalent, there is no need to keep both kinds of comparison in selection operators. However, Postulate 1 does not hold for joins.

Based on these concepts, there are two operators to execute similarity selections and three to execute similarity joins. Now, we recall that the join operator is not one of the six fundamental relational operators discussed in the Section 2, but a derived one that is expressed according to Definition 1. Following that definition, the similarity range join and the k-nearest neighbor join operators can be expressed according to the Equations 1 and 2, respectively:

$$\mathsf{T}_1 \overset{(\mathsf{S}_1 Rng(d,\xi)\mathsf{S}_2)}{\bowtie} \mathsf{T}_2 \Leftrightarrow \sigma_{(\mathsf{S}_1 Rng(d,\xi)\mathsf{S}_2)}\left(\mathsf{T}_1 \times \mathsf{T}_2\right) \quad (1) \qquad \mathsf{T}_1 \overset{(\mathsf{S}_1 k\text{-}NN(d,k)\mathsf{S}_2)}{\ddot{\bowtie}} \mathsf{T}_2 \Leftrightarrow \ddot{\sigma}_{(\mathsf{S}_1 k\text{-}NN(d,k)\mathsf{S}_2)}\left(\mathsf{T}_1 \times \mathsf{T}_2\right) \quad (2)$$

Following Definition 1, the k-closest neighbor join, might be expressed as

$$\mathsf{T}_1 \overset{(\mathsf{S}_1\ k\text{-}CN(d,k)\ \mathsf{S}_2)}{\ddot{\bowtie}} \mathsf{T}_2 \Leftrightarrow \ddot{\sigma}_{(\mathsf{S}_1\ k\text{-}CN(d,k)\ \mathsf{S}_2)}\left(\mathsf{T}_1 \times \mathsf{T}_2\right)$$

but, undoubtedly, that proposition does not hold, even because, according to Postulate 1, a k-closest neighbor selection operator $\left(\ddot{\sigma}_{(k\text{-}CN)}\right)$ does not exist. In fact, although a supposed "$\ddot{\sigma}_{(k\text{-}CN)}$" and the $\ddot{\sigma}_{(k\text{-}NN)}$ are equivalent operators such that $\ddot{\sigma}_{(\mathsf{S}\ k\text{-}CN(d,k)\ s_q)}\mathsf{T} \subseteq \ddot{\sigma}_{(\mathsf{S}\ k\text{-}NN(d,k)\ s_q)}\mathsf{T}$, it is not possible to employ the $\ddot{\sigma}_{(k\text{-}NN)}$ operator in order to represent the k-closest neighbor join because

$$\mathsf{T}_1 \overset{(\mathsf{S}_1\ k\text{-}CN(d,k)\ \mathsf{S}_2)}{\ddot{\bowtie}} \mathsf{T}_2 \nLeftrightarrow \ddot{\sigma}_{(\mathsf{S}_1\ k\text{-}NN(d,k)\ \mathsf{S}_2)}\left(\mathsf{T}_1 \times \mathsf{T}_2\right),$$

and $\ddot{\sigma}_{(\mathsf{S}_1\ k\text{-}NN(d,k)\ \mathsf{S}_2)}\left(\mathsf{T}_1 \times \mathsf{T}_2\right)$ defines a k-nearest neighbor join and not a k-closest neighbor join. The immediate conclusion is that the range join and the k-nearest neighbor join are in fact join operators, but the k-closest neighbor join is not, once it cannot be expressed according to the conceptual definition of the join operation.

Consequently, we refuse the conjecture that the k-join operator accepts either "k-nearest" or "k-closest" comparison operators in its predicate. Thus, whereas the "k-nearest" comparison really generates the k-nearest neighbor join, the "k-closest" comparison only generates a k-closest neighbor join only "extending" the join operation concept.

It is important to highlight that the "k-closest join" is a useful operator, with practical significance and applicability, such as those aforementioned in Section 1. It is not a *stricto sensu* join operator, although we will continue to refer to it as a "join", since this it the term that is broadly used in the literature. Therefore, from an algebraic point of view, the k-closest neighbor join operator is not a join operator: besides a Cartesian product followed by a selection, it needs further processing and another additional selection.

3.1 An algebraic definition for the "k-closest join"

The $\overset{(k\text{-}CN)}{\ddot{\bowtie}}$ operator requires a k-nearest neighbor selection over the Cartesian product of two relations and then, *it requires that this partial result be followed by an additional filtering step: it is necessary to sort the intermediate result tuples by the distances of the complex elements in each pair and, finally, to apply an 'additional selection' to filter the 'k' most similar of them.*

Notice that the additional selection is applied over an attribute that is not present in the original input relations: the ordinal value of the tuple in the sorted list. Although the studies in the literature consider that the "closest" operator is a join operation, indeed it is obtained as the Cartesian product of two relations followed by the execution of a sorting operation over the similarity value and by the selection of the most similar tuples. Executing a k-closest join based on the processing of a k-nearest join is just an algorithmic optimization, once it allows performing the sorting over a reduced amount of tuples. In both cases, the attribute containing the ordinal position of each tuple is computed after the internal join or the Cartesian product. Thus, distinctly from the internal join predicate processing, which can be processed in parallel or after the Cartesian product, the predicate of the second selection only can be executed after the internal join has been finished.

The k-closest join operator can be described in the Relational algebra as follows:

$$\mathsf{T}_1 \overset{(\mathsf{S}_1\ k\text{-}CN(d,k)\ \mathsf{S}_2)}{\bowtie} \mathsf{T}_2 \Leftrightarrow \pi_{\{T_{12}\}}\left(\sigma_{(ord\leq k)}\left(\omega_{\{Rank(dist)\setminus ord\}}\left(\psi_{\{d(\mathsf{S}_1,\mathsf{S}_2)\setminus dist\}}\left(\mathsf{T}_1 \overset{(\mathsf{S}_1\ kNN(d,k)\ \mathsf{S}_2)}{\bowtie} \mathsf{T}_2\right)\right)\right)\right) \quad (3)$$

where $d(\mathsf{S}_1, \mathsf{S}_2)$ is a metric applied to the attributes $\mathsf{S}_1 \in \mathsf{T}_1$ and $\mathsf{S}_2 \in \mathsf{T}_2$, such that $Dom(\mathsf{S}_1) = Dom(\mathsf{S}_2) = \mathbb{S}$, $Rank()$ is an external function that assigns a sequential ordinal number to each tuple in the relation sorted by the attribute ord and T_{12} is the set of all attributes from relations T_1 and T_2.

Equation 3 shows that the k-closest neighbor join does not require a third similarity selection operator (the k-closest neighbor selection), but emphasizes the need for another binary query operator, beyond the join. The new operator is also based on the existing similarity comparison operators and employs the concept of sorting. As the relational model does not support ordering, the definition in Equation 3 handles ordering as an extra attribute whose value is computed by an external function that defines the tuple sequence numbering them, allowing the proper tuples to be selected. The traditional selection operator σ is applied over a new attribute (ord) of the answer relation, containing just numeric values (ordinals). The last projection over the attributes of T_1 and T_2 gets rid of that attribute, so the sorting process exists only during the processing of the operator ω (Equation 3) and does not remain when it finishes. Thus, the k-closest neighbor join does not infringe the postulates of the Relational theory and, therefore, it does not change its representativeness.

3.2 Defining the similarity wide-join operator

Equation 3 also provides the conceptual basis for a new class of similarity-based operators, which includes ordering the result of an internal similarity join. The operators derived from this new class perform joins "in a wider sense", because beside the selection over the Cartesian product result, they execute an ordering (ω) of the distances, which were included in the working relation by a extension (ψ) operation. Notice that the ordering operation also require comparison operators. As the operators of the new class are composed of more operations than the traditional join, we refer to them as **wide-join**, or **w-join**. The wide-join is defined as follows:

Definition 10 – Similarity Wide-Join Operator. *Let T_1 and T_2 be two relations containing complex attributes $\mathsf{S}_1 \in \mathsf{T}_1$ and $\mathsf{S}_2 \in \mathsf{T}_2$, both sampling from the same complex attribute domain $\mathbb{S}$. Let also d be a metric defined on $\mathbb{S}$, k_f be an upper bound parameter and θ_s be a predicate that includes a similarity comparison operator as one of its term. Then, the wide-join* $\boxed{\mathsf{T}_1 \overset{(\mathsf{S}_1\ \theta_s\ \mathsf{S}_2),k_f}{\ddot{\bowtie}} \mathsf{T}_2}$ *is a binary operator that receives relations T_1 and T_2, combines the tuples from both relations satisfying condition $(\mathsf{S}_1\ \theta_s\ \mathsf{S}_2)$, sorts the result by the similarity among attributes S_1 and S_2 and returns the k_f tuples having S_1 and S_2 most similar. The w-join operator expression in relational algebra is:*

$$\mathsf{T}_1 \overset{(\mathsf{S}_1\ \theta_s\ \mathsf{S}_2),k_f}{\ddot{\bowtie}} \mathsf{T}_2 \Leftrightarrow \pi_{\{T_{12}\}}\left(\sigma_{(ord\leq k_f)}\left(\omega_{\{Rank(dist)\setminus ord\}}\left(\psi_{\{d(\mathsf{S}_1,\mathsf{S}_2)\setminus dist\}}\left(\mathsf{T}_1 \overset{(\mathsf{S}_1\ \theta_s\ \mathsf{S}_2)}{\bowtie} \mathsf{T}_2\right)\right)\right)\right) \quad (4)$$

The w-join operators differ from the others aforementioned because they allow flexible ways to express both the predicate to be evaluated by the internal similarity join and the cardinality ("how wide") of the answer relation (at most k_f tuples). In addition, they employ internally the sorting concept in a way compatible to the Relational model.

In the same way that the k-nearest neighbor and the similarity range comparison operators are employed in the join predicate to express, respectively, the k-nearest neighbor join and range join, those comparison operators can also be employed in the w-join in order to generate three instances of our new operator class: the k-nearest neighbor wide-join (or k-wide-join in short), the similarity range wide-join and the k-and-range wide-join.

Following Equation 4, when θ_s is the k-nearest neighbor comparison operator, the resulting w-join operator is the *k-nearest neighbor wide-join* $\mathsf{T}_1 \overset{(\mathsf{S}_1\ k\text{-}NN(d,k)\ \mathsf{S}_2),k_f}{\ddot{\bowtie}} \mathsf{T}_2$. It executes a k-nearest neighbor join internally and returns the k_f most similar tuples in the result. Thus, the k-nearest neighbor w-join corresponds to the k-closest neighbor join when $k_f = k$. The k-nearest neighbor wide-join is useful to answer queries where one intends to return the k_f closest pairs of elements from the T_2 which are among the k closest to each element from T_1, effectively reducing the cardinality $k * |\mathsf{T}_1|$ of a k-nearest neighbor join to the cardinality k_f of the k-nearest neighbor w-join. This fact can be observed in Figure 1(h) when compared with the Figure 1(e). Moreover, although the k-wide-join maintain the same cardinality of the k-closest neighbor join, our operator ensures a minimum number of similar elements from the second relation to each element of the first. For instance, in Figure 1(h), at least 2 elements of the second relation are paired to each element of the first whereas the kCN-join (Figure 1(f)) rejects some parts of the search space.

When θ_s is the similarity range comparison operator, it produces the second w-join variation: the *similarity range wide-join* $\mathsf{T}_1 \overset{(\mathsf{S}_1\ Rng(d,\xi)\ \mathsf{S}_2),k_f}{\ddot{\bowtie}} \mathsf{T}_2$. The range w-join is a novel operator that restricts the tuples of the internal join to those which $d(\mathsf{S}_1, \mathsf{S}_2) \leq \xi$, i.e., it performs an internal similarity range join and returns at most the k_f tuples in which the complex attributes are the most similar.

The range w-join is suitable for applications where the similarity between the original tuples must be restricted to a threshold and a maximum cardinality must be enforced, such as query *Q1* presented in Section 1. In order to process *Q1*, suppose a relation containing traditional data about cities and, in addition to them, the geographic coordinate (latitude and longitude) as complex attribute. Notice that the relation containing cities and Capitals presents a complex attribute sampling from the same complex domain (the geo-coordinate), which enables the execution of similarity queries. *Q1* is an example of query application that cannot be handled by the similarity join operators presented in the Section 2. The range join does not process *Q1* once it does not limit the answer to 8 pairs of cities; whereas the k-neighborhood variations ignore the limit of 10 km. The join variation called join-around does not address that query too; it does not ensure that the answer will be limited to 8 pairs. Those operators can process only part of the query, requiring external algorithmic constructions in order to process the conditions that they do not support. The range w-join is able to deal with all predicates demanded in query *Q1*. Therefore, the range w-join has practical appeal and significance. The Figure 1(g) depicts the range w-join. When compared to the similarity range join (Figure 1(d)), one could note a more pronounced restriction of elements composing the answer set, ensuring only the most similar ones.

The third variation of the w-join operator consists of combining both the similarity range and the k-nearest neighbor comparison operators in the internal join of Equation 4, producing the *k-and-range wide-join* $\mathsf{T}_1 \overset{(\mathsf{S}_1\ kRng(d,k,\xi)\ \mathsf{S}_2),k_f}{\ddot{\bowtie}} \mathsf{T}_2$. It combines the tuples of T_1 and T_2 such that, for each tuple $t_1 \in \mathsf{T}_1$, it retrieves the tuples not farther than ξ from T_2 whose attribute S_2 is one of the k nearest to $t_1[\mathsf{S}_1]$, and from the bulk result selects the k_f having the value $t[\mathsf{S}_1]$ nearest to $t[\mathsf{S}_2]$. This operator is suitable for queries demanding more restrictive conditions than those employed in the k-nearest neighbor w-join, as only the tuples within a given threshold will be considered. The k-and-range w-join operator is able to process query *Q2* presented in Section 1. In order to better understand the query solution based on Equation 4, *Q2* is rewritten as: *for the 5 nearest sensors that are at most 0.5 dissimilar to each other, retrieve the 50 pairs of sensors closest among them (supposing that for each pair returned, one sensor would be powered off).* Query *Q2* cannot be answered by the joins discussed in Section 2. The similarity range join ignores the 5 nearest sensors and the overall 50 pairs, whereas the k-nearest neighbor w-join ignores the 0.5 threshold condition. Also, the join-around always retrieves the 1-nearest neighbor but the query demands for a 5-NN. The k-and-range w-join is expressed according to Equation 5. In Equation 5, when $k_f = |\mathsf{T}_1|$ and $k = 1$, the result is the same of a join-around result. Thus, the k-and-range wide-join also generalizes the functionality of the join-around operator.

Algorithm 1: Nested-loop algorithm to process the wide-join operator

Input : The relations T_1 and T_2; the upper bound k; the query condition θ_s;
Output: The relation T_R containing the most similar joined tuples and the similarity value.

```
1 T_R ← ∅;
2 for each tuple t_1 ∈ T_1 do
3     for each tuple t_2 ∈ T_2 do
4         if (t_1[S_1] θ_s t_2[S_2] is true) then
5             dist ← d(S_1, S_2);
6             T_R ← T_R ∪ ({attribs(t_1), attribs(t_2)}, dist);
7 Sort T_R by dist in ord;
8 Select the k tuples from T_R with the smallest ord, project into {T_1, T_2} and return the result;
```

$$\mathsf{T}_1 \overset{(\mathsf{S}_1\ kRng(d,k,\xi)\ \mathsf{S}_2),k_f}{\ddot{\bowtie}} \mathsf{T}_2 \Leftrightarrow \pi_{\{T_{12}\}}\left(\sigma_{(ord\leq k_f)}\left(\omega_{\{Rank(dist)\backslash ord\}}\left(\psi_{\{d(\mathsf{S}_1,\mathsf{S}_2)\backslash dist\}}\left(\left(\mathsf{T}_1 \overset{(\mathsf{S}_1\ k\text{-}NN(d,k)\ \mathsf{S}_2)}{\ddot{\bowtie}} \mathsf{T}_2\right) \bigcap \left(\mathsf{T}_1 \overset{(\mathsf{S}_1\ Rng(d,\xi)\ \mathsf{S}_2)}{\ddot{\bowtie}} \mathsf{T}_2\right)\right)\right)\right)\right) \quad (5)$$

Figure 1(i) shows that the k-and-range wide-join embraces the join-around (Figure 1(c)) and allows to retrieve a result with higher cardinality with respect to the similarity between the joined pairs.

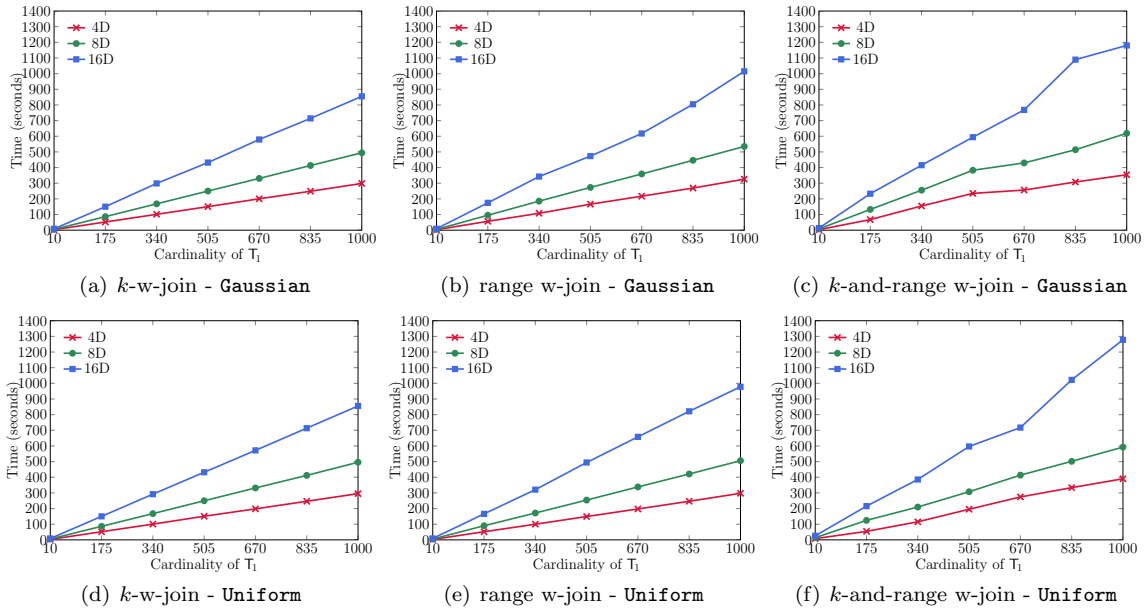
A nested-loop implementation of the similarity wide-join operator is presented as Algorithm 1. The nested-loop in the lines 2−6 executes the internal similarity join. Due to spacing limitations, we do not explore the intricacies of the similarity comparison operator θ_s here. In the line 6, the two tuples meeting the join condition are concatenated and included in the partial result, in addition to the similarity value. Therefore, this line corresponds to the extension operator in Equation 4. Sorting and selection of the nearest tuples already joined are performed in lines 7 and 8, respectively. By presenting this generic algorithm we aim at showing that the wide-join class can be implemented in a RDBMS with an algorithm that executes in feasible time, as shown in the next Section. We note, however, that the basic nested-loop join shown in Algorithm 1 provides plenty of opportunities for algorithmic optimizations, which we will explore in further works.

4. EXPERIMENTS

The main goal of this experimental section is to show the implementation and applicability of the wide-join operator formalized in Section 3. For this purpose, we evaluate the three proposed variations for the wide-join with respect to performance, scalability and query semantics. We employed three real data sets (`BRCities`, `NSF` and `Sensors`) and two synthetic ones (`Gaussian` and `Uniform`).

The `BRCities`[1] data set contains 5,507 Brazilian cities defined by their latitude and longitude values. For this data set, the L_2 metric was applied over the geographic coordinates. The `NSF`[2] data set is composed of 129,000 abstracts and metadata (year, institution etc.) describing the research awards granted by the U.S. National Science Foundation between 1990 and 2003. We employed the Jaccard distance over a complex attribute composed of a bag of words extracted from the abstract to compare `NSF` elements. The `Sensors`[3] data set contains 36 attributes of monthly weather values (precipitation, maximum and minimum temperature, etc.) from 284 Brazilian climate stations for year 2010, as well

[1]BRCities: *ftp://geoftp.ibge.gov.br/organizacao_territorial/localidades/Geomedia_MDB/* Access: Jun 25, 2014
[2]NSF Research Award Abstracts 1990-2003: *http://archive.ics.uci.edu/ml/datasets.html* Access: Jun 25, 2014
[3]Sensors: Obtained from EMBRAPA, not publicly available.



(a) k-w-join - `Gaussian` (b) range w-join - `Gaussian` (c) k-and-range w-join - `Gaussian`

(d) k-w-join - `Uniform` (e) range w-join - `Uniform` (f) k-and-range w-join - `Uniform`

Fig. 2. Scalability measurements: $|\mathsf{T}_2| = 1,000,000$

as the geographic sensor coordinates. A Weighted L_2 metric was employed in this data set, where a weight of 0.25 was assigned to the geo-coordinates and 0.75 to the climate measures. Finally, the `Gaussian` and `Uniform` synthetic data sets consist of 1,000,000 points built following the Gaussian and Uniform distributions, respectively. We generate 3 distinct versions for each distribution considering 4, 8 and 16 dimensions for the data. For both distributions, the L_2 metric was used.

The experiments were performed using a computer built with an Intel® Core™ i7-2600 processor, running at 3.4 GHz, with 8 GB of RAM, on the operating system GNU Lixux - Fedora 20. Following, we present the performance evaluation and semantic results of the wide-join operator.

4.1 Scalability

The scalability experiments evaluated the proposed variations of the wide-join operator regarding execution time and growth of relation T_1 cardinality over the synthetic data sets. For a fixed cardinality of $\mathsf{T}_2 = 1,000,000$ tuples and $k_f = 10$, we set $|\mathsf{T}_1|$ cardinality to {10, 175, 340, 505, 680, 835, 1000} tuples. The results are shown in Figure 2.

Figures 2(a) and 2(d) present the results for k-wide-join ($k = 4$). The graph show that the behavior of every join operator is essentially linear over both cardinality and dimensionality. Also, the results are quite acceptable. For instance, to process $|\mathsf{T}_2| = 1,000,000$ tuples and $|\mathsf{T}_1| = 175$ tuples took about 52.22 seconds in the 4D `Gaussian` data set and 150.15 seconds in the 16D data set; and it took 52.10 seconds in the 4D `Uniform` data set and 150.12 seconds in 16D.

To evaluate the run time of the range wide-join (Figures 2(b) and 2(e)) and k-and-range wide-join (Figures 2(c) and 2(f)), we obtained a radius large enough to retrieve about 1% of T_2 (this threshold distance varies depending on the dimensionality). The additional parameter k in the k-and-range wide-join was also set to 4. As it can be noted in the graphs, they show the same linear behavior of the k-wide-join results, but the k-and-range wide is a little more expensive due to the combination of comparison operators in the inner similarity join. For instance, the range wide-join spent 51.86 seconds to process `Uniform` and 56.66 seconds to process `Gaussian`, considering $|\mathsf{T}_1| = 175$ 4D points. For the k-and-range wide-join, the values of runtime were 54.26 seconds versus 67.72 seconds for the 4D-`Uniform` and 4D-`Gaussian`, respectively.

All graphs of Figure 2 show that the run time always increase linearly, keeping about the same rate at each dimensionality or distribution: a linear growth of the cardinality linearly increases the spent time, even for increasing dimensionality. Therefore, we can say that nor high dimensionality nor data distribution affect the linear behavior of the wide-join operator.

4.2 Semantic evaluation

For the semantic analysis of the wide-join operator, we posed query *Q1* presented in Section 1 over the `BRCities` data set. As aforementioned, none of the join operators presented in the Section 2 can deal with it. However, it can be answered by using the range wide-join as

$$\texttt{Capital} \overset{(GC\ Rng(L_2,10)\ GC),8}{\ddot{\bowtie}} \texttt{City} \Leftrightarrow \pi_{\{T_{12}\}}\Big(\sigma_{(ord\leq 8)}\Big(\omega_{\{Rank(dist)\setminus ord\}}\Big(\psi_{\{d(GC,GC)\setminus dist\}}\Big(\texttt{Capital} \overset{(GC\ Rng(L_2,10)\ GC)}{\bowtie} \texttt{City}\Big)\Big)\Big)\Big)$$

where *GC* represents the geographic coordinate. The execution time for query *Q1* was 0.015 seconds and the result obtained is shown in Table II. Notice that the w-join operators do not restrict retrieving only 1 pair per tuple in T_1 (as shown by the first and eighth pairs) whereas not returning an excessive amount of tuples.

Query *Q2* cannot be answered by the joins discussed in Section 2 too. However, the *k*-and-range wide-join is able to answer it over the `Sensors` data set, for example stating:

$$\texttt{Sensor} \overset{(CD\ kRng(L_2,5,0.5)\ CD),50}{\ddot{\bowtie}} \texttt{Sensor} \Leftrightarrow \pi_{\{T_{12}\}}\Big(\sigma_{(ord\leq 50)}\Big(\omega_{\{Rank(dist)\setminus ord\}}\Big(\psi_{\{d(CD,CD)\setminus dist\}}$$

$$\Big(\Big(\texttt{Sensor} \overset{(CD\ k\text{-}NN(L_2,5)\ CD)}{\ddot{\bowtie}} \texttt{Sensor}\Big) \bigcap \Big(\texttt{Sensor} \overset{(CD\ Rng(L_2,0.5)\ CD)}{\ddot{\bowtie}} \texttt{Sensor}\Big)\Big)\Big)\Big)\Big)$$

where *CD* represents all the climate attributes in addition to the geo-coordinate values. The result of *Q2* was obtained in 0.76 seconds and Table III presents the first eight tuples.

Finally, to complete the semantic analysis with a purely metric data set, we employed the *k*-wide-join operator to answer the following query over the `NSF` data set: *Q3 - Which are the 5 pairs of NSF projects in general closest to projects from the Computer Science Area approved in 1990 considering for each general project the 4 most similar to a Computer Science one?* This query consumed 9.0 minutes to return the answer presented in Table IV. Although the processing spent several minutes, notice that the employed metric is well-known to be computationally very expensive.

5. RELATED WORK

The database literature is rich in studies regarding to the development or improvement of similarity operators, but only a few of them present formal definitions or tackles the integration to a real DBMS. Considering join operations over traditional data, the study of Mishra and Eich [1992] surveys the kinds of join and discuss their main algorithms and applications. Similarly, in Graefe [2012], the main

Table II. Result for query *Q1*

Capital	City
Aracaju-SE	Barra dos Coqueiros-SE
Teresina-PI	Timon-MA
Recife-PE	Olinda-PE
Vitória-ES	Vila Velha-ES
Cuiabá-MT	Várzea Grande-MT
Maceió-AL	Coqueiro Seco-AL
João Pessoa-PB	Bayeux-PB
Aracaju-SE	Nossa Senhora do Socorro-SE

Table III. Result sample for query *Q2*

ID	Sensor's Location	ID	Sensor's Location
282	Dourados-MS	159	Dourados-MS
254	Brasilândia do Sul-PR	238	Formosa do Oeste-PR
186	União da Vitória-PR	54	União da Vitória-PR
217	Ituiutaba-MG	38	Ituiutaba-MG
165	Palotina-PR	51	Guaíra-PR
254	Brasilândia do Sul-PR	165	Palotina-PR
250	Itumirim-MG	104	Macaia-MG
28	Três Marias-MG	11	Três Marias-MG

Table IV. Result of the query *Q3*

NSFID	CS Project title	Year	NSFID	Project title	Year
a9016123	Mathematics Research on Iteration Theories	1990	a9121247	Economics Research on Covering Theorems, Bayesian Cooperative Choice of Strategies and Theory of the Firm	1991
a9001336	Research on Theory of Digital-Analog Nonlinear Computing Structures	1990	a9121247	Economics Research on Covering Theorems, Bayesian Cooperative Choice of Strategies and Theory of the Firm	1991
a9016123	Mathematics Research on Iteration Theories	1990	a9106570	Mathematics Research on Multidimensional Inverse Scattering	1991
a9016123	Mathematics Research on Iteration Theories	1990	a9423154	Research on Flow Karotyping and Sorting of Translocations in Maize	1994
a9016123	Mathematics Research on Iteration Theories	1990	a9401667	Physics Research on Giant Dipole Resonances in Very Heavy Excited Nuclei	1994

focus is on improving the existing techniques employing ordering, specially those based on the sort-merge algorithm. Both researches relate to ours, presenting the formalization of the join operation and providing the base algorithm for our similarity implementation.

The authors of Silva et al. [2013] investigated the similarity select, similarity grouping and similarity join operators, and introduced the fourth similarity join type: the join-around. That work was completed with optimization evaluation and algebraic equivalence rules holding among similarity operators. Our study is different from Silva et al. [2013] once we explore the similarity selection and join operators in order to extend the k-closest neighbor join definition in a new, broader class, that includes the k-closest neighbor join and the join-around and largely extends their usability.

The authors of Jacox and Samet [2008] improved similarity join algorithms for high dimensional data sets using a tree branch pruning technique using trigonometric properties to reduce the search space. Besides being restricted to dimensional data sets, that work only considers the similarity range join, whereas we focus on the k-closest neighbor join and its derivations. Other studies like Bouros et al. [2012] build indexes such as the R-tree to process spatial joins. Spatial structures was also employed in studies of the k-nearest neighbor join [Hjaltason and Samet 1998; Shin et al. 2000]. Those studies are concerned about the similarity variants of the join operation and they do not deal with complex data in metric spaces, whereas we specifically target those data.

Still worth to mention are the so called top-k joins or ranked-joins [Schnaitter and Polyzotis 2008; Zhang et al. 2014]. In this kind of query, there are a score attribute meeting TOR present in the relation and such attribute allows a previously sorting of the tuples. In such way, the selection and join algorithms can benefit from the previously sorting during the processing [Ilyas et al. 2008]. When dealing with similarity-based queries, the data domain does not need to meet TOR and the score attribute is created only after a query center arises, what makes possible to compute the similarity value. In other words, top-k operators rely on an ordering intrinsic to the data, whereas the concept of order in similarity-based operators depends on each query. In our proposal, the tuples are scored only after the join execution, once the sorting relies on the similarity value computed during the join operation, so our solution is able to process queries over data that is either sorted accordingly to the query needs or not.

6. CONCLUSION

Similarity joins are employed in a wide domain of applications, such as data mining, near duplicate detection, data integration, etc. Although the similarity range is the most investigated and the one that executes faster, those based on neighborhood have an ample practical appeal.

In this article, we investigated the properties of the similarity selection and similarity join operators and, gathering evidences that the k-closest neighbor join demands processing beyond that provided by the theoretical join operation definition. Thus, we proposed a new class of binary operators that

embody join and other operations such as sorting, which cannot be supplied by the standard relational operators. To support our new class, we defined the similarity-based **wide-join** operator and identified three derived variations. The first one, the k-nearest neighbor wide-join, relates to the k-nearest neighbor join but providing finer control over the result cardinality, whereas the similarity range wide-join and k-and-range wide-join are novel operators with interesting support for real applications.

We presented an algorithm aiming to showing its usability in real applications. We performed experiments by using real and synthetic data sets and evaluated the scalability of our operators, highlighting they can be seamlessly integrated into RDBMS. Moreover, once they comply with the relational model theory, we provided their formal representation in relational algebra.

As future research, we are working on optimization issues regarding implementing the wide-join operator. Another point is studying the equivalence rules and properties on how it relates with the other operators of the Relational Algebra and with those provided by the current RDBMS.

REFERENCES

Böhm, C. and Krebs, F. The k-Nearest Neighbor Join: turbo charging the KDD process. *Knowledge and Information Systems* 6 (6): 728–749, 2004.

Bouros, P., Ge, S., and Mamoulis, N. Spatial-Textual Similarity Joins. *Proceedings of the VLDB Endowment* 6 (1): 1–12, 2012.

Chaudhuri, S., Ganti, V., and Kaushik, R. A Primitive Operator for Similarity Joins in Data Cleaning. In *Proceedings of the IEEE International Conference on Data Engineering*. Atlanta, USA, pp. 1–12, 2006.

Date, C. J. *SQL and Relational Theory: how to write accurate SQL code*. O' Reilly, 2011.

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*. Arequipa, Peru, pp. 1–10, 2009.

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

Graefe, G. New Algorithms for Join and Grouping Operations. *Computer Science* 27 (1): 3–27, 2012.

Hjaltason, G. R. and Samet, H. Incremental Distance Join Algorithms for Spatial Databases. *SIGMOD Record* 27 (2): 237–248, 1998.

Ilyas, I. F., Beskales, G., and Soliman, M. A. A Survey of Top-k Query Processing Techiniques in Relational Database Systems. *Computing Surveys* 40 (4): 395–420, 2008.

Jacox, E. H. and Samet, H. Metric Space Similarity Joins. *ACM Transactions on Database Systems* 33 (2): 7:1–7:38, 2008.

Liu, Y. and Hao, Z. A kNN Join Algorithm Based on Delta-Tree for High-dimensional Data. *Computer Research and Development* 47 (7): 1234–1243, 2007.

Mishra, P. and Eich, M. H. Join Processing in Relational Databases. *Computing Surveys* 24 (1): 63–113, 1992.

Qin, J., Zhou, X., Wang, W., and Xiao, C. Trie-Based Similarity Search and Join. In *Proceedings of the Joint Extending Database Technology and Database Theory International Conferences*. Genoa, Italy, pp. 392–396, 2013.

Schnaitter, K. and Polyzotis, N. Evaluating Rank Joins with Optimal Cost. In *Proceedings of the ACM Symposium on Principles of Database Systems*. Canada, pp. 43–52, 2008.

Searcóid, M. Ó. *Metric Spaces*. Springer, 2007.

Shin, H., Moon, B., and Lee, S. Adaptive Multi-Stage Distance Join Processing. *SIGMOD Record* 29 (2): 343–354, 2000.

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.

Wang, J., Li, G., and Feng, J. Extending String Similarity Join to Tolerant Fuzzy Token Matching. *ACM Transactions on Database Systems* 39 (1): 7:1–7:45, 2014.

Xiao, C., Wang, W., Lin, X., Yu, X. J., and Wang, G. Efficient Similarity Joins for Near Duplicate Detection. *ACM Transactions on Database Systems* 36 (3): 15:1–15:41, 2011.

Yu, C., Cui, B., Wang, S., and Su, J. Efficient Index-Based kNN Join Processing for High-Dimensional Data. *Information and Software Technology* 49 (4): 332–344, 2007.

Zhang, C., Li, F., and Jestes, J. Efficient Parallel kNN Joins for Large Data in MapReduce. In *Proceedings of the International Conference on Extending Database Technology*. Berlin, Germany, pp. 38–49, 2012.

Zhang, W., Zhan, L., Zhang, Y., Cheema, M. A., and Lin, X. Efficient Top-k Similarity Join Processing Over Multi-Valued Objects. *World Wide Web* 17 (3): 285–309, 2014.