Logical Foundations for Similarity-Based Databases*

Radim Belohlavek[1,2] and Vilem Vychodil[1,2]

[1] T. J. Watson School of Engineering and Applied Science
Binghamton University–SUNY, P.O. Box 6000, Binghamton, NY 13902–6000, USA
[2] Dept. Computer Science, Palacky University, Olomouc
Tomkova 40, CZ-779 00 Olomouc, Czech Republic
rbelohla@binghamton.edu, vychodil@binghamton.edu

Abstract. Extensions of relational databases which aim at utilizing various aspects of similarity and imprecision in data processing are widespread in the literature. A need for development of solid foundations for such extensions, sometimes called similarity-based relational databases, has repeatedly been emphasized by leading database experts. This paper argues that, contrary to what may be perceived from the literature, solid foundations for similarity-based databases can be developed in a conceptually simple way. In this paper, we outline such foundations and develop in detail a part of the the facet related to similarity-based queries and relational algebra. The foundations are close in principle to Codd's foundations for relational databases, yet they account for the main aspects of similarity-based data manipulation. A major implication of the paper is that similarity-based data manipulation can be made an integral part of an extended, similarity-based, relational model of data, rather than glued atop the classic relational model in an ad hoc manner.

1 Introduction

Uncertainty, Similarity-Based Databases, and the Need for Foundations Uncertainty abounds in data management. In the past, numerous studies were devoted to uncertainty and imprecision management in database systems. Yet, the problem of uncertainty and imprecision management is considered a challenge with no satisfactory solutions obtained so far. As an example, the report from the Lowell debate by 25 senior database researchers [1] says "...current DBMS have no facilities for either approximate data or imprecise queries." According to this report, the management of uncertainty and imprecision is one of the six currently most important research directions in database systems. Uncertainty has several facets. In this paper, we address one which gained considerable attention in the past, namely similarity and imprecision and related topics such as approximate/imprecise matches and similarity-based queries.

Sometimes, a simple idea regarding foundational aspects has a groundbreaking impact on a field. Codd's idea regarding the relational database model is an

---

example in the field of database systems: "A hundred years from now, I'm quite sure, database systems will still be based on Codd's relational foundation." [7, p. 1]. The main virtues of Codd's model are due to its reliance on a simple yet powerful mathematical concept of a relation and first-order logic: "The relational approach really is rock solid, owing (once again) to its basis in mathematics and predicate logic." [7, p. 138]. This paper presents a simple idea regarding foundations of similarity-based databases. Put in short, we argue that clear conceptual foundations for similarity-based databases, which have not yet been provided, can be developed in a purely logical and relational way by revisiting the classic Codd's relational model of data.

Most of the literature offers a view in which a similarity-based database is a classic relational database equipped with a "similarity-processing module", which is as if glued atop the classic database and in which the relational database has its conceptual foundations in Codd's relational model, while the "similarity-processing module" is conceptually rooted in the theory of metric spaces. Under such a view, various fundamental classic concepts of relational databases, such as functional or other data dependencies, remain unchanged and thus untouched by similarity considerations, unless a more or less suitable merge of the relational and metric frameworks is proposed for pragmatic reasons which results in a concept with both the relational and the metric component. The concept of a similarity-based query is an example.

Our Approach and Contribution: Contrary to this heterogeneous view, we offer a homogeneous view in which similarity is understood as a particular many-valued, or fuzzy, relation. This way we obtain a purely relational model of similarity-based databases, very much in the spirit of the classic Codd's relational model. From the methodological point of view, we add the concept of similarity to the very basic concept of the classic Codd's model, namely to the concept of a relation (data table), contrary to gluing a "similarity-processing module" atop Codd's model, and obtain a concept of a ranked table over domains with similarities which is illustrated in Tab. 1. As Tab. 1 suggests, we take the classic concept of a relation, add similarity relations to domains (right and bottom part of Tab. 1) and add degrees, which we call ranks, to tuples of the relation (first column of the table in Tab. 1). Note that the similarity relations on numerical domains, such as $year$, can be defined (by users) using absolute distance and a simple scaling function in this example (similarity degree $y_1 \approx_{year} y_2$ of years y_1 and y_2 is $s_y(|y_1 - y_2|)$). The similarities on domains can be seen as an additional subjective information which is supplied by users of the database system. Notice that the ranked table in Tab. 1 can be interpreted as a result of a similarity-based query "select all properties which are sold for approximately \$250,000". In general, every ranked table can be interpreted in such a way—this is an important feature which our model shares with Codd's relational model, see also Remark 1. This naturally leads to a methodical development of our model because all the concepts derived from the concept of a ranked table over domains with similarities can and need to take similarity and approximate matches into account. As a result, degrees of similarity and approximate matches employed

Table 1. Ranked data table over domains with similarities

	name	type	bdrms	year	price	tax
1.0	Miller	Penthouse (P)	2	1994	250,000	2,100
1.0	Ortiz	Single Family (S)	3	1982	250,000	4,350
0.7	Nelson	Ranch (R)	4	1969	320,000	6,500
0.4	Lee	Single Family (S)	4	1975	370,000	8,350
0.2	Kelly	Log Cabin (L)	1	1956	85,000	1,250

$\approx_{type}$	L	P	R	S
L	1	0	0.3	0.2
P	0	1	0	0.4
R	0.3	0	1	0.8
S	0.2	0.4	0.8	1

$$n_1 \approx_{name} n_2 = \begin{cases} 1, & \text{if } n_1 = n_2, \\ 0, & \text{if } n_1 \neq n_2, \end{cases} \qquad b_1 \approx_{bdrms} b_2 = 1 - \min\left(1, \frac{|b_1 - b_2|}{2}\right),$$

$$y_1 \approx_{year} y_2 = s_y(|y_1 - y_2|),$$
$$p_1 \approx_{price} p_2 = s_p(|p_1 - p_2|),$$
$$t_1 \approx_{tax} t_2 = s_t(|t_1 - t_2|).$$

in our model replace 1 (representing exact match, or equality) and 0 (mismatch, unequality) of the classic Codd's model. In the classic model, 1 and 0 are manipulated according to the rules of classical first-order logic (in this sense, Codd's model is based on classical logic). In order to manipulate the degrees of similarity and approximate matches in very much the same way 1 and 0 are manipulated in Codd's model, we employ the recently developed calculus of first-order fuzzy logic [9,10]. This is an important step toward a transparent model with solid logical foundations. Namely, the resultant model is conceptually clear and simple, yet powerful. For example, functional dependencies in the new model naturally take similarities on domains into account and have a simple axiomatization using Armstrong-like rules; relational algebra in the new model automatically offers similarity-based queries; several seemingly non-relational concepts of similarity-based databases turn out to be relational in the new view—a nice example is the top_k query which, contrary to what one can find in the literature, becomes a relational query in the new model, and is thus on the same conceptual level as the other similarity-based and classic relational queries.

Thus, we propose a conservative step back from the currently available view on similarity-based databases which has two facets, namely relational and metric, to a purely relational view. The paper is meant to be a programmatic contribution which outlines the foundations of similarity-based databases, with emphasis on relational algebra and similarity-based queries.

Content of This Paper: In Section 2, we provide an overview of principal concepts from fuzzy logic. Section 3 presents the concept of a ranked table with domains over similarities and overviews some of our previous results. Section 4 presents basic traits of a relational algebra in our model, with emphasis on similarity-based operations. Section 5 surveys the future research.

Previous Work and Related Approaches: Our previous work on this topic includes [2,3], where we presented functional dependencies for domains with similarities, their completeness theorem, a procedure for extracting a non-redundant basis

of such functional dependencies from data, and a sketch of relational algebra with implementation considerations. These papers can be seen as developing technically the model, proposed from a foundational perspective in this paper. Another paper is [4] where we provided a detailed critical comparison of our model with related ones proposed previously in the literature. Namely, the idea of adding similarity to domains in the relational model appeared in several papers, but as a rule in an *ad hoc* manner, without proper logical foundations and comprehensive treatment, see e.g. [5,6,13,14,15] for selected papers.

2   Fuzzy Logic as the Underlying Logic

We use fuzzy logic [9,10] to represent and manipulate truth degrees of propositions like "u is similar to v", cf. also [8]. Fuzzy logic is a many-valued logic with a truth-functional semantics which is an important property because, as a result, it does not depart too much from the principles of classical logic (truth functionality has important mathematical and computational consequences). Fuzzy logic allows us to process (aggregate) truth degrees in a natural way. For instance, consider a query "show all properties which are sold for about \$250,000 and are built around 1970". According to Tab. 1, the property owned by Nelson satisfies subqueries concerning price and year to degrees 0.7 and 0.9, respectively. Then, we combine the degrees using a fuzzy conjunction connective $\otimes$ to get a degree $0.7 \otimes 0.9$ to which the property owned by Nelson satisfies the conjunctive query.

When using fuzzy logic, we have to pick an appropriate scale L of truth degrees (they serve as degrees similarity, degrees to which a tuple matches a query, etc.) and appropriate fuzzy logic connectives (conjunction, implication, etc.). We follow a modern approach in fuzzy logic in that we take an arbitrary partially-ordered scale $\langle L, \leq \rangle$ of truth degrees and require the existence of infima and suprema (for technical reasons, to be able to evaluate quantifiers). Furthermore, we consider an *adjoint pair* of a fuzzy conjunction $\otimes$ and the corresponding fuzzy implication $\rightarrow$ (called residuum) and require some further natural conditions, see [9]. Adjointness is crucial from the point of view of mathematical properties of our model. This way, we obtain a structure $\mathbf{L} = \langle L, \leq, \otimes, \rightarrow, \ldots \rangle$ of truth degrees with logical connectives. Such an approach, even thought rather abstract, is easier to handle theoretically and supports the symbolical character of our model. Moreover, the various particular logical connectives typically used in fuzzy logic applications are particular cases of our structure $\mathbf{L}$. Technically speaking, our structure of truth degrees is assumed to be a complete residuated lattice $\mathbf{L} = \langle L, \wedge, \vee, \otimes, \rightarrow, 0, 1 \rangle$, see [9,10] for details.

A favorite choice of $\mathbf{L}$ is $L = [0,1]$ or a subchain of $[0,1]$. Examples of pairs of important pairs of adjoint operations are Łukasiewicz ($a \otimes b = \max(a + b - 1, 0)$, $a \rightarrow b = \min(1 - a + b, 1)$), and Gödel ($a \otimes b = \min(a, b)$, $a \rightarrow b = 1$ if $a \leq b$, $a \rightarrow b = b$ else). For instance, $0.7 \otimes 0.9 = 0.6$ if $\otimes$ is the Łukasiewicz conjunction; $0.7 \otimes 0.9 = 0.7$ if $\otimes$ is the Gödel one. Further logical connectives are often considered in fuzzy logic, such as the truth-stressing hedges (they model linguistic modifiers such as "very") [10], i.e. particular monotone unary functions $^* : L \rightarrow L$. Two

boundary cases of hedges are (i) identity, i.e. $a^* = a$ ($a \in L$); (ii) globalization: $1^* = 1$, and $a^* = 0$ ($a \neq 1$). Note that a special case of a complete residuated lattice is the two-element Boolean algebra **2** of classical (bivalent) logic.

Having **L**, we define the usual notions: an **L**-set (fuzzy set) A in universe U is a map $A: U \to L$, $A(u)$ being interpreted as "the degree to which u belongs to A". The operations with **L**-sets are defined componentwise. Binary **L**-relations (binary fuzzy relations) between X and Y can be thought of as **L**-sets in the universe $X \times Y$. A fuzzy relation E in U is called reflexive if for each $u \in U$ we have $E(u, u) = 1$; symmetric if for each $u, v \in U$ we have $E(u, v) = E(v, u)$. We call a reflexive and symmetric fuzzy relation a similarity. We often denote a similarity by $\approx$ and use an infix notation, i.e. we write $(u \approx v)$ instead of $\approx(u, v)$.

3 Ranked Tables over Domains with Similarities

This section presents a concept of a ranked table over domains with similarities. We use Y to denote a set of attributes (attribute names) and denote the attributes from Y by $y, y_1, \ldots$; **L** denotes a fixed structure of truth degrees and connectives.

Definition 1. A *ranked data table over domains with similarity relations* (with Y and **L**) is given by

- *domains*: for each $y \in Y$, D_y is a non-empty set (domain of y, set of values of y);
- *similarities*: for each $y \in Y$, $\approx_y$ is a binary fuzzy relation (called similarity) in D_y (i.e. a mapping $\approx_y: D_y \times D_y \to L$) which is reflexive (i.e. $u \approx_y u = 1$) and symmetric ($u \approx_y v = v \approx_y u$);
- *ranking*: for each tuple $t \in \prod_{y \in Y} D_y$, there is a degree $\mathcal{D}(t) \in L$ (called rank of t in $\mathcal{D}$) assigned to t.

Remark 1. (a) $\mathcal{D}$ can be seen as a table with rows and columns corresponding to tuples and attributes, like in Tab. 1. Ranked tables with similarities represent a simple concept which extends the concept of a table (relation) of the classical relational model by two features: similarity relations and ranks.

(b) Formally, $\mathcal{D}$ is a fuzzy relation between domains D_y ($y \in Y$). $t[y]$ denotes a value from D_y of tuple t on attribute y. We require that $\mathcal{D}$ has a finite support, i.e. there is only a finite number of tuples t with a non-zero degree $\mathcal{D}(t)$. If $L = \{0, 1\}$ and if each $\approx_y$ is ordinary equality, the concept of a ranked data table with similarities coincides with that of a data table over set Y of attributes (relation over a relation scheme Y) of a classic model.

(c) Rank $\mathcal{D}(t)$ is interpreted as the degree to which a tuple t satisfies requirements posed by a similarity-query. A table $\mathcal{D}$ representing just stored data, i.e. data prior to querying, has all the ranks equal to 1, i.e. $\mathcal{D}(t) = 1$ for each tuple t. Hence again, $\mathcal{D}$ can be thought of as a result of a query, namely, the query "show all stored data". Therefore, a general interpretation is: a ranked table over domains with similarities is a result of a similarity-based query. Thus in principle, the role of ranked tables over domains with similarities is the same as the role of tables (relations) in the classic relational model.

In [2], we introduced functional dependencies for ranked tables over domains with similarities, their Armstrong-like rules, and completeness theorems. From the technical point of view, this paper demonstrates well the advantage of using (formal) fuzzy logic. The main point is that even though degrees of similarity are taken into account and several aspects thus become more involved (e.g. proofs regarding properties of functional dependencies), functional dependencies over domains with similarities are conceptually simple and feasible both from the theoretical and computational point of view.

4   Relational Algebra and Similarity-Based Queries

In the original Codd's relational model, the relational algebra is based on the calculus of classical relations and on first-order predicate logic. In the same spirit, we introduce a relation algebra for our model with similarities. It will be based on the calculus of fuzzy relations and will have foundations in first-order predicate fuzzy logic [9,10]. A development of the algebra in full is beyond the scope of this paper. Therefore, we present selected groups of relational operations. For each group we present the operations, their properties, and demonstrate them using illustrative examples. At the end, we briefly comment on further topics and results.

4.1   Basic Relational Operations

This group of operations contains our counterparts to the basic Boolean operations of Codd's model—union, intersection, relational difference, etc. These operations emerge naturally because in our model, we actually replace the two-element Boolean algebra by a general scale $\mathbf{L} = \langle L, \otimes, \rightarrow, \wedge, \vee, 0, 1 \rangle$ of truth degrees (residuated lattice). For instance, the classic union $\mathcal{D}_1 \cup \mathcal{D}_2$ of data tables $\mathcal{D}_1$ and $\mathcal{D}_2$ is defined as $\mathcal{D}_1 \cup \mathcal{D}_2 = \{t \,|\, t \in \mathcal{D}_1 \text{ or } t \in \mathcal{D}_2\}$ where "or" stands for Boolean disjunction. Replacing the Boolean disjunction by $\vee$ (supremum from $\mathbf{L}$; $\vee$ is considered a truth function of disjunction in fuzzy logic) we can express the rank $(\mathcal{D}_1 \cup \mathcal{D}_2)(t)$ of t in the union $\mathcal{D}_1 \cup \mathcal{D}_2$ by $\mathcal{D}_1(t) \vee \mathcal{D}_2(t)$, i.e.

$$(\mathcal{D}_1 \cup \mathcal{D}_2)(t) = \mathcal{D}_1(t) \vee \mathcal{D}_2(t). \tag{1}$$

In a similar way, we define two kinds of intersections of ranked data tables.

$$(\mathcal{D}_1 \cap \mathcal{D}_2)(t) = \mathcal{D}_1(t) \wedge \mathcal{D}_2(t), \tag{2}$$

$$(\mathcal{D}_1 \otimes \mathcal{D}_2)(t) = \mathcal{D}_1(t) \otimes \mathcal{D}_2(t). \tag{3}$$

$\mathcal{D}_1 \cap \mathcal{D}_2$ and $\mathcal{D}_1 \otimes \mathcal{D}_2$ are called the $\wedge$-intersection and $\otimes$-intersection of ranked data tables, respectively.

Remark 2. (a) Notice that since both $\mathcal{D}_1$ and $\mathcal{D}_2$ have finite supports (see Remark 1 (b)), the results of union and both the intersections have finite supports as well, i.e., they represent (finite) ranked data tables.

   (b) If $\mathcal{D}_1$ is a result of query Q_1 and $\mathcal{D}_2$ is a result of query Q_2, then $(\mathcal{D}_1 \cup \mathcal{D}_2)(t)$ should be interpreted as "a degree to which t matches Q_1 or t matches Q_2". In

most situations, $\vee$ coincides with maximum. Therefore, $(\mathcal{D}_1 \cup \mathcal{D}_2)(t)$ is just the maximum of $\mathcal{D}_1(t)$ and $\mathcal{D}_2(t)$.

(c) Operations (1)–(3) generalize the classical two-valued operations in the following sense. If the underlying complete residuated lattice **L** equals **2** (two-element Boolean algebra) then (1)–(3) become the Boolean operations.   ■

A new relational operation is obtained based on the residuum $\rightarrow$ which represents a "fuzzy implication". In this case, however, we cannot put

$$(\mathcal{D}_1 \rightarrow \mathcal{D}_2)(t) = \mathcal{D}_1(t) \rightarrow \mathcal{D}_2(t),$$

as the resulting ranked data table may be infinite. Indeed, if the relation scheme of $\mathcal{D}_1$ contains at least one attribute y with an infinite domain D_y, there are infinitely many tuples t such that $(\mathcal{D}_1 \rightarrow \mathcal{D}_2)(t) = 1$. This is due to the fact that if $t[y]$ is not a value of any tuple with a nonzero rank in $\mathcal{D}_1$ then $\mathcal{D}_1(t) = 0$ and consequently $(\mathcal{D}_1 \rightarrow \mathcal{D}_2)(t) = 0 \rightarrow \mathcal{D}_2(t) = 1$. This problem is analogous to the problem of relational complements in the Codd's model [12] where infinite domain can also produce infinite relations. We overcome the problem by considering only tuples whose values belong to so-called active domains of $\mathcal{D}_1$, see [12]. For any ranked data table $\mathcal{D}$ over attributes Y and an attribute $y \in Y$, $adom(\mathcal{D}, y) \subseteq D_y$ is defined as follows

$$adom(\mathcal{D}, y) = \{d \in D_y \,|\, \text{there is tuple } t \text{ such that } \mathcal{D}(t) > 0 \text{ and } t[y] = d\}.$$

Hence, the active domain $adom(\mathcal{D}, y)$ of y in $\mathcal{D}$ is the set of all values of attribute y which appear in $\mathcal{D}$. Furthermore, we let

$$adom(\mathcal{D}) = \prod_{y \in Y} adom(\mathcal{D}, y). \tag{4}$$

Note that $adom(\mathcal{D})$ defined by (4) is a finite ranked data table in its own right. Using (4), we define an active residuum $\mathcal{D}_1 \rightsquigarrow \mathcal{D}_2$ of $\mathcal{D}_1$ in $\mathcal{D}_2$ by

$$(\mathcal{D}_1 \rightsquigarrow \mathcal{D}_2)(t) = \mathcal{D}_1(t) \rightarrow \mathcal{D}_2(t), \quad \text{for each } t \in adom(\mathcal{D}), \tag{5}$$

and $(\mathcal{D}_1 \rightsquigarrow \mathcal{D}_2)(t) = 0$ otherwise. Notice that $(\mathcal{D}_1 \rightsquigarrow \mathcal{D}_2)(t)$ can be seen as a degree to which it is true that "if t matches Q_1 then t matches Q_2". The following proposition shows basic properties of the operations defined so far.

Proposition 1. *For any ranked data tables* $\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_3$,

$$\mathcal{D}_1 \otimes (\mathcal{D}_2 \cup \mathcal{D}_3) = (\mathcal{D}_1 \cup \mathcal{D}_2) \otimes (\mathcal{D}_1 \cup \mathcal{D}_3), \tag{6}$$

$$\mathcal{D}_1 \rightsquigarrow (\mathcal{D}_2 \cap \mathcal{D}_3) = (\mathcal{D}_1 \rightsquigarrow \mathcal{D}_2) \cap (\mathcal{D}_1 \rightsquigarrow \mathcal{D}_3), \tag{7}$$

$$adom(\mathcal{D}_1 \cap \mathcal{D}_2) \cap ((\mathcal{D}_1 \cup \mathcal{D}_2) \rightsquigarrow \mathcal{D}_3) = (\mathcal{D}_1 \rightsquigarrow \mathcal{D}_3) \cap (\mathcal{D}_2 \rightsquigarrow \mathcal{D}_3), \tag{8}$$

$$adom(\mathcal{D}_1 \otimes \mathcal{D}_2) \cap (\mathcal{D}_1 \rightsquigarrow (\mathcal{D}_2 \rightsquigarrow \mathcal{D}_3)) = (\mathcal{D}_1 \otimes \mathcal{D}_2) \rightsquigarrow \mathcal{D}_3 \tag{9}$$

Proof. For illustration, we prove (6). By definition, $(\mathcal{D}_1 \otimes (\mathcal{D}_2 \cup \mathcal{D}_3))(t) = \mathcal{D}_1(t) \otimes (\mathcal{D}_2 \cup \mathcal{D}_3)(t) = \mathcal{D}_1(t) \otimes (\mathcal{D}_2(t) \vee \mathcal{D}_3(t))$. Using $a \otimes \bigvee_i b_i = \bigvee_i (a \otimes b_i)$ which is true in any complete residuated lattice, we get $(\mathcal{D}_1 \otimes (\mathcal{D}_2 \cup \mathcal{D}_3))(t) = (\mathcal{D}_1(t) \otimes \mathcal{D}_2(t)) \vee (\mathcal{D}_1(t) \otimes \mathcal{D}_3(t)) = (\mathcal{D}_1 \otimes \mathcal{D}_2)(t) \vee (\mathcal{D}_1 \otimes \mathcal{D}_3)(t) = ((\mathcal{D}_1 \otimes \mathcal{D}_2) \cup (\mathcal{D}_1 \otimes \mathcal{D}_3))(t)$. (7) follows from $a \rightarrow \bigwedge_i b_i = \bigwedge_i (a \rightarrow b_i)$. In order to prove (8) and (9) we have to take care about different active domains on both sides of the equalities. Details are postponed to the full version of this paper.   □

The only unary Boolean operation that is considered in the usual model is the active complement of a data table. In our model, there are other nontrivial unary operations like hedges (see Section 2) and shifts. For any ranked data table $\mathcal{D}$, $0 \neq a \in L$, and a unary truth function $^*: L \to L$, we define $\sim\mathcal{D}$, $\mathcal{D}^*$, and $a \rightsquigarrow \mathcal{D}$ by

$$(\sim\mathcal{D})(t) = \mathcal{D}(t) \to 0, \qquad \text{for each } t \in adom(\mathcal{D}), \qquad (10)$$

$$(a \rightsquigarrow \mathcal{D})(t) = a \to \mathcal{D}(t), \qquad \text{for each } t, \qquad (11)$$

$$(\mathcal{D}^*)(t) = \mathcal{D}(t)^*, \qquad \text{for each } t. \qquad (12)$$

$\sim\mathcal{D}$ and $a \rightsquigarrow \mathcal{D}$ are called an active complement and an a-shift of $\mathcal{D}$, respectively. $\mathcal{D}^*$ is a data table with ranks obtained by applying * to the ranks of the original data table.

Remark 3. (a) Notice that $\sim\mathcal{D}$ equals $\mathcal{D} \rightsquigarrow \emptyset_{\mathcal{D}}$ where $\emptyset_{\mathcal{D}}$ is an empty ranked data table, i.e. we have expressed complementation based on residuation and an empty table. This technique is widely used in the two-valued logic as well (e.g., a formula $\neg\varphi$ is logically equivalent to $\varphi \Rightarrow \overline{0}$ where $\overline{0}$ is a nullary connective representing falsity).

(b) Residuated lattices are, in general, weaker structures than Boolean algebras, i.e. not all laws satisfied by Boolean algebras are satisfied by residuated lattices. For instance, the law of excluded middle $a \vee (a \to 0) = 1$ is satisfied by a residuated lattice **L** iff **L** a Boolean algebra. Therefore, some properties of the operations from the original Codd's model are not preserved in our model. ∎

Example 1. Since $a \to b = 1$ iff $a \leq b$, $(a \rightsquigarrow \mathcal{D})(t)$ defined by (11) can be interpreted as a degree to which "t matches (a query) at least to degree a". For instance, if we consider $0.7 \rightsquigarrow \mathcal{D}$ with $\mathcal{D}$ from Tab. 1, the resulting table

	name	type	bdrms	year	price	tax
1.0	Miller	Penthouse	2	1994	250,000	2,100
1.0	Nelson	Ranch	4	1969	320,000	6,500
1.0	Ortiz	Single Family	3	1982	250,000	4,350
0.7	Lee	Single Family	4	1975	370,000	8,350
0.5	Kelly	Log Cabin	1	1956	85,000	1,250

represents the answer to query the "show all properties for sale where *price* = 250,000 at least to degree 0.7, i.e. where *price* is more or less equal to 250,000." Shifts play an important role in our model because they enable us to select tuples with sufficient large ranks (in a logically clean way). ∎

4.2 Derived Relational Operations

In this section we briefly discuss operations which can be derived from those presented in the previous section. Recall that the original Codd's model considers relational difference $\mathcal{D}_1 - \mathcal{D}_2$ such that $t \in \mathcal{D}_1 - \mathcal{D}_2$ iff $t \in \mathcal{D}_1$ and $t \notin \mathcal{D}_2$. In our setting there are multiple choices to define a generalization of this operation. Our intention is to chose the "best definition" which behaves naturally with respect to the other operations, the main criteria being expressiveness and feasibility of the resulting algebra.

In case of $\mathcal{D}_1 - \mathcal{D}_2$, we can proceed as follows: in the ordinary model, $t \in \mathcal{D}_1 - \mathcal{D}_2$ iff it is not true that if $t \in \mathcal{D}_1$ then $t \in \mathcal{D}_2$. Thus, we can define

$$\mathcal{D}_1 - \mathcal{D}_2 = \sim(\mathcal{D}_1 \rightsquigarrow \mathcal{D}_2), \tag{13}$$

i.e., $(\mathcal{D}_1 - \mathcal{D}_2)(t) = (\mathcal{D}_1(t) \rightarrow \mathcal{D}_2(t)) \rightarrow 0$ for any t (as one can check, active domain can be disregarded in this case). Clearly, if **L** is **2**, (13) is equivalent to the ordinary difference of relations.

Our relational algebra contains operations which either have no counterparts within the classic operations or the counterparts are trivial, (e.g. a-shifts introduced in Section 4.1). Another example of a useful operation is a so-called a-cut. For a ranked table $\mathcal{D}$ and $a \in L$, an a-cut of $\mathcal{D}$ is a ranked table ${}^a\mathcal{D}$ defined by

$$({}^a\mathcal{D})(t) = \begin{cases} 1, & \text{if } \mathcal{D}(t) \geq a, \\ 0, & \text{otherwise.} \end{cases} \tag{14}$$

That is, ${}^a\mathcal{D}$ is a ("non-ranked") table which contains just the tuples of $\mathcal{D}$ with ranks greater or equal to a. This is a useful operation for manipulation with ranked tables as it allows the user to select only a part of a query result given by threshold a. An a-cut is indeed a derived operation because ${}^a\mathcal{D} = (a \rightsquigarrow \mathcal{D})^*$ where * is globalization, see Section 2.

Note that in combination with intersection, we can use a-cut to get the part of $\mathcal{D}$ with ranks at least a. Namely, we can put $above(\mathcal{D}, a) = \mathcal{D} \cap {}^a\mathcal{D}$. Thus,

$$(above(\mathcal{D}, a))(t) = \begin{cases} \mathcal{D}(t), & \text{if } \mathcal{D}(t) \geq a, \\ 0, & \text{otherwise.} \end{cases}$$

Example 2. $above(\mathcal{D}, 0.7)$ in case of $\mathcal{D}$ from Tab. 1 is the following:

	name	type	bdrms	year	price	tax
1.0	Miller	Penthouse	2	1994	250,000	2,100
1.0	Ortiz	Single Family	3	1982	250,000	4,350
0.7	Nelson	Ranch	4	1969	320,000	6,500

The result of ${}^{0.7}\mathcal{D}$ is the same as $above(\mathcal{D}, a)$ except for the ranks all being 1. ∎

Another operation which is derivable in our model is top_k which has gained considerable interest recently, see [8] and also [11]. $top_k(\mathcal{D})$ contains first k ranked tuples according to rank ordering of $\mathcal{D}$. Therefore, the result of $top_k(\mathcal{D})$ is a ranked data table containing k best matches of a query (if there are less than k ranks in $\mathcal{D}$ then $top_k(\mathcal{D}) = \mathcal{D}$; and $top_k(\mathcal{D})$ includes also the tuples with rank equal to the rank of the k-th tuple). It can be shown that top_k is a derivable operation in our relational model. Namely,

$$(top_k(\mathcal{D}))(t) = \mathcal{D}(t) \otimes (Q_{<k}t')\big(\neg(\mathcal{D}(t') \rightarrow \mathcal{D}(t))^* \otimes (\mathcal{D}(t) \rightarrow \mathcal{D}(t'))^*\big)$$

where * is globalization and $(Q_{<k}t')$ is a quantifier "there are at most k tuples t'" (details will be presented in a full version of this paper).

Remark 4. $top_k(\mathcal{D})$ naturally emerges as a derived relational operation. This illustrates the versatility of our approach.

4.3   Projections of Ranked Data Tables

The role of projection in our model is the same as in the Codd's model. Projection produces a ranked data table with tuples containing a subset of attributes from the original ranked data table. Notice that we may have a situation such that $\mathcal{D}(t) \neq \mathcal{D}(t')$ and both tuples t and t' have common values on all attributes from $A \subseteq Y$, i.e. $t[A] = t'[A]$. Therefore, the rank of t in the projection of $\mathcal{D}$ onto A will be computed as a supremum of ranks of all tuples in $\mathcal{D}$ which agree with t on the attributes from A. Thus, the projection $\pi_A(\mathcal{D})$ of $\mathcal{D}$ onto $A \subseteq Y$ is defined by

$$(\pi_A(\mathcal{D}))(t) = \bigvee\nolimits_{s[A]=t} \mathcal{D}(s), \quad \text{for each } t \in \prod\nolimits_{y \in A} D_y. \tag{15}$$

As in the classic model, if two (or more) projections are applied in a row, the last one subsumes the previous ones. Therefore, for any $A \subseteq B \subseteq Y$,

$$\pi_A(\pi_B(\mathcal{D})) = \pi_A(\mathcal{D}).$$

The following proposition shows relationship of projections w.r.t. our basic relational operations.

Proposition 2. *For any $\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}, A \in L^Y, a \in L$,*

$$\pi_A(\mathcal{D}_1 \cup \mathcal{D}_2) = \pi_A(\mathcal{D}_1) \cup \pi_A(\mathcal{D}_2), \qquad \pi_A(\mathcal{D}^*) \subseteq \pi_A(\mathcal{D}), \tag{16}$$

$$\pi_A(\mathcal{D}_1 \cap \mathcal{D}_2) \subseteq \pi_A(\mathcal{D}_1) \cap \pi_A(\mathcal{D}_2), \qquad \pi_A(a \rightsquigarrow \mathcal{D}) \subseteq a \rightsquigarrow \pi_A(\mathcal{D}), \tag{17}$$

$$\pi_A(\mathcal{D}_1 \otimes \mathcal{D}_2) \subseteq \pi_A(\mathcal{D}_1) \otimes \pi_A(\mathcal{D}_2), \qquad \pi_A(a \otimes \mathcal{D}) = a \otimes \pi_A(\mathcal{D}). \tag{18}$$

Proof. We prove (18). Observe that $(\pi_A(\mathcal{D}_1 \otimes \mathcal{D}_2))(t) = \bigvee_{s[A]=t}(\mathcal{D}_1 \otimes \mathcal{D}_2)(s) = \bigvee_{s[A]=t}(\mathcal{D}_1(s) \otimes \mathcal{D}_2(s)) \leq \bigvee_{s_1[A]=t} \bigvee_{s_2[A]=t}(\mathcal{D}_1(s_1) \otimes \mathcal{D}_2(s_2)) = (\bigvee_{s_1[A]=t} \mathcal{D}_1(s_1) \otimes \bigvee_{s_2[A]=t} \mathcal{D}_2(s_2)) = (\pi_A(\mathcal{D}_1))(t) \otimes (\pi_A(\mathcal{D}_2))(t) = (\pi_A(\mathcal{D}_1) \otimes \pi_A(\mathcal{D}_2))(t)$. The rest can be proved by similar arguments. $\qquad\square$

4.4   Similarity-Based Selection

Relational operations considered in previous sections did not utilize similarities of values on domains. We now turn our attention to the similarity-based selection which is a counterpart to ordinary selection. Our selection can be seen as an operation which selects from a data table all tuples which approximately match a given condition. The condition can be in the form of an equality "$y = c$", saying that the value of attribute y should be similar (i.e., approximately equal) to c. Clearly, a tuple t matches $y = c$ to a degree to which $t[y]$ is similar to c, which equals $t[y] \approx_y c$. This is where the similarity $\approx_y$ on the domain of y comes into play.

In general, the selection formula which poses a restriction on tuples may be more complex than an atomic equality. In this paper, we will consider selection formulas as follows: each expression "$p = q$" where p, q are variables or (constants for) values (in the same domains) is a selection formula; each (constant for) a truth degree $a \in L$ is a selection formula; if φ and ψ are selection formulas then $(\varphi \ \& \ \psi)$, $(\varphi \sqcup \psi)$, $(\varphi \sqcap \psi)$, $(\varphi \Rightarrow \psi)$, and φ^* are selection formulas.

Since fuzzy logic [10] is truth-functional, there is a regular way to define a degree $||\varphi||_t \in L$ *to which tuple t matches a selection formula φ* as follows: if φ is a (constant for) a truth degree a then $||\varphi||_t = a$; if φ equals "$y = c$" where y is a variable and $c \in D_y$, $||\varphi||_t = t[y] \approx_y c$ (analogously for φ being an equality of the form *variable = variable, constant = variable, constant = constant*); if φ equals $(\vartheta \Rightarrow \chi)$ then $||\varphi||_t = ||\vartheta||_t \rightarrow ||\chi||_t$ and similarly for other connectives $\&, \sqcup, \sqcap$ and their truth functions $\otimes, \vee, \wedge$.

Remark 5. Our approach strictly adheres to mathematical fuzzy logic. Selection formulas are defined as well-formed formulas. Such formulas are a part of the syntax of our relational algebra. In order to interpret such formulas, we supply a semantic component—a tuple of values. Therefore, we have the usual distinction between a syntax (a formula) and a semantics (interpretation) as we know it from the classical Boolean logic. The only difference is in that we use a more general structure of truth degrees and, therefore, the degree $||\varphi||_t$ to which t matches φ may be an arbitrary value in L. For instance, $||\varphi||_t = 1$ means a "perfect match", $||\varphi||_t = 0.9$ is an "almost perfect match", $||\varphi||_t = 0.7$ can be seen as a "more or less good match", $||\varphi||_t = 0$ is "not a match at all". The strong connection to mathematical fuzzy logic we have allows us to utilize both the symbolic level of our calculus where we can deal with formulas as with symbolic expressions and the numerical level enabling us to deal with ranks which are directly interpretable as numerical degrees of matches. ∎

The selection $\sigma_\varphi(\mathcal{D})$ of tuples in $\mathcal{D}$ matching φ is defined by

$$\sigma_\varphi(\mathcal{D}) = \mathcal{D}(t) \otimes ||\varphi||_t. \tag{19}$$

Considering $\mathcal{D}$ as a result of query Q, the rank of t in $\sigma_\varphi(\mathcal{D})$ can be interpreted as a degree to which "t matches the query Q and in addition it matches the selection formula φ". Hence, if $\mathcal{D}$ is a table where all rows have ranks 1, $(\sigma_\varphi(\mathcal{D}))(t) = ||\varphi||_t$, i.e. it is a degree to which "t matches condition posed by φ".

Selection satisfies several properties of which are analogous to properties of the classic selection. Due to the limited scope, we present just the following.

Proposition 3. *For any $\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}, A \subseteq Y$, and φ*

$$\sigma_\varphi(\mathcal{D}_1 \cup \mathcal{D}_2) = \sigma_\varphi(\mathcal{D}_1) \cup \sigma_\varphi(\mathcal{D}_2), \qquad \sigma_\varphi(\mathcal{D}_1 \cap \mathcal{D}_2) \subseteq \sigma_\varphi(\mathcal{D}_1) \cap \mathcal{D}_2, \tag{20}$$

$$\sigma_\varphi(\mathcal{D}_1 \otimes \mathcal{D}_2) = \mathcal{D}_1 \otimes \sigma_\varphi(\mathcal{D}_2), \qquad \sigma_\varphi(\mathcal{D}_1 \otimes \mathcal{D}_2) = \sigma_\varphi(\mathcal{D}_1) \otimes \mathcal{D}_2. \tag{21}$$

Proof. In case of $\sigma_\varphi(\mathcal{D}_1 \cap \mathcal{D}_2) \subseteq \sigma_\varphi(\mathcal{D}_1) \cap \mathcal{D}_2$, $\sigma_\varphi(\mathcal{D}_1 \cap \mathcal{D}_2)(t) = (\mathcal{D}_1 \cap \mathcal{D}_2)(t) \otimes ||\varphi||_t = (\mathcal{D}_1(t) \wedge \mathcal{D}_2(t)) \otimes ||\varphi||_t$. Now, $(\mathcal{D}_1(t) \wedge \mathcal{D}_2(t)) \otimes ||\varphi||_t \leq \mathcal{D}_1(t) \otimes ||\varphi||_t$ and $(\mathcal{D}_1(t) \wedge \mathcal{D}_2(t)) \otimes ||\varphi||_t \leq \mathcal{D}_2(t)$. Putting the latter inequalities together, we get $(\mathcal{D}_1(t) \wedge \mathcal{D}_2(t)) \otimes ||\varphi||_t \leq (\mathcal{D}_1(t) \otimes ||\varphi||_t) \wedge \mathcal{D}_2(t) = (\sigma_\varphi(\mathcal{D}_1) \cap \mathcal{D}_2)(t)$. The other equalities can be shown by analogous arguments. □

Our selection preserves important properties with respect to nested selections and projections [12]. It can be shown that for any $\mathcal{D}, A \subseteq Y, \varphi, \psi$, and ϑ which does not contain attributes from A,

$$\pi_A(\sigma_\vartheta(\mathcal{D})) = \sigma_\vartheta(\pi_A(\mathcal{D})),$$

$$\sigma_\varphi(\sigma_\psi(\mathcal{D})) = \sigma_{\varphi \& \psi}(\mathcal{D}) = \sigma_{\psi \& \varphi}(\mathcal{D}) = \sigma_\psi(\sigma_\varphi(\mathcal{D})).$$

The latter equality can be extended to arbitrary many selections:

$$\sigma_{\varphi_1 \& \varphi_2 \& \cdots \& \varphi_k}(\mathcal{D}) = \sigma_{\varphi_1}(\sigma_{\varphi_2}(\cdots(\sigma_{\varphi_k}(\mathcal{D})\cdots)),$$

saying that the selection of tuples matching the conjunctive formula $\varphi_1 \& \varphi_2 \& \cdots \& \varphi_k$ tantamount to performing a sequence of selections matching φ_i. If each φ_i is an equality $y_i = d_i$ then, by previous observations,

$$(\sigma_{y_1=d_1 \& \cdots \& y_k=d_k}(\mathcal{D}))(t) = \mathcal{D}(t) \otimes (t[y_1] \approx_{y_1} d_1) \otimes \cdots \otimes (t[y_k] \approx_{y_k} d_k),$$

i.e. it is a degree of "t matches Q and value of t in y_1 is similar to d_1 and $\cdots$ and value of t in y_k is similar to d_k".

Example 3. Let $\mathcal{D}_2$ be a data table of possible buyers of properties:

	name	type	bdrms	price	score
1.0	Adams	Single Family	3	250,000	628
1.0	Black	Single Family	3	325,000	769
1.0	Chang	Residential	4	300,000	535
1.0	Davis	Condominium	1	200,000	567
1.0	Enke	Ranch	3	240,000	635
1.0	Flores	Penthouse	2	200,000	659

The attributes *price* and credit *score* contain the amount which the buyer is willing to spend and his FICO credit score, respectively. Let us assume that we use a reasonable similarity on the domain of credit scores which agrees with how agents perceive different credit score values (i.e., we assume that the similarities are user-supplied). The result of the selection of tuples with credit *score* approximately equal to 600 is $\sigma_{score=600}(\mathcal{D})$ and it may be the following

	name	type	bdrms	price	score
0.9	Adams	Single Family	3	250,000	628
0.9	Davis	Condominium	1	200,000	567
0.9	Enke	Ranch	3	240,000	635
0.8	Chang	Residential	4	300,000	535
0.8	Flores	Penthouse	2	200,000	659
0.4	Black	Single Family	3	325,000	769

Selection $\sigma_{score=600 \,\&\, type=\text{"Single Family"} \,\&\, price=250,000}(\mathcal{D})$ uses a compound formula containing three approximate equalities connected by conjunctions. The result of such a selection of tuples with credit *score* approximately 600, housing *type* similar to "Single Family", and *price* approximately 250,000 is

	name	type	bdrms	price	score
0.9	Adams	Single Family	3	250,000	628
0.6	Enke	Ranch	3	240,000	635
0.2	Chang	Residential	4	300,000	535

Selection $top_4(\sigma_{(0.7\Rightarrow(score=600)) \,\&\, type=\text{"Single Family"} \,\&\, price=250,000}(\mathcal{D}))$ yields

	name	type	bdrms	price	score
1.0	Adams	Single Family	3	250,000	628
0.7	Enke	Ranch	3	240,000	635
0.4	Chang	Residential	4	300,000	535
0.3	Black	Single Family	3	325,000	769

Notice that in the last case, our requirement on credit *score* has been relaxed using a 0.7-shift. Indeed, the subformula $0.7 \Rightarrow (score = 600)$ can be read "*score* is similar to 600 at least to degree 0.7". As we can see, the ranks in the last table are higher than the ranks in the previous table which corresponds to the fact that $0.7 \Rightarrow (score = 600)$ represents a weaker constraint than $score = 600$. This example demonstrates that truth degrees can be used as natural weights in queries allowing us to put more emphasis on particular requirements. Note that in addition to similarity, we may consider other approximate comparators like "approximately greater", etc. ∎

4.5 Similarity-Based Join

A similarity-based join is a fundamental operation used to combine information from two tables into a single one based on similarity of values of two columns or, in a more general setting, on a comparator based on a selection formula. The Codd's model has several notions of a join. In our case, the family of operations generalizing joins is even wider. Because of the limited scope, we discuss one particular join operation generalizing the classic theta-join.

Recall that the ordinary theta-join can be seen as a selection from a Cartesian product of two data tables. Therefore, one option is to define a join in our setting based on this property. For ranked data tables $\mathcal{D}_1$ and $\mathcal{D}_2$ with disjoint relation schemes we define a Cartesian product $\mathcal{D}_1 \times \mathcal{D}_2$ by

$$(\mathcal{D}_1 \times \mathcal{D}_2)(st) = \mathcal{D}_1(s) \otimes \mathcal{D}_2(t), \quad \text{where } \mathcal{D}_1(s) > 0 \text{ and } \mathcal{D}_2(t) > 0. \qquad (22)$$

If $\mathcal{D}_1$ and $\mathcal{D}_2$ are results of queries Q_1 and Q_2, respectively, the rank of st in $\mathcal{D}_1 \times \mathcal{D}_2$ is a degree to which "s matches Q_1 and t matches Q_2". Obviously, if **L** is the two-element Boolean algebra, $\mathcal{D}_1 \times \mathcal{D}_2$ becomes the ordinary Cartesian product of relations.

Using (22), we can define a join $\mathcal{D}_1 \bowtie_\varphi \mathcal{D}_2$ of $\mathcal{D}_1$ and $\mathcal{D}_2$ by selection formula φ which may contain attributes from both $\mathcal{D}_1$ and $\mathcal{D}_2$ as follows

$$\mathcal{D}_1 \bowtie_\varphi \mathcal{D}_2 = \sigma_\varphi(\mathcal{D}_1 \times \mathcal{D}_2). \qquad (23)$$

Using previous definitions, (23) is equivalent to

$$(\mathcal{D}_1 \bowtie_\varphi \mathcal{D}_2)(st) = \mathcal{D}_1(s) \otimes \mathcal{D}_2(t) \otimes ||\varphi||_{st}. \qquad (24)$$

If φ is in the form of $y_1 = y_2$ where y_1 and y_2 are attributes from $\mathcal{D}_1$ and $\mathcal{D}_2$ defined on the same domain with similarity then

$$(\mathcal{D}_1 \bowtie_{y_1=y_2} \mathcal{D}_2)(st) = \mathcal{D}_1(s) \otimes \mathcal{D}_2(t) \otimes (s[y_1] \approx_{y_1} t[y_2]), \qquad (25)$$

which can be seen as a generalization of the classic equi-join.

Remark 6. Note that unlike the classic equi-join, $\mathcal{D}_1 \bowtie_{y_1=y_2} \mathcal{D}_2$ includes both the attributes y_1 and y_2 because we are performing similarity-based join, i.e. the join runs not only over tuples with equal values of y_1 and y_2 but also over tuples with similar values. This way, we obtain joins over values which may also be interesting but which are not covered by the classic join. There are ways to introduce joins using just one attribute for both y_1 and y_2, see [3] for details.

Example 4. Let $\mathcal{D}_1$ be the ranked data table of buyers from Example 3 and let $\mathcal{D}_2$ be the ranked table of sellers from Tab. 1 prior to querying, i.e. with all nonzero ranks equal to 1. If we wish to reveal which buyers may want properties according to their type and price, we can use the following similarity-based join $top_5(\mathcal{D}_1 \bowtie_{type1=type2\ \&\ price1=price2} \mathcal{D}_2)$ projected to interesting attributes:

	name1	price1	price2	type1	type2	bdrms2
1.0	Adams	250,000	250,000	Single Family	Single Family	3
0.8	Black	325,000	370,000	Single Family	Single Family	4
0.8	Flores	200,000	250,000	Penthouse	Penthouse	2
0.7	Black	325,000	320,000	Single Family	Ranch	4
0.7	Enke	240,000	250,000	Ranch	Single Family	3

As we can see, our best 5 matches contain one perfect match and 4 matches which, according to their ranks, can be considered as almost perfect or very good. With data like these, it does not make much sense to consider the usual equality joins because they may often produce empty data tables although there are interesting pairs of tuples which are almost joinable. For instance the query $top_5(\mathcal{D}_1 \bowtie_{type1=type2\ \&\ price1=price2\ \&\ score=750} \mathcal{D}_2)$ projected to interesting attributes gives

	name1	price1	price2	type1	type2	score
0.7	Black	325,000	370,000	Single Family	Single Family	769
0.6	Adams	250,000	250,000	Single Family	Single Family	628
0.6	Black	325,000	320,000	Single Family	Ranch	769
0.5	Black	325,000	250,000	Single Family	Single Family	769
0.5	Flores	200,000	250,000	Penthouse	Penthouse	659

Hence, there is no exact match although `Black` should be seen as a potential good candidate for buying the corresponding property. In this example, comparators other than "=" may be useful. When posing requirements on `score`, one usually wants it to be approximately greater (i.e., greater with a tolerance) than a value rather than similar to a value. Because of the limited scope of this paper, we are not going to discuss such comparators here but they can also be introduced in our model. ∎

4.6   Domain Calculus, Tuple Calculus, and Equivalence Theorem

One of the fundamental results regarding the classic relational algebra is the so-called equivalence theorem which says that the expressive power of the algebra is equivalent to that of a domain calculus as well as to that of a tuple calculus. We obtained a counterpart to this result in our framework. Due to the limited scope, we will present it in a full version of this paper.

5   Conclusions and Further Research

We outlined logical foundations for similarity-based databases, with emphasis on relational algebra and similarity-based queries. Our future research will be

directed toward the development of further foundational issues in our model including standard issues from relational databases (further data dependencies, redundancy, normalization and design of databases, optimization issues, etc.), with a particular focus on similarity-related aspects. In addition, we will continue [3] and develop a prototype implementation of our model by means of existing relational database management systems.

References

1. Abiteboul, S., et al.: The Lowell database research self-assessment. Comm. ACM 48(5), 111–118 (2005)
2. Bělohlávek, R., Vychodil, V.: Data tables with similarity relations: functional dependencies, complete rules and non-redundant bases. In: Li Lee, M., Tan, K.-L., Wuwongse, V. (eds.) DASFAA 2006. LNCS, vol. 3882, pp. 644–658. Springer, Heidelberg (2006)
3. Belohlavek, R., Opichal, S., Vychodil, V.: Relational algebra for ranked tables with similarities: properties and implementation. In: Berthold, M.R., Shawe-Taylor, J., Lavrač, N. (eds.) IDA 2007. LNCS, vol. 4723, pp. 140–151. Springer, Heidelberg (2007)
4. Belohlavek, R., Vychodil, V.: Codd's relational model from the point of view of fuzzy logic. J. Logic and Computation (to appear)
5. Bosc, P., Kraft, D., Petry, F.: Fuzzy sets in database and information systems: status and opportunities. Fuzzy Sets and Syst. 156, 418–426 (2005)
6. Buckles, B.P., Petry, F.E.: Fuzzy databases in the new era. In: ACM SAC 1995, Nashville, TN, pp. 497–502 (1995)
7. Date, C.J.: Database Relational Model: A Retrospective Review and Analysis. Addison Wesley, Reading (2000)
8. Fagin, R.: Combining fuzzy information: an overview. ACM SIGMOD Record 31(2), 109–118 (2002)
9. Gottwald, S.: Mathematical fuzzy logics. Bulletin for Symbolic Logic 14(2), 210–239 (2008)
10. Hájek, P.: Metamathematics of Fuzzy Logic. Kluwer, Dordrecht (1998)
11. Li, C., Chang, K.C.-C., Ilyas, I.F., Song, S.: RankSQL: Query Algebra and Optimization for Relational top-k queries. In: ACM SIGMOD 2005, pp. 131–142
12. Maier, D.: The Theory of Relational Databases. Computer Science Press, Rockville (1983)
13. Prade, H., Testemale, C.: Generalizing database relational algebra for the treatment of incomplete or uncertain information and vague queries. Inf. Sci. 34, 115–143 (1984)
14. Raju, K.V.S.V.N., Majumdar, A.K.: Fuzzy functional dependencies and lossless join decomposition of fuzzy relational database systems. ACM Trans. Database Systems 13(2), 129–166 (1988)
15. Takahashi, Y.: Fuzzy database query languages and their relational completeness theorem. IEEE Trans. Knowledge and Data Engineering 5, 122–125 (1993)