Relational Algebra for Ranked Tables with Similarities: Properties and Implementation*

Radim Belohlavek[1,3], Stanislav Opichal[2], and Vilem Vychodil[3]

[1] Dept. Systems Science and Industrial Engineering
T. J. Watson School of Engineering and Applied Science
Binghamton University–SUNY, PO Box 6000, Binghamton, NY 13902–6000, USA
rbelohla@binghamton.edu
[2] PIKE ELECTRONIC, Ltd., Czech Republic
sopichal@pikeelectronic.com
[3] Dept. Computer Science, Palacky University, Olomouc
Tomkova 40, CZ-779 00 Olomouc, Czech Republic
vilem.vychodil@upol.cz

Abstract. The paper presents new developments in an extension of Codd's relational model of data. The extension consists in equipping domains of attribute values with a similarity relation and adding ranks to rows of a database table. This way, the concept of a table over domains (i.e., relation over a relation scheme) of the classical Codd's model extends to the concept of a ranked table over domains with similarities. When all similarities are ordinary identity relations and all ranks are set to 1, our extension becomes the ordinary Codd's model. The main contribution of our paper is twofold. First, we present an outline of a relational algebra for our extension. Second, we deal with implementation issues of our extension. In addition to that, we also comment on related approaches presented in the literature.

1 Introduction

1.1 Motivation and Outline of the Paper

Most of the current database systems are based on the well-known Codd's relational model of data: "A hundred years from now, I'm quite sure, database systems will still be based on Codd's relational foundation." [9, p. 1]. Main virtues of Codd's model are due to the reliance of the model 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." [9, p. 138].

Our paper is concerned with a particular extension of the relational model which is concerned with imprecision and uncertainty. Management of uncertainty and imprecision is one of the six currently most-important research directions

* Supported by by grant No. 1ET101370417 of GA AV ČR and by institutional support, research plan MSM 6198959214.

Table 1. Ranked data table over domains with similarities

$\mathcal{D}(t)$	*name*	*age*	*education*
1.0	Adams	30	Comput. Sci.
1.0	Black	30	Comput. Eng.
0.9	Chang	28	Accounting
0.8	Davis	27	Comput. Eng.
0.4	Enke	36	Electric. Eng.
0.3	Francis	39	Business

$$n_1 \approx_n n_2 = \begin{cases} 1 & \text{if } n_1 = n_2 \\ 0 & \text{if } n_1 \neq n_2 \end{cases}$$

$$a_1 \approx_a a_2 = s_a(|a_1 - a_2|) \quad \text{with scaling } s_a : \mathbb{Z}^+ \to [0,1]$$

$$\begin{array}{c|ccccc} \approx_e & \text{A} & \text{B} & \text{CE} & \text{CS} & \text{EE} \\ \hline \text{A} & 1 & .7 & & & \\ \text{B} & .7 & 1 & & & \\ \text{CE} & & & 1 & .9 & .7 \\ \text{CS} & & & .9 & 1 & .6 \\ \text{EE} & & & .7 & .6 & 1 \end{array}$$

proposed in the report from the Lowell debate by 25 senior database researchers [1]: "... current DBMS have no facilities for either approximate data or imprecise queries." Similarity, approximate matches, similarity-based queries, and related issues are the main motivations for our extension of the relational model. These issues are not new and have been approached in numerous papers in the past. The issues result in situations when one considers similarity of elements of domains rather than exact equality, i.e. when it is desirable to consider degrees of similarity rather than just "equal" and "not equal". For example, consider attribute age. The corresponding domain consists of positive integers. One might be interested in all persons in a given database with age equal to 30. Such a query is typical in the classical relational model (in terms of relational algebra: selection of all tuples with attribute age = 30). One might, however, be also interested in all persons in the database with age approximately 30. Intuitively, a person with age 30 satisfies this query completely (degree of satisfaction is 1.0), a person with age 29 satisfies this query rather well (degree of satisfaction is, say, 0.9), a person with age 25 satisfies this query but only to a small degree (say, 0.2), etc. The above degrees are, in fact, degrees of similarity, see [23], of the actual age to the reference age 30, i.e. 1.0 is a degree of similarity of 30 to 30, 0.9 is a degree of similarity of 29 to 30, 0.2 is a degree of similarity of 25 to 30. Of course, the degrees depend on how the similarity relation is defined. The above example, however simple, clearly demonstrates that taking similarity into account leads to qualitatively new features in querying and manipulation of data. Our attempt in previous papers as well as in this paper is to develop systematically an extension of the classical Codd's relational model which would play the same role in case when similarities are considered as the ordinary Codd's model plays in the classical case.

The main concept in our approach is that of a ranked data table (relation) over domains with similarities, see Tab. 1. This concept is our counterpart to the concept of a data table (relation) over domains of a classical relational model. A ranked data table over domains with similarities consists of three parts: data table (relation), domain similarities, and ranking. The data table (right top table in Tab. 1) coincides with a data table of a classical relational model. Domain similarities and ranking are what makes our model an extension of the classical model. The domain similarities (bottom part of Tab. 1) assign degrees of similarity to pairs of values of the respective domain. For instance, a degree of similarity of "Computer Science" and "Computer Engineering" is 0.9 while a degree of similarity of "Computer Science" and "Electrical Engineering" is 0.6. The ranking assigns to each row (tuple) of the data table a degree of a scale bounded by 0 and 1 (left top table in Tab. 1), e.g. 0.9 assigned to the tuple $\langle \text{Chang}, 28, \text{Accounting} \rangle$. The ranking allows us to view the ranked table as an answer to a similarity-based query (rank = degree to which a tuple matches a query). For instance, the ranked table of Tab. 1 can result as an answer to query "show all candidates with age about 30". In a data table representing stored data (i.e. prior to any querying), ranks of all tuples of the table are equal to 1. Therefore, the same way as tables in the classical relational model, ranked tables represent both stored data and outputs to queries. This is an important feature of our model.

We use fuzzy logic as our formal framework. We use a formal system of first-order fuzzy logic the same way as the system of first-order classical logic is used in the classical relational model. This way, our model keeps the user-friendly symbolical character of the classical model and adds a quantitative layer which takes care of the management of uncertainty. This is an important distinction from other "fuzzy approaches" to the relational model which, from our point of view, are often *ad-hoc*.

In our previous papers, we developed selected issues within the framework of our extension, e.g., functional dependencies, computation of non-redundant sets of functional dependencies, Armstrong-like axiomatization, and related issues are studied in [2,3,4,5].

In the present paper, we focus on relational algebra for our extension of the relational model. In particular, we present an overview of operations of the relational algebra, present illustrative examples and selected results on properties of the relational algebra. In addition to that, we focus on the problem of implementation of our extension. Section 1.2 briefly reviews related approaches. Section 1.3 summarizes preliminaries from fuzzy logic. In Section 2.1 we introduce our model. Section 2.2 presents relational algebra and related results. Section 2.3 deals with implementation of our relational model. Section 3 outlines future research.

1.2 Related Approaches

The first paper on a "fuzzy approach" to the relational model is [7]; [6] provides an overview with many references. We found over 100 contributions related to "fuzzy approach" to the relational model. A main feature of almost all of the approaches is that they are *ad-hoc* in that an analogy of a clear relationship between a relational model and first-order fuzzy logic is missing in the approaches which leads to the impression of arbitrariness of the approaches. This is partly because fully fledged logical calculi have not been developed until quite recently, see e.g. [12,13]. On the other hand, several ideas including some of those presented in our paper were already discussed in the literature. For instance, the idea of considering domains with similarity relations goes back to [7]. The idea of assigning ranks to tuples appeared in [21] although with not quite a clear meaning of ranks ("fuzzy measure of association among a set of domain values" [21]).

1.3 Preliminaries

We use fuzzy logic to represent and manipulate truth degrees of propositions like "u is similar to v". Moreover, we need to process (aggregate) the degrees. For instance, consider a query "show all candidates which are about 30 years old and a degree in specialization similar to Computer Science". According to Tab. 1, Davis satisfies subqueries concerning age and education to degrees 0.8 and 0.9, respectively. Then, we combine the degrees using a fuzzy conjunction connective $\otimes$ to get a degree $0.8 \otimes 0.9$ to which Davis satisfies the conjunctive query.

When using fuzzy logic, we have to pick an appropriate scale L of truth degrees (which serve e.g. as grades for evaluating similarity of two objects) 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, instead of taking one particular fuzzy conjunction $\otimes$ and fuzzy implication $\rightarrow$, we take any $\otimes$ and $\rightarrow$ which satisfy certain conditions. This way, we obtain a structure $\mathbf{L} = \langle L, \leq, \otimes, \rightarrow, \dots \rangle$ of truth degrees with logical connectives. Although more general than one particular choice of a scale and connectives, such an approach is easier to handle theoretically and supports the symbolical character of our model. 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 [12,13] 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). Note that a special case of a complete residuated lattice is a two-element Boolean algebra of classical (bivalent) logic.

Having $\mathbf{L}$, we define usual notions [12,13,15]: an $\mathbf{L}$-set (fuzzy set) A in universe U is a mapping $A \colon U \rightarrow L$, $A(u)$ being interpreted as "the degree to which u belongs to A". The operations with $\mathbf{L}$-sets are defined componentwise. Binary $\mathbf{L}$-relations (binary fuzzy relations) between X and Y can be thought of as $\mathbf{L}$-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 $E(u,v) = E(v,u)$. A reflexive and symmetric fuzzy relation is called a similarity. We often denote a similarity by $\approx$ and use an infix notation, i.e. we write $(u \approx v)$ instead of $\approx(u,v)$.

2 Relational Algebra and Implementation of Relational Model over Domains with Similarities

2.1 Ranked Tables over Domains with Similarities

We use Y for a set of attributes (attribute names) and denote the attributes by $y, y_1, \ldots$; $\mathbf{L}$ denotes a fixed structure of truth degrees and connectives.

Definition 1. A *ranked data table over domains with similarity relations* (with Y and $\mathbf{L}$) 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 $t \in \times_{y \in Y} D_y$, there is a degree $\mathcal{D}(t) \in L$ (called rank of t in $\mathcal{D}$) assigned to t.

Remark 1. (1) $\mathcal{D}$ 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. In the classical relational model, similarity relations are not present.

(2) $t[y]$ denotes a value from D_y of tuple t on attribute y. We require that there is only a finite number of tuples with non-zero degree. 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 classical model.

(3) Formally, $\mathcal{D}$ is a fuzzy relation between domains D_y ($y \in Y$). Rank $\mathcal{D}(t)$ is interpreted as a degree to which the tuple t satisfies requirements posed by a query. A table $\mathcal{D}$ representing just stored data, i.e. data prior to querying, has all the ranks equal to 0 or to 1, i.e. $\mathcal{D}(t) = 0$ or $\mathcal{D}(t) = 1$ for each tuple t. Here again, $\mathcal{D}$ can be thought of as a result of a query, namely, a query "show all stored data".

2.2 Relational Algebra

In the classical model, relational algebra is based on the calculus of classical relations. In the same spirit, since ranked tables are in fact fuzzy relations, our relational algebra is based on the calculus of fuzzy relations [12,15]. Due to the limited scope, we present only selected parts of our algebra and leave details and further parts to a full version of the paper.

Table 2. Ranked tables over domains with similarities

$\mathcal{D}_1(t)$	*name*	*age*	*education*
1.0	Black	30	Comput. Eng.
0.9	Chang	28	Accounting
0.1	Francis	39	Business

$\mathcal{D}_2(t)$	*name*	*age*	*education*
1.0	Adams	30	Comput. Sci.
0.8	Davis	27	Comput. Eng.
0.5	Black	30	Comput. Eng.
0.4	Enke	36	Electric. Eng.
0.3	Francis	39	Business

Operations of our relational algebra can be basically classified as follows.

Counterparts to Boolean operations of classical model. These are operations like union, intersection, etc. For instance, a union $\mathcal{D}_1 \cup \mathcal{D}_2$ of two ranked tables with similarities, $\mathcal{D}_1$ and $\mathcal{D}_2$, is defined by

$$[\mathcal{D}_1 \cup \mathcal{D}_2](t) \;=\; \mathcal{D}_1(t) \vee \mathcal{D}_1(t).$$

This says that a rank of t in $\mathcal{D}_1 \cup \mathcal{D}_2$ is given by taking a rank of t in $\mathcal{D}_1$ and a rank of t in $\mathcal{D}_2$, and applying $\vee$ to these ranks. 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 $\mathcal{D}_2(t)$. For example, the ranked table $\mathcal{D}$ from Tab. 1 is a result of union of ranked tables $\mathcal{D}_1$ and $\mathcal{D}_2$ depicted in Tab. 2, i.e. $\mathcal{D} = \mathcal{D}_1 \cup \mathcal{D}_2$.

More generally, for any binary (and similar for other arities) operation $\odot$ with fuzzy relations, we define a corresponding operation (denoted again) $\odot$ which yields for any two ranked tables $\mathcal{D}_1$ and $\mathcal{D}_2$ (with common Y, domains, and similarities) a ranked table $\mathcal{D}$ assigning to any tuple t a rank $\mathcal{D}(t)$ defined componentwise by

$$\mathcal{D}(t) \;=\; \mathcal{D}_1(t) \odot \mathcal{D}_2(t).$$

New operations based on calculus of fuzzy relations. The calculus of fuzzy relations contains operations which either have no counterparts with classical relations or the counterparts are trivial. An interesting example is a so-called a-cut of a fuzzy relation. For a ranked table $\mathcal{D}$ and a rank $a \in L$, an a-cut of $\mathcal{D}$ is a ranked table ${}^a\mathcal{D}$ defined by

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

That is, ${}^a\mathcal{D}$ is a non-ranked table which contains those tuples of $\mathcal{D}$ with ranks greater or equal to a. This is quite a natural operation for manipulation of ranked tables which allows the user to select only a part of a query result given by threshold a.

Note that in combination with intersection, we can use a-cut to get the part of $\mathcal{D}$ with ranks at least a, i.e. we can get $\mathrm{Above}_a(\mathcal{D})$ defined by

$$[\mathrm{Above}_a(\mathcal{D})](t) = \begin{cases} \mathcal{D}(t) & \text{for } \mathcal{D}(t) \geq a, \\ 0 & \text{otherwise.} \end{cases}$$

Namely, we have

Table 3. Results of $\text{Above}_{0.7}$ and $\sigma_{e=\text{CE}}$ applied to $\mathcal{D}$ from Tab. 1

$[\text{Above}_{0.7}(\mathcal{D})](t)$	<u>n</u>ame	<u>a</u>ge	<u>e</u>ducation
1.0	Adams	30	CS
1.0	Black	30	CE
0.9	Chang	28	AC
0.8	Davis	27	CE

$\mathcal{D}(t)$	<u>n</u>ame	<u>a</u>ge	<u>e</u>ducation
1.0	Black	30	CE
0.9	Adams	30	CS
0.8	Davis	27	CE
0.1	Enke	36	EE

Lemma 1. *For $\mathcal{D}$, $a \in L$, we have* $\text{Above}_a(\mathcal{D}) = \mathcal{D} \cap {}^a\mathcal{D}$.

For instance, $\text{Above}_{0.7}(\mathcal{D})$ for $\mathcal{D}$ from Tab. 1 is depicted in Tab. 3 (left).

Counterparts to selection, join, projection, etc. These operations stem basically from the classical ones by taking into account similarity relations (or, in general fuzzy relations θ in place of classical comparators). For illustration, we consider only a similarity-based selection and a similarity-based join.

The basic form of selection works as follows. Given a value $u \in D_y$, a select operator with inputs $y = u$ yields a ranked table $\sigma_{y=u}(\mathcal{D})$ for which a rank of a tuple $t \in \times_{y \in Y} D_y$ is given by

$$\mathcal{D}(t) \otimes (t[y] \approx_y u),$$

i.e.

$$[\sigma_{y=u}(\mathcal{D})](t) \;=\; \mathcal{D}(t) \otimes (t[y] \approx_y u),$$

$[\sigma_{y=u}(\mathcal{D})](t)$ can be read as follows: One takes a degree $\mathcal{D}(t)$ (rank of t in $\mathcal{D}$) and a degree $t[y] \approx_y u$ (degree of similarity between $t[y]$ and u) and applies a "fuzzy conjunction" $\otimes$ to $\mathcal{D}(t)$ and $t[y] \approx_y u$. That is, a rank of t in $\sigma_{y=u}(\mathcal{D})$ can be seen as a truth degree of proposition "t is in $\mathcal{D}$ *and* the value of t in attribute y is similar to u". Tab. 3 (right) shows $\sigma_{e=\text{CE}}(\mathcal{D})$ (e denotes "education", CE denotes "Computer Engineering") for $\mathcal{D}$ from Tab. 1 for $\otimes$ being Łukasiewicz conjunction.

As in the ordinary case, selection can be extended to several input values. Given $\mathcal{D}$, a select operator with inputs given by $y_1 = u_1, \ldots, y_k = u_k$ yields a ranked table $\sigma_{y_1=u_1,\ldots,y_k=u_k}(\mathcal{D})$ for which a rank of a tuple t is given by

$$\mathcal{D}(t) \otimes (x[y_1] \approx_{y_1} u_1) \otimes \cdots \otimes (x[y_k] \approx_{y_k} u_k),$$

i.e. a degree of "t is in $\mathcal{D}$ and value of t in y_1 is similar to u_1 and $\cdots$ and value of t in y_k is similar to u_k".

To illustrate similarity-based join, consider a ranked table $\mathcal{D}_1$ from Tab. 1 which can be thought of as a result to a query "select candidates with age about 30") and a ranked table $\mathcal{D}_2$ from Tab. 4 (left) describing open positions with required education. A similarity-based join $\mathcal{D}_1 \bowtie \mathcal{D}_2$ then describes possible job assignments. A rank $[\mathcal{D}_1 \bowtie \mathcal{D}_2](n, a, e, p)$ of tuple $\langle n, a, e, p\rangle$ in $\mathcal{D}_1 \bowtie \mathcal{D}_2$ is given by

$$\textstyle\bigvee_{e_1,e_2}(\mathcal{D}_1(n, a, e_1) \otimes (e_1 \approx_e e) \otimes (e \approx_e e_2) \otimes \mathcal{D}_2(p, e_2))$$

Table 4. Illustration of similarity-based join

$\mathcal{D}(t)$	<u>p</u>osition	<u>e</u>ducation
1.0	programmer	Comput. Sci.
1.0	syst. technician	Comput. Eng.

$\mathcal{D}(t)$	<u>n</u>ame	<u>p</u>osition
1.0	Adams	programmer
1.0	Black	syst. technician
0.9	Adams	syst. technician
0.9	Black	programmer

where e_1, e_2 range over the domain corresponding to *education*. That is, the join runs not only over equal values but also over similar values. $[\mathcal{D}_1 \bowtie \mathcal{D}_2](n, a, e, p)$ can be seen as a truth degree of "candidate with name n, age a, and education e_1 belongs to table $\mathcal{D}_1$ and e_1 is similar to e_2 and e_2 is a required education for position p". The table of Tab. 4 (right) shows a result of $\text{Above}_{0.9}(\mathcal{D}_1 \bowtie \mathcal{D}_2)$, cf. above, projected to *name* and *position*.

Further operations. Among others, here belong some interesting operations studied in databases and information retrieval. As an example, consider top_k which gained a considerable interest recently, see [10,11] and also [14]. We define $\text{top}_k(\mathcal{D})$ to contain the first k tuples (according to rank ordering) of $\mathcal{D}$ with their ranks (if there are less than k ranks in $\mathcal{D}$ then $\text{top}_k(\mathcal{D}) = \mathcal{D}$; and $\text{top}_k(\mathcal{D}))$ includes also the tuples with rank equal to the rank of the k-th tuple). Note that top_k is a part of a query language described in [19]. As presented in [4], top_k is indeed a relational operator, i.e. there is a corresponding formula in the corresponding relational calculus which is based on first-order fuzzy logic.

Remark 2. In [4], we presented a tuple and domain relational calculi for our relational algebra with the completeness theorem.

We obtained several results on properties of the operations of our relational algebra which are analogous to properties of classical relational algebra. Due to the limited scope, we present just the following properties of selection:

Lemma 2. *For $\mathcal{D}$ and $u_i \in D_{y_i}$ we have*

$$\sigma_{y_1=u_1,\dots,y_k=u_k}(\mathcal{D}) = \sigma_{y_1=u_1}(\cdots(\sigma_{y_k=u_k}(\mathcal{D}))\cdots),$$
$$\sigma_{y_1=u_1}(\sigma_{y_2=u_2}(\mathcal{D})) = \sigma_{y_2=u_2}(\sigma_{y_1=u_1}(\mathcal{D})).$$

2.3 Implementation Issues

This section presents, by means of examples, considerations on and proposal of implementation of the extended relational model. Our basic aim to make use of existing relational database management systems (RDBMS) and to develop a prototype implementation of the extension by ranks and similarities. We used a RDBMS Oracle9i, query language SQL, and its procedural extension PL/SQL. Oracle9i enables us to use stored procedures and functions. This feature enables us to store with a database scheme some functions that can be used in SQL queries and other SQL statements.

We did use stored functions to implement similarity relations on particular domains. Recall that a similarity relation on a domain D_y is, in fact, a function of the Cartesian product $D_y \times D_y$ into a scale L of truth degrees, e.g. into $[0, 1]$. For example, a function `sim_age` implementing similarity $\approx_{age}$ on the domain of ages (see Table 1) can be defined by

```
CREATE FUNCTION sim_age(age1 NUMBER, age2 NUMBER)
  RETURN NUMBER IS
x NUMBER;
BEGIN
  x := 1 - abs(age1-age2)/SCON;
  IF x < 0 THEN
     x := 0;
  END IF;
  RETURN NUMBER(x);
END sim_age;
```

where `SCON` is an appropriate scaling constant. Similarities on non-numerical domains D_y can be implemented as relations over relation scheme given by three attributes: y (first), y (second), and an attribute representing truth degrees (third). For instance, if we consider the ranked data table from Tab. 1, the information about the similarity on the domain of education can be stored in a data base table which is created by the following SQL command:

```
CREATE TABLE sim_education_tab (
  val1 VARCHAR (30) NOT NULL,
  val2 VARCHAR (30) NOT NULL,
  degree NUMBER NOT NULL,
  PRIMARY KEY (val1, val2),
  CHECK (val1 < val2)
);
```

Notice that `val1` and `val2` are string attributes representing education descriptions and `degree` is their similarity degree. Since similarity relations are reflexive, there is no need to store information about similarities when `val1` and `val2` agree on their values. Furthermore, similarity relations are always symmetric. Hence, if `sim_education_tab` contains information about the similarity degree for `val1` $= e_1$ and `val2` $= e_2$, there is no need to store the information for `val1` $= e_2$ and `val2` $= e_1$. Since the domain of education descriptions, which are encoded by strings of literals, can be lexically ordered, we can store in `sim_education_tab` only records representing similarities of education values `val1` $= e_1$ and `val2` $= e_2$ such that e_1 is lexically smaller than e_2 and $e_1 \approx_e e_2 > 0$. The latter SQL command creates `sim_education_tab` with explicit constraint saying that the value of `val1` should be lexically smaller than the value of `val2` which reflects the organization of the data table just mentioned. The table `sim_education_tab` has a primary key $\{$`val1`, `val2`$\}$. In most RDBMS the definition of such a primary key is accompanied by creation of a unique multi-column index which can significantly improve the efficiency of querying `sim_education_tab` regarding the similarity degree. Following the example from Tab. 1, `sim_education_tab` can be filled as follows:

```
INSERT INTO sim_education_tab VALUES ('A', 'B', 0.7);
INSERT INTO sim_education_tab VALUES ('CE', 'CS', 0.9);
INSERT INTO sim_education_tab VALUES ('CE', 'EE', 0.7);
INSERT INTO sim_education_tab VALUES ('CS', 'EE', 0.6);
```

In order to achieve flexibility, we should create a stored function `sim_education` (its source code is not shown here) which given two education descriptions returns their similarity degree, querying the `sim_education_tab` table. Using functions implementing similarities on domains and using standard features of SQL, one can implement queries corresponding to the operations of the relational algebra for ranked tables over domains with similarities.

The similarity-based queries are intended to query ranked tables, i.e. the arguments of similarity-based queries are, in principle, ranked tables. In a conventional RDBMS, a ranked table over relation scheme Y can be represented as a relation over relation scheme Y to which we add an attribute for ranks (first column). However, from the users' point of view, it is convenient to have the option to apply similarity-based queries to ordinary relations (i.e., tables without ranks) as well. Suppose that we have in our RDBMS a relation (called `candidates`) depicted in Tab. 1 (without ranks) and want to see the candidates with age similar to 30. This can be accomplished by SQL statement

```
SELECT sim_age(age, 30) AS sim_age_30, *
  FROM candidates
  ORDER BY sim_age_30 DESC;
```

The result is a relation with the first column representing ranks. In terms of our relational algebra, this ranked table is a result of $\sigma_{age=30}(\mathcal{D})$ where $\mathcal{D}$ is the table from Tab. 1. In a similar way, using a function implementing fuzzy logical conjunction, one can form SQL statements implementing queries like "select candidates with age around 30 and education similar to electrical engineering". For instance, the standard Łukasiewicz conjunction can be represented by the following stored function:

```
CREATE FUNCTION luk_conj(degree1 NUMBER, degree2 NUMBER)
  RETURN NUMBER IS
x NUMBER;
BEGIN
  x := degree1 + degree2 - 1;
  IF x < 0 THEN
     x := 0;
  END IF;
  RETURN NUMBER(x);
END luk_conj;
```

Using `luk_conj` together with `sim_age` and `sim_education`, we can formulate the above-mentioned similarity-based query as follows:

```
SELECT luk_conj(sim_age(age, 30),
                sim_education(education, 'EE')) AS sim, *
  FROM candidates
  ORDER BY sim DESC;
```

Remark 3. Note that we have not discussed here a user-friendly syntax of an extension of standard SQL which would enable the user to express similarity-based queries in a comfortable form. Such an extension can be accomplished, e.g., by devising a preprocessor which would translate statements of the SQL extension to statements of ordinary SQL. Due to the limited scope, we do not deal with this issue in the present paper. Our intention was to demonstrate that the similarity-based queries can be implemented by means of the stored functions for similarities and by means of the standard SQL.

3 Future Research

Main topics for future research are:

– development of relational algebra with focus on new (non-standard) operations and features;
– development of prototype implementation of the extended Codd's model by means of existing relational data base management systems;
– design of an extension of standard SQL for the extended Codd's model and its implementation (some proposals for "fuzzy SQL" can be found in the literature but our relational model of data is different from the respective models in the literature);
– development of standard issues from relational databases in our extended setting (e.g., data dependencies, redundancy, normalization, and design of databases, optimization issues, etc.).

References

1. Abiteboul, S., et al.: The Lowell database research self-assessment. Comm. ACM 48(5), 111–118 (2005)
2. Belohlavek, R., Vychodil, V.: Functional dependencies of data tables over domains with similarity relations. In: IICAI 2005, Pune, India, December 2005, pp. 2486–2504 (2005)
3. Belohlavek, R., Vychodil, V.: Data tables with similarity relations: functional dependencies, complete rules and non-redundant bases. In: Lee, M.L., Tan, K.-L., Wuwongse, V. (eds.) DASFAA 2006. LNCS, vol. 3882, pp. 644–658. Springer, Heidelberg (2006)
4. Belohlavek, R., Vychodil, V.: Relational model of data over domains with similarities: an extension for similarity queries and knowledge extraction. In: IEEE IRI 2006, The 2006 IEEE Int. Conf. Information Reuse and Integration, Waikoloa Village, Hawaii, USA, September 16–18, 2006, pp. 207–213 (2006)

5. Belohlavek, R., Vychodil, V.: Codd's relational model of data and fuzzy logic: comparisons, observations, and some new results. In: Proc. CIMCA 2006, Sydney, Australia, p. 6 (2006) ISBN 0–7695–2731–0
6. Bosc, P., Kraft, D., Petry, F.: Fuzzy sets in database and information systems: status and opportunities. Fuzzy Sets and Syst. 156, 418–426 (2005)
7. Buckles, B.P., Petry, F.: A fuzzy representation of data for relational databases. Fuzzy Sets Syst. 7, 213–226 (1982)
8. Buckles, B.P., Petry, F.E.: Fuzzy databases in the new era. In: ACM SAC 1995, pp. 497–502, Nashville, TN (1995)
9. Date, C.J.: Database Relational Model: A Retrospective Review and Analysis. Addison-Wesley, Reading (2000)
10. Fagin, R.: Combining fuzzy information from multiple systems. J. Comput. System Sci. 58, 83–99 (1999)
11. Fagin, R.: Combining fuzzy information: an overview. ACM SIGMOD Record 31(2), 109–118 (2002)
12. Gottwald., S.: A Treatise on Many-Valued Logics. Research Studies Press, Beldock, England (2001)
13. Hájek, P.: Metamathematics of Fuzzy Logic. Kluwer, Dordrecht (1998)
14. Illyas, I.F., Aref, W.G., Elmagarmid, A.K.: Supporting top-k join queries in relational databases. The VLDB Journal 13, 207–221 (2004)
15. Klir, G.J., Yuan, B.: Fuzzy Sets and Fuzzy Logic. Theory and Applications. Prentice-Hall, Englewood Cliffs (1995)
16. Li, C., Chang, K.C.-C., Ilyas, I.F., Song, S.: RanSQL: Query Algebra and Optimization for Relational top-k queries. In: ACM SIGMOD 2005, pp. 131–142 (2005)
17. Maier, D.: The Theory of Relational Databases. Computer Science Press, Rockville (1983)
18. Medina, J.M., Vila, M.A., Cubero, J.C., Pons, O.: Towards the implementation of a generalized fuzzy relational database model. Fuzzy Sets and Systems 75, 273–289 (1995)
19. Penzo, W.: Rewriting rules to permeate complex similarity and fuzzy queries within a relational dastabase system. IEEE Trans. Knowledge and Data Eng. 17, 255–270 (2005)
20. 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)
21. 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)
22. Takahashi, Y.: Fuzzy database query languages and their relational completeness theorem. IEEE Trans. Knowledge and Data Engineering 5, 122–125 (1993)
23. Zadeh, L.A.: Similarity relations and fuzzy orderings. Information Sciences 3, 177–200 (1971)