Relational Model of Data over Domains with Similarities: 
An Extension for Similarity Queries and Knowledge Extraction

Radim Belohlavek and Vilem Vychodil 
Dept. Comp. Science, Palacky University, Tomkova 40, CZ-779 00, Olomouc, Czech Republic 
e-mail: {radim.belohlavek, vilem.vychodil}@upol.cz

Abstract— We present an extension of Codd’s relational model of data. Our extension is motivated by similarity-based querying. It consists in equipping each domain of attribute values with a similarity relation and in modifying the classical relational model in order to account for issues generated by adding similarities. As a counterpart to data tables over a set of domains of Codd’s model, we introduce ranked data tables over domains with similarities. We present a relational algebra, and as an application, we introduce ranked data tables in modifying the classical relational model in order to account for issues raised by similarity-based information retrieval. In addition to completeness result w.r.t. Armstrong-like rules, describe non-redundant attributes
bases and provide an algorithm for computing the bases. In addition to its basis in mathematics and predicate logic.” [11, p. 138]). (“The relational approach really is rock solid, owing (once again) 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.” [11, p. 138]).

Since the inception of the relational model, there have been proposed various extensions of the model. Relevant to this paper are the extensions aiming at the capability of the relational model to deal with various forms of uncertainty. Management of uncertainty is one of the six currently most-important research directions proposed in the report from the Lowell debate by 25 senior database researchers [1]. In particular, it was pointed out in [1] that “...current DBMS have no facilities for either approximate data or imprecise queries.”

Similarity, approximate matches, and the corresponding domains are the main motivations for our extension of the relational model. In particular, our primary concern is with the situation when domains are equipped with similarity relations, i.e. when it is desirable to consider degrees of similarity rather than only “equal” and “not equal”. Such a concern comes primarily from the standpoint of information retrieval when considering similarity-based queries like “show all candidates with age about 30”. Therefore, our attempt can be seen as extending the relational model by taking into account issues raised by similarity-based information retrieval. In addition to similarity-based queries, domains with similarity generate topics related to data mining. An example dealt with in more detail in our paper are functional dependencies which, in their extended version, represent new type of data dependencies like “similarity in values of attributes A implies similarity in values of attributes B”.

The main concept we deal with is that of a ranked data table (relation) over domains with similarities. This concept is our counterpart to the concept of a data table (relation) over domains of a classical relational model. The concept is illustrated in Tab. I. It 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.I) 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 (Chang,30,Accounting). 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.I 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. In particular, 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. Using a formal system of first-order fuzzy logic enables us to utilize both the symbolical and the numerical character of fuzzy logic. That is, we work with first-order formulas which can be read in natural language and have thus a clear meaning (symbolical character). According to the rules of fuzzy logic, the formulas get assigned degrees, e.g. from [0,1], which are being processed according to the rules of fuzzy logic (numerical character). 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

<table>
<thead>
<tr>
<th>Table I</th>
<th>Ranked data table over domains with similarities</th>
</tr>
</thead>
<tbody>
<tr>
<td>D(t)</td>
<td>Age</td>
</tr>
<tr>
<td>0.9</td>
<td>Chang 28 Accounting</td>
</tr>
<tr>
<td>0.8</td>
<td>Davis 27 Comput. Eng.</td>
</tr>
<tr>
<td>0.4</td>
<td>Enke 36 Electric. Eng.</td>
</tr>
<tr>
<td>0.3</td>
<td>Francis 39 Business</td>
</tr>
</tbody>
</table>

A1 ≈a A2 = \{ 1 if n₁ = n₂, 0 if n₁ ≠ n₂ \}

A1 ≈a A2 = sₐ(∥a₁ - a₂∥)

with scaling sₐ: Z₊ → [0, 1]

\[ sₐ(n₁, n₂) = \begin{cases} 1 & \text{if } n₁ = n₂ \\ 0 & \text{if } n₁ \neq n₂ \end{cases} \]

\[ A₁ \approx_a A₂ = sₐ(∥A₁ - A₂∥) \]

\[ sₐ: Z₊ \rightarrow [0, 1] \]
approaches” to the relational model which, from our point of view, are often ad-hoc.

The paper is organized as follows. In Section I-B we review related approaches. Section I-C contains preliminaries from fuzzy logic. In Section II-A we introduce our model and basic related notions. Section II-B outlines relational algebra, tuple relational calculus, and domain relational calculus for our model and outlines their equivalence. Here, instead of going to technical details, we aim to emphasize the fact that the manipulative part of our model contains interesting operations which cannot be expressed in the classical model. We focus on non-classical issues like similarity-based selection, join, etc. operations which cannot be expressed in the classical model. We the fact that the manipulative part of our model contains interesting
clause. Here, instead of going to technical details, we aim to emphasize
relevance. Instead, we go directly to our paper we need to
approaches. Section I-C contains preliminaries from fuzzy logic.

[7]. The idea of assigning ranks to tuples appeared in [28] although
the idea of considering domains with similarity relations goes back to
in their paper were already discussed in the literature. For instance,
did in fact. Most of the contributions were presented inside the fuzzy
use of fuzzy logic in information systems did not spread wider than it
membership degrees. In our opinion, this is the main reason why the
at some important features of the model; several concepts of the
ad-hoc

B. Related approaches

Extensions of the classical relational model of data attempting to
capture uncertainty and indeterminacy can be classified in a number of
ways. One of them is according to the type of uncertainty. Before
going to the approaches directly related to our paper we need to
clearly distinguish our approach from probabilistic ones to prevent a
confusion. The probabilistic models, see e.g. [12], [15], basically aim at modeling of probabilistic uncertainty of data which is very different from what we deal with, see e.g. [21]. Our model is deterministic; the only point where we depart from the classical model is that we use graded (fuzzy) predicates.

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. An ad-hoc choice of fuzzy logic connectives, the lack of symbolic level in the formalism, plus not clearly justifying why this and that is fuzzy rather than bivalent in a model has some important consequences. First of all are the impression of arbitrariness of the model; difficulties to handle the model theoretically and to arrive at some important features of the model; several concepts of the models are difficult to read and thus user-unfriendly; the impression of unjustified theorizing due to not explaining the meaning of fuzzy membership degrees. In our opinion, this is the main reason why the use of fuzzy logic in information systems did not spread wider that it did. In fact, most of the contributions were presented inside the fuzzy community. The primary reason of the above-described shortcomings is that an analogy of a clear relationship between a relational model and first-order fuzzy logic is missing in the approaches. This is partly because fully fledged logical calculi have not been developed until quite recently, see e.g. [16], [17].

Another feature of the approaches is that a consideration of computational tractability of the proposed concepts seems to be an exception rather than a rule.

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 [28] although with not quite a clear meaning of ranks (values of a “possibility
distribution function”, “fuzzy measure of association among a set of
domain values” [28]). Quite several approaches exist in the literature on “fuzzy functional dependencies” and we comment on them in
Section II-C. Another idea, which we do not consider here is the possibility for the tuples to contain also fuzzy sets of (or possibility distributions on) attribute values in addition to the attribute values themselves, [27] is perhaps the first paper developing this issue.

We comment on the relationships of our model to previous ap-
proaches later on; a detailed description will be presented elsewhere.

C. Preliminaries

We use fuzzy logic to represent and manipulate truth degrees of propositions like “a 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. I, Davis satisfies subqueries concerning age and education in degrees 0.8 and 0.9, respectively. Then, we combine the degrees using a fuzzy conjunction connective ⊗ to get a degree 0.8 ⊗ 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.). Most of the existing fuzzy approaches to databases use the real interval $[0,1]$ and one particular couple of connectives on $[0,1]$. Our approach is different in that we do not say which particular scale and connectives we take. Rather, we postulate the required properties of the scale and of the connectives. Thus, we take an arbitrary partially-ordered scale $(L,\leq)$ 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 ⊗ and fuzzy implication −→, we take any ⊗ and −→ which satisfy certain conditions. For instance, our fuzzy conjunctions are order-preserving functions on $L$ satisfying some further requirements. This way, we obtain a structure $L = (L,\leq,\otimes,\rightarrow,\cdots)$ 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 symbolic character of our model.

In what follows, we present technical details of the preliminaries; for further information, the reader is referred to [16], [17], [20].

For structures $L$ of truth degrees, we use so-called complete residuated lattices, i.e. structures $L = (L,\wedge,\vee,\otimes,\rightarrow,0,1)$ such that $(L,\wedge,\vee,0,1)$ is a complete lattice with 0 and 1 being the least and greatest element of $L$, respectively; $(L,\otimes,1)$ is a commutative monoid (i.e. $\otimes$ is commutative, associative, and $a\otimes 1 = 1\otimes a = a$ for each $a \in L$); $\otimes$ and $\rightarrow$ satisfy so-called adjointness property, i.e. $a \otimes b \leq c$ iff $a \leq b \rightarrow c$, for each $a, b, c \in L$. A truth-stressing hedge (shortly, a hedge) [17] on $L$ is a unary operation $^{*}:L \rightarrow L$ satisfying (i) $1^{*} = 1$, (ii) $a^{*} \leq a$, (iii) $(a \rightarrow b)^{*} \leq a^{*} \rightarrow b^{*}$, (iv) $(a\otimes b)^{*} = a^{*} \otimes b^{*}$, for all $a, b \in L$. Elements of $L$ are called truth degrees. Hedge $^{*}$ is a truth function of logical connective “very true” and properties (i)–(iv) have natural interpretations, see [17].

A favorite choice of $L$ 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). Two boundary cases of hedges are (i) identity, i.e. $a^{*} = a$ (for all $a \in L$); (ii) globalization: $1^{*} = 1$, and $a^{*} = 0$ ($a \neq 1$). Note that a special case of a complete residuated lattice with a hedge is a two-element Boolean algebra of classical (bivalent) logic.

Having $L$, we define usual notions [16], [17], [20]: an $L$-set (fuzzy set) $A$ in universe $U$ is a mapping $A: U \rightarrow L$, $A(a)$ being
interpreted as “the degree to which a belongs to A”. If U is finite, we write \( A = \{ \ldots, a_n, \ldots \} \) to denote that \( A(u) = a \neq 0 \). Let \( U \) denote the collection of all \( L \)-sets in \( U \). 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 \( E(u, v) = E(v, u) \). 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 \( \approx (u, v) \). For fuzzy sets \( A, B \in L^U \), a degree \( S(A, B) \) to which \( A \) is a subset of \( B \) is defined by \( S(A, B) = \int_{u \in U} \mu_A(u) \rightarrow \mu_B(u) \).

II. RELATIONAL MODEL OVER DOMAINS WITH SIMILARITIES

A. Basic concepts

In this section, we describe the basic concepts of our extended relational model. If not defined otherwise, we use the notions related to the relational model as defined in [23]. In our description, we concentrate on the issues related to domain similarities and table ranks. We use \( Y \) for a set of attributes (attribute names) and denote the attributes 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 \( \approx_y : D_y \times D_y \rightarrow L \) which is reflexive (i.e. \( u \approx_y u \)) 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 2: (1) \( \mathcal{D} \) can be seen as a table with rows and columns corresponding to tuples and attributes, like in Tab. I. By \( t[y] \) we denote a value from \( D_y \) of tuple \( t \) on attribute \( y \). We require that there is only a finite number of tuples which get assigned a non-zero degree (i.e. the corresponding table is finite). Clearly, if \( L = \{ 0, 1 \} \) and if each \( \approx_y \) is equality, the concept of a ranked data table with similarities coincides with that of a data table (relation) of a classical model.

(2) Formally, \( \mathcal{D} \) is a fuzzy relation between domains \( D_y (y \in Y) \).

As mentioned above, \( \mathcal{D}(t) \) is interpreted as a degree to which the tuple \( t \) satisfies requirements posed by a query. We use “non-ranked table” if for each tuple \( t \), \( \mathcal{D}(t) = 0 \) or \( \mathcal{D}(t) = 1 \). This accounts for tables representing stored data (prior to querying).

(3) Sometimes, we add additional requirements for \( \approx_y \), e.g. transitivity w.r.t. a binary operation \( \ominus \) on \( L \) or separability (\( u \approx_v v = 1 \)) if \( u = v \). We are not concerned here with how the similarities are represented (we assume they can either be computed or, if \( D_y \) is small, are stored).

(4) Ranked tables over domains with similarities appear in [28]. However, the authors consider only \( \{0,1\} \) as a scale and no logical connectives.

B. Relational algebra and calculus

1) Relational algebra: Relational algebra of the classical model 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 [16], [20].

Due to the limited scope, we present in detail only selected parts of our algebra and leave the rest in an outline. Details will be presented in a full version of the paper.

| \( \mathcal{D}(t) \) | \begin{tabular}{c|c} \textbf{position} & \textbf{education} \\
\hline
1.0 & programmer Comput. Sci. \\
\end{tabular} |
| --- | --- |
| \( \mathcal{D}(t) \) | \begin{tabular}{c|c} \textbf{name} & \textbf{position} \\
\hline
1.0 & Adams programmer \\
1.0 & Black syst. technician \\
1.0 & Adams syst. technician \\
1.0 & Black programmer
\end{tabular} |

Our relational algebra is relative to \( L \) and manipulates ranked data tables with common \( Y \), domains, and similarities. Operations of our relational algebra can be classified as follows.

Counterparts to Boolean operations of classical model Here, 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 \( \mathcal{D}_1 \odot \mathcal{D}_2 \) assigning to any tuple \( t \) a rank \( \mathcal{D}(t) \) defined componentwise by

\[
\mathcal{D}(t) = \mathcal{D}_1(t) \odot \mathcal{D}_2(t).
\]

This accounts for operations based on \( \land, \lor, \neg, \rightarrow \), etc. (this way, we obtain our counterparts to intersection, union, etc.). Note that, one has to be careful when reducing operations to other operations. For instance, unlike classical case, De Morgan law is not available in fuzzy logic in general and, as a consequence, union cannot be expressed by intersection and complement.

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

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

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, \( a \)-cut is able to keep the original ranks. Namely, we have \( [\mathcal{D} \land ^a\mathcal{D}](t) = \mathcal{D}(t) \) if \( \mathcal{D}(t) \geq a \) and \( 0 \) otherwise.

Counterparts to selection, join, projection, etc. These operation stem basically from the classical ones by taking into account similarity relations (or, in general fuzzy relations \( \theta \) in place of classical comparators). For illustration, we consider a similarity-based join. For simplicity, consider a ranked table \( \mathcal{D}_1 \) from Tab.I (result to a query “... candidates with age about 30”) and a ranked table \( \mathcal{D}_2 \) from Tab.II (top describing open positions with required education. A similarity-based join \( \mathcal{D}_1 \Join \mathcal{D}_2 \) then describes possible job assignments. A rank \( [\mathcal{D}_1 \Join \mathcal{D}_2]((n,a,e,p) \in (n,a,e,p)) \) of tuple \( (n,a,e,p) \) in \( \mathcal{D}_1 \Join \mathcal{D}_2 \) is given by

\[
\mathcal{V}_{e_1,e_2}(\mathcal{D}_1(n,a,e_1) \odot (e_1 \approx e) \odot (e \approx e_2) \odot \mathcal{D}_2(p,e_2))
\]

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 at the cost of decreasing the value of the resulting tuples by degrees of similarity. The bottom table of Tab.II shows a result of a 0.9-cut of \( \mathcal{D}_1 \Join \mathcal{D}_2 \) projected to name and position.
Further operations (top₁ etc.) Here, we put operations interesting from the point of information retrieval which cannot be accounted for in classical model. As an example, consider top₁ which gained a considerable interest recently, see [13], [14] and also [18]. We define top₁(D) to contain the first k tuples (according to rank ordering) of D with their ranks (if there are less than k ranks in D then top₁(D) = D; and top₁(D)) includes also the tuples with rank equal to the rank of the k-th tuple. Note that top₁ is a part of a query language described in [26].

2) Tuple and domain relational calculi: The tuple calculus of classical model is based on classical predicate logic. In the same spirit (here again, as with relational algebra), our tuple calculus is based on fuzzy predicate logic. It is important for our purpose that predicate fuzzy logic(s) are developed nowadays and that they are in a relationship to the calculus of fuzzy relations similar to the spirit (here again, as with relational algebra), our tuple calculus is based on classical database similar to the calculus of fuzzy relations similar to the predicate fuzzy logic due to which both the relational databases establish an interesting framework different from our one but with similar aims.

C. Functional dependencies

Functional dependencies (FDs) describe a particular form of relationship. FDs are traditionally used for issues related to database design [23] and for obtaining information from data [24]. We are going to argue that in our setting, FDs extended by taking into account the domain similarities (1) provide us with a new type of data dependency; (2) leave many of the previous approaches to fuzzy FD particular cases; (3) are tractable both theoretically and computationally in an analogous way as with classical FDs. Claim (3) is particularly important since most of the previous approaches to fuzzy FD are confined to definitions and illustrative examples.

1) Definition and related approaches: In our setting, a (fuzzy) FD is a formula A ⇒ B where A and B are fuzzy sets of attributes (A, B ∈ L_f). We first present a definition of validity of A ⇒ B in a ranked data table D and then add comments.

Definition 5: For a ranked data table D, tuples t₁, t₂ and a fuzzy set C ∈ L^f of attributes, we introduce a degree t₁(C) ≈ t₂(C) which to t₁ and t₂ have similar values on attributes from C by

\[ t_1(C) \approx_d t_2(C) = (D(t_1) \oplus D(t_2)) \setminus \forall y \in \text{Y}(y \rightarrow (t_1[y] \approx_d t_2[y])) \]  \hspace{1cm} (1)

A degree \(|A \Rightarrow B|_D|_D|_D_0 \) to which a FD A ⇒ B is true in D is defined by \(|A \Rightarrow B|_D|_D|_D_0 = \forall t_1, t_2 ((t_1(A) \approx_d t_2(A) \rightarrow (t_1(B) \approx_d t_2(B))) \) . (2)

Remark 6: (1) By basic rules of semantics of predicate fuzzy logic [17], \( t_2(C) \approx_d t_2(C) \) is just the truth degree of a formula “if \( t_1, t_2 \) are from D then for each attribute \( y \in C \). t₁ and t₂ have similar values on \( y \)”,

(2) Therefore, using predicate fuzzy logic again, \(|A \Rightarrow B|_D|_D|_D_0\) is a truth degree of a formula “for any tuples \( t_1, t_2 \) if \( t_1, t_2 \) have similar values on attributes from A then \( t_1 \) and \( t_2 \) have similar values on attributes from B”. Note that due to our adherence to predicate fuzzy logic, the meaning of \( A \Rightarrow B \) is given by a simple formula which we just described in natural language. Note that, in fact, the antecedent in formula (2) is modified by a hedge *. This has technical reasons not discussed in detail here (note only that setting * to globalization or identity enables as to have some of the previous approaches as particular cases of our ones).

(3) Note also that \(|A \Rightarrow B|_D|_D|_D_0|_D\) is a truth degree from our scale L, not necessarily being 0 or 1. That is, our FDs may be true to a degree, e.g., 0.9 (approximately true) which is natural when considering approximate concepts like similarity. The particular value of \(|A \Rightarrow B|_D|_D|_D_0|_D|_D|_D|_D_0\) depends on our choice of the scale and the connectives. For illustration, if the ranks in D are all 0 or 1 and * is globalization then for any choice of a scale L and connectives \( \ominus, \oplus \), we have that \(|A \Rightarrow B|_D|_D|_D_0 = 1 \) (A ⇒ B is true in D) means that for any tuples \( t_1, t_2 \) from D: if \( A(y) \leq (t_1[y] \approx_d t_2[y]) \) for any attribute \( y \in C \) then \( B(y) \leq (t_1[y] \approx_d t_2[y]) \) for any attribute \( y \in Y \). This also shows that degrees \( A(y) \) and \( B(y) \) serve basically as similarity thresholds.

(4) Compared to previous approaches to fuzzy FDs, see [4], [5], [10], [19], [27], [28], [30] for a representative sample, the following are the main distinctions of our approach. (i) Previous approaches use ordinary sets A and B in a fuzzy FD A ⇒ B which is a special case in our setting since we allow fuzzy sets for A and B. This allows us to express similarity thresholds (see above) which means a greater but still natural expressive power. (ii) Previous approaches do not consider approximate validity of FDs and related notions (like degree of entailment). (iii) Previous approaches use one particular scale, namely \[0, 1], and one particular choice of connectives and do not consider relationship to predicate fuzzy logic. As a result, a simple natural language description of the meaning of fuzzy FD is missing. Furthermore, most of the previous approaches are a particular case of our approach. Due to the limited scope of our paper, we present a detailed comparison elsewhere.

2) Armstrong-like axioms and completeness: In this section, we present two kinds of complete axiomatization of our FDs by means of Armstrong-like axioms. Instead of proceeding directly (which is possible), we follow a shorter path by using results from [2] where complete axiomatizations were shown for fuzzy FDs evaluated over non-ranked data tables (i.e., ranks equal to 0 or 1 in our setting).

First, we need the following concepts. For a set T of fuzzy FDs, let Mod(T) be a set of all ranked data tables with similarities in which
Our axiomatic system consists of the following deduction rules. As usual, \( A \rightarrow B \) is called 
\begin{align*}
\text{(Ax)} & \quad \text{infer } A \land B \Rightarrow A, \\
\text{(Cut)} & \quad \text{from } A \Rightarrow B \text{ and } B \cup C \Rightarrow D \text{ infer } A \cup C \Rightarrow D, \\
\text{(Mul)} & \quad \text{from } A \Rightarrow B \text{ infer } c \ast A \Rightarrow c \ast B
\end{align*}
for each \( A, B, C, D \in \mathbb{L}^2 \), and \( c \in \mathbb{L} \). Here, \( c \ast A \in \mathbb{L}^2 \) is defined by \( (c \ast A)(y) = c \ast (A(y)) \). We can show that \( A \Rightarrow B \) holds.

Theorem 8 (Completeness): Let \( T \) be a set of \( \mathbb{L} \)-valued formulas, and \( \mathbf{L} \) and \( \mathbf{Y} \) be finite. For each \( A \Rightarrow B \) we have
\[ T \vdash A \Rightarrow B \quad \text{iff} \quad | A \Rightarrow B |_T = 1. \]

Proof: Sketch of the proof: The “\( \Rightarrow \)”-part of the claim (soundness) is routine to check by induction on length of a proof. Hint: observe that \( (A \Rightarrow B) \) is fully true in each ranked data table, and (Cut) and (Mul) infer fully true \( \mathbb{L} \)-valued formulas (in \( \mathbb{L} \) from fully true \( \mathbb{L} \)-valued formulas (in \( \mathbb{L} \)).

Due to Theorem 7, we can restrict ourselves only to models from \( \mathcal{M}(T) \).

In order to show the “\( \Leftarrow \)”-part of the claim, it suffices to show that \( T \vdash A \Rightarrow B \) implies \( | A \Rightarrow B |_T \neq 1 \). Assuming \( T \vdash A \Rightarrow B \), we find a ranked data table \( D \in \mathcal{M}(T) \) such that \( | A \Rightarrow B |_D \neq 1 \). We can check that \( \mathcal{M}(T) \) is a model of \( T \), and that \( | A \Rightarrow B |_D \neq 1 \). Details are postponed to the full version of the paper.

Theorem 9 (Graded Completeness): Let \( \mathbf{L} \) and \( \mathbf{Y} \) be finite. Then for every \( T \) and \( A \Rightarrow B \) we have
\[ | A \Rightarrow B |_T = | A \Rightarrow B |_T^\mathbf{L}. \]

Proof: Due to Theorem 8, it is enough to show that \( | A \Rightarrow B |_T = | A \Rightarrow B |_T^{\mathbf{L}} \) (i.e., the set of all \( \mathbb{L} \)-valued formulas such that any other FD true in \( \mathcal{D} \) that \( A \Rightarrow B \) is provable from \( T \) by
\[ | A \Rightarrow B |_T = \mathcal{V} \subseteq \mathbb{L}. \]
Then, the concept of a degree of provability coheres with that of a degree of semantic entailment.

Theorem 10: (1) The presented results generalize well-known results on completeness of Armstrong axioms [23]. Our results “became” the classical ones if we take a two-element Boolean algebra for our scale of truth degrees with connectives (classical conjunction and implication for \( \ast \) and \( \Rightarrow \), and identity for \( = \)).

(2) Compared to the previous approaches to axiomatization of fuzzy FDs, note first that we deal with a general notion of a FD (see above). Our axioms differ from those reported in the literature on fuzzy FDs in that all of the reported results use only a set of classical Armstrong axioms. Contrary to that, we need a new rule (Mul). Furthermore, the previous approaches did not consider entailment in degrees and thus there are no attempts reported on graded completeness.

(3) Note also that in the previous approaches, the authors prove their completeness results directly. Since they use only the classical Armstrong axioms, it might be interesting to see if their completeness results follow from the completeness of classical FDs. This is, indeed, the case; we omit details (sketch for the case presented in [28]: for each data table with similarities one can construct a classical data table such that the tables have the same true FDs; the result then follows by a simple reasoning on semantic entailment).

3) Computing non-redundant basis: In this section, we focus on non-redundant bases of FDs of ranked data tables, i.e. minimal sets \( T \) of FDs which are fully true in a given ranked table \( \mathcal{D} \) and such that any other FD true in \( \mathcal{D} \) follows semantically from \( T \) in degree 1. Non-redundant bases are therefore minimal sets of FDs...
which convey information about all fully true FDs in the table. The interest in obtaining non-redundant bases is basically twofold. First, from the point of view of knowledge extraction, a ranked data table $\mathcal{D}$ represents an answer to a similarity-based query. A non-redundant basis of $\mathcal{D}$ thus represents an additional information to the query which describes all dependencies satisfied by the result to the query. Second, as in the classical case, non-redundant sets of FDs are important in considerations concerning data redundancy and normalization (this applies particularly to non-ranked tables).

Computational aspects of fuzzy approaches to FDs are scarcely discussed in the literature and [30] seems to be an exception. However, since the aim in [30] is different from computing non-redundant bases, we do not discuss it here (in [30], the authors compute all FDs satisfying some non-triviality conditions).

In what follows, we make use of [3] where the problem of description and computation of a particular non-redundant basis was solved for non-ranked data tables with similarities (i.e., all ranks equal to 1 or 0). We present a couple of results which make it possible to apply results from [3] to the problem of non-redundant bases of ranked data tables. This way, we extend the results and methods of [3] to account for the more general case of ranked data tables.

Let thus $\mathcal{D}$ be a ranked data table with similarities.

**Definition 11:** A set $T$ of FDs is complete in $\mathcal{D}$ if, for each $A \Rightarrow B$, $|A| = |B|$. Moreover, if $T$ is complete in $\mathcal{D}$ and no proper subset of $T$ is complete in $\mathcal{D}$, we call $T$ a non-redundant basis of $\mathcal{D}$. $T$ is called a minimal basis of $\mathcal{D}$ if $T$ is complete in $\mathcal{D}$ and for each $T'$ which is complete in $\mathcal{D}$ we have $|T| \leq |T'|$.

We now proceed in two steps: First, we define a special closure operator $C_{\mathcal{D}}$ which assigns to any fuzzy set $A$ of its attributes closure $C_{\mathcal{D}}(A)$, which is again a fuzzy set of attributes, so that $T = \{ A \Rightarrow C_{\mathcal{D}}(A) | A \in \mathcal{L}^1 \}$ is complete in $\mathcal{D}$. Second, we describe a “small subset” of $T$ which is non-redundant (and minimal in size in some important cases) and computationally tractable. The first part of the procedure (description of $C_{\mathcal{D}}$) is treated in more detail because it is a non-trivial extension of previous results. The second part (selecting a non-redundant subset) follows the same procedure as in [3], so we give only a hint.

**Definition 12:** For a ranked data table $\mathcal{D}$ over attributes $\mathcal{Y}$ define an operator $C_{\mathcal{D}}: \mathcal{L}^1 \rightarrow \mathcal{L}^1$ by

$$(C_{\mathcal{D}}(A))(\gamma) = \bigwedge_{\mathcal{Y}} \{ (\mathcal{D}(t) \otimes (\gamma(t) = t(\gamma(A)) \Rightarrow t(\gamma))) \} \otimes (\gamma(\gamma) \approx t(\gamma))$$

Observe that the tuples $t$ for which $\mathcal{D}(t) = 0$ can be disregarded in the formula for $C_{\mathcal{D}}$.

**Theorem 13:** For each $\mathcal{D}$, $C_{\mathcal{D}}$ is a closure operator, and $T = \{ A \Rightarrow C_{\mathcal{D}}(A) | A \in \mathcal{L}^1 \}$ is complete in $\mathcal{D}$.

**Proof:** Sketch of the proof: Using properties of residuated lattices and hedges, one can show that $C_{\mathcal{D}}$ is a closure operator (the proof is technically involved and omitted due to the lack of space). For the second part, it suffices to show that models of $T$ entail exactly the same FDs as $\mathcal{D}$ does. This can be proved by showing that $\mathcal{D}$ is a model of $T$ (which follows from the definition of $C_{\mathcal{D}}$) and that each model of $T$ entails all FDs which are entailed by $\mathcal{D}$ (hint: suppose some model of $T$ does not entail $A \Rightarrow B$, from which one gets $A \Rightarrow B \notin T$, i.e., $B \not\subseteq C_{\mathcal{D}}(A)$, which further gives $|A| = |B| \not\subseteq 1$ by definition of $C_{\mathcal{D}}$).

We now focus on finding a non-redundant basis of $\mathcal{D}$ which is a subset of the set $T$ described in Theorem 13. Similarly as in [3], we take advantage of the technical concept of a system of pseudo-closed fuzzy sets of attributes. In the present setting of tables with truth weighted tuples, we define the notion as follows. Given $\mathcal{D}$, a collection $P \subseteq \mathcal{L}^1$ of fuzzy sets of attributes is called a system of pseudo-closed fuzzy sets w.r.t. $\mathcal{D}$ if, for each $P \in \mathcal{L}^1$, we have:

$$P \in P \text{ if } P \neq C_{\mathcal{D}}(P) \text{ and for each } Q \in P$$

such that $Q \neq P$: $S(Q, P) = S(C_{\mathcal{D}}(Q), P)$, where “$S(\cdot, \cdot)$” denotes degrees of subsethood, see Section I-C. Each $P \in \mathcal{L}^1$ is then called a pseudo-closed fuzzy set of attributes. One can prove the following assertion (the proof is postponed to the full version of the paper).

**Theorem 14:** If $P$ is a system of pseudo-closed fuzzy sets w.r.t. $\mathcal{D}$, then $T = \{ P \Rightarrow C_{\mathcal{D}}(P) | P \in \mathcal{L}^1 \}$ is a non-redundant basis of $\mathcal{D}$. If

**Remark 15:** The non-redundant basis $T$ of a ranked table $\mathcal{D}$ of Theorem 14 can be efficiently computed by an algorithm with polynomial time delay. Namely, the systems of pseudo-closed fuzzy sets introduced in our paper satisfy the requirements of the algorithms proposed for non-ranked data tables [3, Theorem 5, Algorithm 1]. We omit the presentation of the resulting algorithm due to space limitations.

**Example.** We now present an example of a non-redundant basis of a ranked table. We consider a linear scale of 11 truth

**TABLE III**

<table>
<thead>
<tr>
<th>country</th>
<th>coal</th>
<th>gr</th>
<th>water</th>
<th>nuclear</th>
</tr>
</thead>
<tbody>
<tr>
<td>China</td>
<td>498.0</td>
<td>246</td>
<td>196</td>
<td>34.6</td>
</tr>
<tr>
<td>India</td>
<td>154.3</td>
<td>1032</td>
<td>75</td>
<td>26.8</td>
</tr>
<tr>
<td>USA</td>
<td>570.7</td>
<td>2533</td>
<td>330</td>
<td>753.9</td>
</tr>
<tr>
<td>Russia</td>
<td>115.8</td>
<td>54</td>
<td>157</td>
<td>122.5</td>
</tr>
<tr>
<td>Japan</td>
<td>0.0</td>
<td>120</td>
<td>90</td>
<td>293.8</td>
</tr>
<tr>
<td>Germany</td>
<td>56.4</td>
<td>3817</td>
<td>50</td>
<td>161.2</td>
</tr>
<tr>
<td>UK</td>
<td>19.5</td>
<td>350</td>
<td>8</td>
<td>81.7</td>
</tr>
<tr>
<td>France</td>
<td>0.0</td>
<td>63</td>
<td>62</td>
<td>394.4</td>
</tr>
<tr>
<td>Spain</td>
<td>10.9</td>
<td>1180</td>
<td>11</td>
<td>58.9</td>
</tr>
</tbody>
</table>

**TABLE IV**

<table>
<thead>
<tr>
<th>country</th>
<th>coal</th>
<th>gr</th>
<th>water</th>
<th>nuclear</th>
</tr>
</thead>
<tbody>
<tr>
<td>China</td>
<td>498.0</td>
<td>246</td>
<td>196</td>
<td>34.6</td>
</tr>
<tr>
<td>India</td>
<td>154.3</td>
<td>1032</td>
<td>75</td>
<td>26.8</td>
</tr>
<tr>
<td>USA</td>
<td>570.7</td>
<td>2533</td>
<td>330</td>
<td>753.9</td>
</tr>
<tr>
<td>Russia</td>
<td>115.8</td>
<td>54</td>
<td>157</td>
<td>122.5</td>
</tr>
<tr>
<td>Japan</td>
<td>0.0</td>
<td>120</td>
<td>90</td>
<td>293.8</td>
</tr>
<tr>
<td>Germany</td>
<td>56.4</td>
<td>3817</td>
<td>50</td>
<td>161.2</td>
</tr>
<tr>
<td>UK</td>
<td>19.5</td>
<td>350</td>
<td>8</td>
<td>81.7</td>
</tr>
<tr>
<td>France</td>
<td>0.0</td>
<td>63</td>
<td>62</td>
<td>394.4</td>
</tr>
<tr>
<td>Spain</td>
<td>10.9</td>
<td>1180</td>
<td>11</td>
<td>58.9</td>
</tr>
</tbody>
</table>
degrees 0 (falsity) < 0.1 < 0.2 < · · · < 1 (full truth) equipped with Lukasiewicz connectives [17] and globalization. Table III describes power consumption of selected countries. The attributes denote name of the county, mass of coal (megatons) produced for power purposes, electricity (MW) produced by air power-plants, electricity (MW) produced by water power-plants, electricity (MW) produced by nuclear power-plants. For simplicity, we use names as tuples’ identifiers of tuples instead of values of attributes.

Introducing similarity relations enables us to gain more information from the data. Let our similarities be given by Table IV. Our purpose is either to study methods of specifying suitable similarities for particular data nor argue that our choice of similarities is “the best one”–this is a matter connected with particular problem domain (geography and economy, in this particular example) and should be left to experts in the areas.

Suppose first that a rank of each tuple in Table III is 1. Then the minimal basis of such a data table (with the underlying similarity relations) consists of 56 FDs.

If ranks of tuples are as given by the D(r)-column of Table III, then the table can be seen as a result of a query “select power consumption of countries with very large populations”. Intuitively, one may expect the the minimal basis of such a table would be smaller than the basis of the latter one because now several tuples (like Spain, France, . . . ) have a low rank (the populations are rather small). Indeed, the minimal basis given by the algorithm described in previous section is (after reduction of left-hand and right-hand sides of FDs) the following:

\[
\begin{align*}
\{c, 0.0/w\} & \Rightarrow \{w\}, \\
\{0.1/c\} & \Rightarrow \{0.4/c, 0.4/w\}, \\
\{0.09/c\} & \Rightarrow \{0.09/w\}, \\
\{0.09/a\} & \Rightarrow \{0.7/c, 0.3/a, 0.7/w\}, \\
\{0.09/c\} & \Rightarrow \{a, n\}, \\
\{0.09/n\} & \Rightarrow \{n\}, \\
\{0.09/a\} & \Rightarrow \{0.09/n\}, \\
\{0.09/w\} & \Rightarrow \{0.09/n\}, \\
\{0.09/a\} & \Rightarrow \{0.09/n\}, \\
\{0.09/w\} & \Rightarrow \{0.09/n\}, \\
\{0.09/c\} & \Rightarrow \{0.09/w\}. 
\end{align*}
\]

The basis can be seen as an additional information supplied along with the query result. Note that if Table III is considered as a classical one (no ranks, no similarities), its minimal basis consists of three (classical) FDs, namely \(\{a\} \Rightarrow \{c, w, n\}\), \(\{w\} \Rightarrow \{c, a, n\}\), and \(\{n\} \Rightarrow \{c, a, w\}\). Thus, attributes a, w, and n are all keys of the table. Contrary to the previous case with similarities and ranks, the basis does not yield any other (nontrivial) information.

### III. Future research

Future research needs to focus on further development of the relational algebra and calculi (both classical aspects like query optimization and the new ones arising due to taking degrees into account); development of functional dependencies and further types of data dependencies; data redundancy and related issues (keys, normalization in presence of similarity, preliminary results are available).

### Acknowledgment

Supported by grant No. 1ET101370417 of GA AV ČR, by grant No. 201/05/0079 of the Czech Science Foundation, and by institutional support, research plan MSM 6198959214.

### References


