General Relational Data Model with Preferences

Post-Graduate Student:
RADIM NEDBAL
Institute of Computer Science
Academy of Sciences of the Czech Republic
Pod Vodárenskou věží 2
182 07 Praha 8
Czech Republic ,

Department of mathematics
Faculty of nuclear science and physical engineering
ČVUT
Trojanova 13
120 00 Praha 2

Czech Republic

radned@seznam.cz

Supervisor:
ING. JÚLIUS ŠTULLER, CSC.

Institute of Computer Science
Academy of Sciences of the Czech Republic
Pod Vodárenskou věží 2
182 07 Praha 8

Czech Republic

stuller@cs.cas.cz

Field of Study:
Mathematical Engineering

This work was supported by the project 1ET100300419 of the Program Information Society (of the Thematic Program II of the National Research Program of the Czech Republic) "Intelligent Models, Algorithms, Methods and Tools for the Semantic Web Realization", and by the Institutional Research Plan AV0Z10300504 "Computer Science for the Information Society: Models, Algorithms, Applications".

Abstract

The aim of the paper is to present a novel, general approach to preference modelling in the framework of the relational data model. To allow nonmonotonic operations, the preferences are defined between sets of relational instances. The aim is the generalization of the relational algebra that is as minimal as possible, in the sense that the formal fundamentlas of the relational data model are preserved. At the same time, the extended model should be formal enough to provide a sound basis for the investigation of other new preference constructors and operations and for new possible applications.

1. Related Work

Lacroix and Lavency [1] originated the study of preference queries. They proposed an extension of the relational calculus in which preferences for tuples satisfying given logical conditions can be expressed. For instance, one could say: Pick up the tuples of R satisfying $Q \wedge P1 \wedge P2$; if the result is empty, pick the tuples satisfying $Q \wedge P1 \wedge \neg P2$; if the result is empty, pick the tuples satisfying $Q \wedge \neg P1 \wedge P2$. The composition or iteration of preferences, however, is not considered. Neither is addressed the issue of algebraic optimization of preference queries.

Kießling et al. [2, 3, 4, 5, 6, 7] and Chomicki et al. [8, 9, 10, 11] proposed independently a similar framework based on a formal language for formulating preference relations. The embedding (called *Best Match Only* – BMO and *WinNow* – WN respectively) into relational query languages they use is identical. Many possible rewritings for preference queries are presented.

Kießling et al. [2, 6] introduce a number of base preference constructors and their combinators (Pareto and lexicographic composition, intersection, disjoint union, and others). The possibility of having arbitrary constraints in preference formulas is not considered.

The framework of Chomicki et al. [8] emphasizes the view of preferences as strict partial orders and defines preferences more generally as arbitrary logical formulas. Intrinsic and extrinsic classes of preference formulas are studied.

Börzsönyi et al. [12] introduced the skyline operator and described several evaluation methods for this operator. Skyline is a special case of WN and BMO. It is restricted to use an intrinsic preference formula which is a conjunction of pairwise comparisons of corresponding tuple components. Some examples of possible rewritings for skyline queries are given but no general rewriting rules are formulated.

Argawal and Wimmers [13] use quantitative preferences (scoring functions) in queries and focus on the issues arising in combining such preferences. Hristidis et al. [14] explore in this context the problems of efficient query processing using materialized views. As pointed out repeatedly in their paper, the approach based on scoring functions is inherently less expressive than the one based on preference relations. In particular, skyline queries cannot be captured using scoring functions. Moreover, since the quantitative approach is based on comparing the scores of individual tuples under the given scoring functions, the preferences represented in this way have to be intrinsic. In addition, it is not clear how to compose scoring functions to achieve an effect similar to various preference relation composition operations.

A more general approach is proposed in [15], where the relational data model is extended to incorporate partial orderings into data domains. Within the extended model, the partially ordered relational algebra (the PORA) is defined by allowing the ordering predicate to be used in formulae of the selection operation. The PORA expresses exactly the set of all possible relations that are invariant under order-preserving automorphism of databases. This result characterizes the expressiveness of the PORA and justifies the development of Ordered SQL (OSQL) as a query language for ordered databases. OSQL provides users with the capability of capturing the semantics of ordered data in many advanced applications, such as those having temporal or incomplete information.

A similar approach to preference modelling is presented in [16]. A declarative query interface for Web repositories that supports complex expressive Web queries is defined. Such queries have two key characteristics: (i) They view a Web repository simultaneously as a collection of text documents, as a navigable directed graph, and as a set of relational tables storing properties of Web pages (length, URL, title, etc.). (ii) The queries employ application-specific ranking and ordering relationships over pages and links to filter out and retrieve only the "best" query results. The Web repository is modelled in terms of "Web relations". A description of an algebra for expressing complex Web queries is given. The algebra extends traditional relational operations as well as graph navigation operations to uniformly handle plain, ranked, and ordered Web relations. In addition, an overview of the cost-based optimizer and execution engine is presented.

In [17], actual values of an arbitrary attribute are allowed to be partially ordered as well. Accordingly, relational algebra operations, aggregation functions and arithmetic are redefined. Thus, on one side, the expressive power of the classical relational model is preserved, and, at the same time, as the new operations operate on and return ordered relations, information of preference, which is represented by a partial ordering, can be handled. Nevertheless, the redefinition of the relational operations causes loss of some of their common properties. For instance, $A \cap B = A - (A - B)$ does not hold. To rectify this weak point, more general approach is needed.

A comprehensive work on partial order in databases is [18]. It presents the partially ordered sets as the basic construct for modelling data. Collection of algebraic operations for manipulating ordered sets is investigated, and their implementation based on the use of realizers as a data structure is presented. An algorithm for generating realizers for arbitrary finite partial orders is provided.

Various kinds of ordering on powerdomains have been considered in context of modelling incomplete information. A very extensive and general study is provided in [19].

In the context of financial and statistical applications, systems such as SEQUIN [20], SRQL [21], and more recently Aquery [22, 23] have proposed SQL extensions to incorporate ordering.

2. Proposed Approach

All the above approaches miss out defining nonmonotonic operations, e.g. set difference, on relations with partial ordering incorporated into attribute or relation domains. An exception is [17] whose generalized difference operation, however, does not preserve common algebraical equalities. Obviously, a more general model is needed. The proposed solution is to take all relational instances into account instead of mere tuples. To see why, refer to the following paragraph and the following example.

The framework of relational instances with ordered tuples can be understood as a generalization of *fuzzy sets-based model* [24, 25] of uncertainty. The same way as a fuzzy relational instance can be described by its alfa cuts, an instance R^* of an arbitrary relation R with an ordering $\preceq^{R^*}$ can be described by its subset containing appropriate tuples. The intuition suggests considering tuples according to their preferences. That is to say, we take into account the more preferred tuples ahead of those with lower preference. Thus, for each tuple, t_i, we take into account a set, $t_i{\uparrow}$, containing this tuple and all the tuples with higher preferences:

$$t_i{\uparrow} = \{t \mid t \in R^* \ \wedge \ t_i \preceq^{R^*} t\}$$

Note that $t_i{\uparrow} \in \mathscr{I}(R)$. Next, considering a unary relational operation

$$\mathcal{O}_u : \mathscr{I}(R) \to \mathscr{I}(Q),$$

which is a mapping from a set of relational instances of R into a set of relational instances of resulting relation Q, it is applied to each set $t_i{\uparrow}$. Note that $\mathcal{O}_u(t_i{\uparrow}) \in \mathscr{I}(Q)$. Finally, the order $\preceq^{\mathscr{I}(Q)}$ on the resulting collection

$$\{\mathcal{O}_u(t_i{\uparrow}) \mid t_i \in R^*\} \subseteq \mathscr{I}(Q)$$

of sets is to be determined.

Example 1 Consider a set $\{\mathcal{O}_u(t_i{\uparrow}) \mid t_i \in R^*\} \subseteq \mathscr{I}(Q)$ with a relation $\sqsubseteq$ implied by preference $\preceq^{R^*}$ on R^*:

$$\mathcal{O}_u(t_i{\uparrow}) \sqsubseteq \mathcal{O}_u(t_j{\uparrow}) \Leftrightarrow t_i \preceq^{R^*} t_j$$

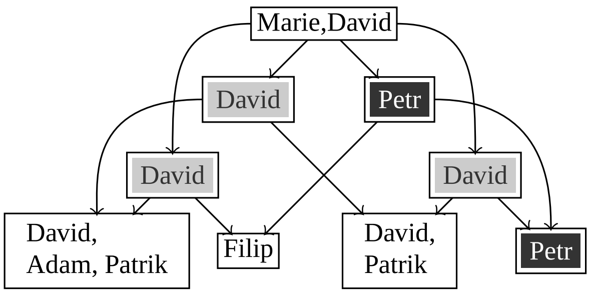

Figure 1: $\{\mathcal{O}_u(t_i{\uparrow}) \mid t_i \in R^*\} \subseteq \mathscr{I}(Q)$ with a relation $\sqsubseteq$

Note that $\mathcal{O}_u$ generally is not an injection. In other words, $\mathcal{O}_u(t_i{\uparrow}) = \mathcal{O}_u(t_j{\uparrow})$ for some $t_i{\uparrow} \neq t_j{\uparrow}$. In particular, $\mathcal{O}_u(t_i{\uparrow}) = \mathcal{O}_u(t_j{\uparrow}) =$ "Petr" and $\mathcal{O}_u(t_k{\uparrow}) = \mathcal{O}_u(t_l{\uparrow}) = \mathcal{O}_u(t_m{\uparrow}) =$ "David". To get an ordering, we need to resolve these duplicities:

- First, as the occurrences of "Petr" are in the relation $\sqsubseteq$, we drop the less "preferred" one.
- In the case of the triplet of occurrences of "David", we are unable to determine the one with the highest "preference". Nevertheless, notice that:
  - The set $\{$Marie, David$\}$ is preferred to any of the occurrences of "David". In other words, whichever the most preferred occurrence of "David" is, it is less preferred then the set $\{$Marie, David$\}$.

– There is a unique occurrence of “Filip”, for which we can find an occurrence of “David” with a higher preference. In other words, whichever the most preferred occurrence of “David” is, it is preferred more then the occurrence of “Filip”. The same rationale applies for the sets $\{$David, Adam, Patrik$\}$ and $\{$David, Patrik$\}$.

Thus, we get the resulting order, depicted in the following figure:

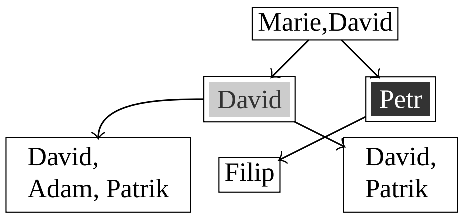

Figure 2: Ordering $\preceq^{\mathscr{I}(Q)}$ on $\{\mathcal{O}_u(t_i\!\upharpoonleft) \mid t_i \in R^*\} \subseteq \mathscr{I}(Q)$ $\square$

To sum up, the order $\preceq^{\mathscr{I}(Q)}$ on the resulting collection $\{\mathcal{O}_u(t_i\!\upharpoonleft) \mid t_i \in R^*\} \subseteq \mathscr{I}(Q)$ of sets is defined as:

$$\mathcal{O}_u(t_i\!\upharpoonleft) \preceq^{\mathscr{I}(Q)} \mathcal{O}_u(t_j\!\upharpoonleft) \Leftrightarrow \\ (\forall t_k \in R^*)\big([\mathcal{O}_u(t_k\!\upharpoonleft) = \mathcal{O}_u(t_i\!\upharpoonleft)] \Rightarrow (\exists t_l \in R^*)[\mathcal{O}_u(t_l\!\upharpoonleft) = \mathcal{O}_u(t_j\!\upharpoonleft) \ \wedge \ t_k \preceq^{R^*} t_l]\big)$$

As the minimal set of relational algebra operations consists of two unary (restriction and projection) and three binary operations (set difference, set union, and cartesian product), a binary operations need also to be considered. In general, a binary operation

$$\mathcal{O}_b : \mathscr{I}(R) \times \mathscr{I}(R') \to \mathscr{I}(Q)$$

is a mapping from a couple of sets of relational instances of R and R' into a set of relational instances of resulting relation Q. As in the foregoing example, we get a resulting collection

$$\{\mathcal{O}_b(t_i\!\upharpoonleft, t'_k\!\upharpoonleft) \mid (t_i, t'_k) \in R^* \times R'^*\} \subseteq \mathscr{I}(Q)$$

of sets, and the order $\preceq^{\mathscr{I}(Q)}$ definition:

$$\mathcal{O}_b(t_i\!\upharpoonleft, t'_k\!\upharpoonleft) \preceq^{\mathscr{I}(Q)} \mathcal{O}_b(t_j\!\upharpoonleft, t'_l\!\upharpoonleft) \Leftrightarrow \\ [\forall (t_m, t'_p) \in R^* \times R'^*]\big([\mathcal{O}_b(t_m\!\upharpoonleft, t'_p\!\upharpoonleft) = \mathcal{O}_b(t_i\!\upharpoonleft, t'_k\!\upharpoonleft)] \Rightarrow \\ [\exists (t_n, t'_q) \in R^* \times R'^*][\mathcal{O}_b(t_n\!\upharpoonleft, t'_q\!\upharpoonleft) = \mathcal{O}_b(t_j\!\upharpoonleft, t'_l\!\upharpoonleft) \ \wedge \ t_m \preceq^{R^*} t_n \ \wedge \ t'_p \preceq^{R'^*} t'_q]\big)$$

What are the consequences of this approach? Generally

$$\mathcal{O}_b(t_i\!\upharpoonleft, t'_k\!\upharpoonleft) \preceq^{\mathscr{I}(Q)} \mathcal{O}_b(t_j\!\upharpoonleft, t'_l\!\upharpoonleft) \Rightarrow \mathcal{O}_b(t_i\!\upharpoonleft, t'_k\!\upharpoonleft) \supseteq \mathcal{O}_b(t_j\!\upharpoonleft, t'_l\!\upharpoonleft)$$

does not hold for nonmonotonic operations (consider the relational operation of difference). With respect to the relational property of *closure*, it is clear that the framework for defining preference on the tuples of relational instances needs to be generalized. We need to express the preference structure on the powerset $\mathscr{I}(R)$ of all possible instances R^* of a relation R.

The foregoing definitions of orderings on sets correspond to Hoare’s ordering. It is one of the orderings provided by the study of semantics of non-determinism [26, 19]. Possibility of employing another semantics, however, needs further investigation.

3. Summary and Future Work

A framework for relational algebra with preferences is proposed. It should present a basis for semantically rich, easy to handle and flexible preference model aiming at deep personalization of database queries. It differs from the other above mentioned approaches in that the preferences are not part of the queries but part of relational instances. Is is shown that they need to be defined between sets of elements – relational instances. This approach is strictly more general allowing to model nonmonotonic operations.

It can also be shown that the proposed approach generalizes the fuzzy sets-based approach to modelling uncertainty in database systems. The reason is that there exists a homomorphism (A and B stand for sets of attributes of relations R):

$$\Big(\big\{\mathscr{I}_F\big(R(A{\cup}B)\big){\cup}\mathscr{I}_F\big(R(A)\big){\cup}\mathscr{I}_F\big(R(B)\big)\big\};\Omega_{\mathcal{F}}\Big) \longrightarrow \Big(\big\{\mathscr{I}_O\big(R(A{\cup}B)\big){\cup}\mathscr{I}_O\big(R(A)\big){\cup}\mathscr{I}_O\big(R(B)\big)\big\};\Omega_O\Big)$$

of the algebra consisting of

- the support comprising all the fuzzy relation instances that can arise by means of monotonic fuzzy relational algebra operations applied on relations $R(A \cup B), R(A), R(B)$, and of
- monotonic fuzzy relational algebra operations $\Omega_{\mathcal{F}}$

into the algebra consisting of

- the support comprising all the relation instances with ordering that can arise and whose ordering is defined by means of monotonic preference algebra operations applied on relations $R(A \cup B), R(A), R(B)$, and of
- monotonic preference relational algebra operations Ω_O

Moreover, this homomorphism is unique for all t-norms.

Furthermore, it can be shown that associativity and commutativity of the original union, product, restrict, and project operations are retained. Specifically for the generalized restrict operation, the following equivalences, which hold for the classical restrict operation, are retained:

$$\begin{aligned}
\sigma_{\varphi_1\vee\varphi_2}(R) &\equiv \sigma_{\varphi_1}(R)\cup\sigma_{\varphi_2}(R)\\
\sigma_{\varphi_1\wedge\varphi_2}(R) &\equiv \sigma_{\varphi_1}(R)\cap\sigma_{\varphi_2}(R)\\
\sigma_{\neg\varphi}(R) &\equiv R-\sigma_{\varphi}(R)
\end{aligned}$$

Using the proposed approach, other relational operations (intersect, join, and divide), also, retain the usual properties of their classical relational counterparts:

$$\begin{aligned}
R\cap S &\equiv R-(R-S)\\
R\div S &\equiv \pi_{A-B}(R)-\pi_{A-B}\Big(\big(\pi_{A-B}(R)\times S\big)-R\Big)\\
R\bowtie S &\equiv \pi_{A\cup B}\big(\sigma_{\varphi}(R\times S)\big)
\end{aligned}$$

These results are promising for many query optimization issues, which present many open problems. In particular, evaluation and optimization of preference queries, including cost-base optimization need to be addressed.

Another open problem is how the proposed framework fits into the relational data model. A promising approach seems to be an array-based data model proposed by [22, 23]. The key component of this model is an ordered data structure called an *arrable*, for array-table. Informally, it is a collection of named arrays that, in their simplest form, are vectors of elements of base type. In this form, an arrable is essentially a table organized by columns. Among others, arrable facilitates the implementation of algebraic operators by means of *realizers* [18] – a set of linear extensions uniquely characterizing a given partial order. Realizers offer an effective way to implement relational operations with ordering. Nevertheless, a comparison with other possibly effective implementations of relational operations needs further investigation.

List of symbols

$\sqcap(R^*)$	$\{A \mid A$ is a fuzzy subset of $R^*\}$,
$\mathscr{I}(R)$	set of all possible instances R^* of a relation R
$\mathscr{I}_F(R)$	set of all possible fuzzy instances R^{F*} of a relation R
$\mathscr{I}_O(R)$	set of all possible ordered instances of a relation R

References

[1] M. Lacroix and P. Lavency, “Preferences; Putting More Knowledge into Queries.,” in *VLDB* (P. M. Stocker, W. Kent, and P. Hammersley, eds.), pp. 217–225, Morgan Kaufmann, 1987.

[2] W. Kießling, “Foundations of Preferences in Database Systems,” in *Proceedings of the 28th VLDB Conference*, (Hong Kong, China), pp. 311–322, 2002.

[3] W. Kießling, “Optimatization of Relational Preference Queries,” in *Conferences in Research and Practice in Information Technology* (H. Williams and G. Dobbie, eds.), vol. 39, (University of Newcastle, Newcastle, Australia), Australian Computer Society, 2005.

[4] W. Kießling, “Preference Queries with SV-Semantics.,” in *COMAD* (J. Haritsa and T. Vijayaraman, eds.), pp. 15–26, Computer Society of India, 2005.

[5] W. Kießling and B. Hafenrichter, “Algebraic optimization of relational preference queries,” Tech. Rep. 2003-01, Institute of Computer Science, University of Augsburg, February 2003.

[6] W. Kießling, “Preference constructors for deeply personalized database queries,” Tech. Rep. 2004-07, Institute of Computer Science, University of Augsburg, March 2004.

[7] B. Hafenrichter and W. Kießling, “Optimization of relational preference queries,” in *CRPIT ’39: Proceedings of the sixteenth Australasian conference on Database technologies*, (Darlinghurst, Australia, Australia), pp. 175–184, Australian Computer Society, Inc., 2005.

[8] J. Chomicki, “Preference Formulas in Relational Queries,” *ACM Trans. Database Syst.*, vol. 28, no. 4, pp. 427–466, 2003.

[9] J. Chomicki, “Semantic optimization techniques for preference queries,” 2005.

[10] J. Chomicki and J. Song, “Monotonic and nonmonotonic preference revision,” 2005.

[11] J. Chomicki, S. Staworko, and J. Marcinkowsk, “Preference-driven querying of inconsistent relational databases,” in *Proc. International Workshop on Inconsistency and Incompleteness in Databases*, (Munich, Germany), March 2006.

[12] S. Börzsönyi, D. Kossmann, and K. Stocker, “The skyline operator,” in *Proceedings of the 17th International Conference on Data Engineering*, (Washington, DC, USA), pp. 421–430, IEEE Computer Society, 2001.

[13] R. Agrawal and E. Wimmers, “A Framework for Expressing and Combining Preferences.,” in *SIGMOD Conference* (W. Chen, J. F. Naughton, and P. A. Bernstein, eds.), pp. 297–306, ACM, 2000.

[14] V. Hristidis, N. Koudas, and Y. Papakonstantinou, “Prefer: a system for the efficient execution of multi-parametric ranked queries,” in *SIGMOD ’01: Proceedings of the 2001 ACM SIGMOD international conference on Management of data*, (New York, NY, USA), pp. 259–270, ACM Press, 2001.

[15] W. Ng, “An Extension of the Relational Data Model to Incorporate Ordered Domains,” *ACM Transactions on Database Systems*, vol. 26, pp. 344–383, September 2001.

[16] S. Raghavan and H. Garcia-Molina, “Complex queries over web repositories,” 2003.

[17] R. Nedbal, “Relational Databases with Ordered Relations,” *Logic Journal of the IGPL*, vol. 13, no. 5, pp. 587–597, 2005.

[18] D. R. Raymond, *Partial-order databases*. PhD thesis, University of Waterloo, Waterloo, Ontario, Canada, 1996. Adviser-W. M. Tompa.

[19] L. Libkin, *Aspects of partial information in databases*. PhD thesis, University of Pensylvania, Philadelphia, PA, USA, 1995.

[20] P. Seshadri, M. Livny, and R. Ramakrishnan, “The design and implementation of a sequence database system,” in *VLDB ’96: Proceedings of the 22th International Conference on Very Large Data Bases*, (San Francisco, CA, USA), pp. 99–110, Morgan Kaufmann Publishers Inc., 1996.

[21] R. Ramakrishnan, D. Donjerkovic, A. Ranganathan, K. S. Beyer, and M. Krishnaprasad, “Srql: Sorted relational query language,” in *SSDBM ’98: Proceedings of the 10th International Conference on Scientific and Statistical Database Management*, (Washington, DC, USA), pp. 84–95, IEEE Computer Society, 1998.

[22] A. Lerner and D. Shasha, “Aquery: Query language for ordered data, optimization techniques, and experiments,” in *29th International Conference on Very Large Data Bases (VLDB’03)*, (Berlin, Germany), pp. 345–356, Morgan Kaufmann Publishers, September 2003.

[23] A. Lerner, *Querying Ordered Databases with AQuery*. PhD thesis, ENST-Paris, France, 2003.

[24] V. Subrahmanian, *Uncertainty in Databases and Knowledge Bases*, pp. 315–411. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1997.

[25] R. Nedbal, “Fuzzy database systems – concepts and implementation,” Master’s thesis, Czech Technical University, Faculty of Nuclear Sciences and Physical Engineering, Prague, June 2003.

[26] P. Buneman, A. Jung, and A. Ohori, “Using powerdomains to generalize relational databases,” *Theor. Comput. Sci.*, vol. 91, no. 1, pp. 23–55, 1991.