Relational Databases with Ordered Relations

RADIM NEDBAL, *Institute of Computer Science, Academy of Sciences of the Czech Republic, Pod Vodárenskou věží 2, 182 07 Prague 8, Czech Republic, E-mail: radned@seznam.cz*

Abstract

The paper[1] deals with expressing preferences in the framework of the relational data model. Preferences have usually a form of a partial ordering. Therefore the question arises how to provide the relational data model with such an ordering.

Keywords: relational database model, preference, partial ordering, relational algebra operation, aggregation function, arithmetic

1 Introduction

When retrieving data, it is difficult for a user of a classical relational database to express various levels of preferences. Let us start with an illustrative and motivating example.

Example 1.1 (Preferences represented by an ordering)
How could we express our intention to find employees if we have preference for those who speak English, or at least German, or at worst any other germanic language? At the same time, we may similarly have preference for Spanish or French speaking employees to those speaking any other romanic languages. To sum up, we have the following preferences:

A. Germanic languages:
   1. English,
   2. German,
   3. other germanic languages.

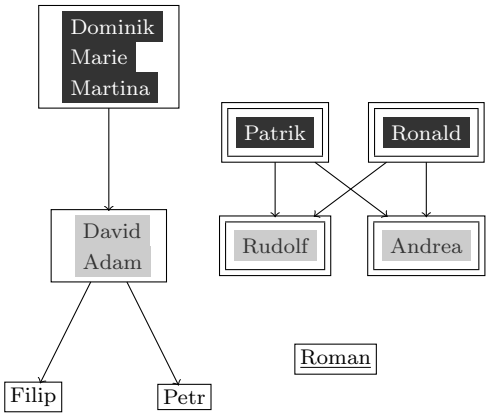
B. Romanic languages:
   1. Spanish of French,
   2. other romanic languages.

These preferences can be formalized by an ordering, in a general case by a partial ordering. The situation is depicted in the following figure. We represent the relation $R(\underline{NAME}, POSITION, LANGUAGE)$ of employees as a table and the above preferences by means of the standard Hasse diagram notation (see the following figure).

Marie is preferred to David as she speaks English and David speaks "just" German. Analogically, Patrik is preferred to Andrea due to his knowledge of French. However Patrik and David, for instance, are "incomparable" as we have expressed no preference order between German and French. Similarly, Roman is "incomparable" to any other employee as Russian is in preference relation with no other language. □

[1]The work was partially 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"

Dominik	President	English
Marie	Manager	English
David	Manager	German
Petr	Manager	Swedish
Adam	Manager	German
Filip	Programmer	Dutch
Martina	Programmer	English
Patrik	Programmer	French
Rudolf	Programmer	Italian
Ronald	Programmer	Spanish
Andrea	Programmer	Portuguese
Roman	Programmer	Russian



The aim of this paper is to define and include semantics of (partial) ordering into relational algebra operations. The resulting data model should provide users with the most relevant data (according to their preferences).

2 The Relational Data Model

The relational data model is based on the mathematical term of a relation. A "table" of a relational database corresponds to a relation and a row of a table is an element of the corresponding relation. However, the relational data model consists not only of the relations themselves, but it provides also operations on relations.

As a relation R is a set, we have all the classical **set operations** and on top of that **aggregation functions** (unary operations returning a number) and **arithmetic** for performing all the usual operations on numbers. The relational data model has been originally defined with eight relational operations:

- Intersection ($\cap$),
- Union ($\cup$),
- Cartesian product ($\times$),
- Difference ($\setminus$),
- Projection ($R[A]$),
- Restriction ($R(\phi)$),
- Join ($\bowtie$),
- Division ($\div$)

Some of the above operations are, however, not primitive [1] – they can be defined in terms of the others. For instance, the intersection, the join, and the division can be defined in terms of the other five. These five operations (union, cartesian product, difference, projection, and restriction) can be then regarded as primitive ones, in the sense that none of them can be defined in terms of the other four. Thus, a minimal set of relational operations is the set consisting of, e.g., these five primitive operations – the so called *minimal set of relational algebra operations.*

3 Operations on Ordered Relations

The ordering represents a new information. To handle this information, we need appropriate operations. To maintain the same expressive power, we need operations corresponding to those that we have for the traditional relational model. In the

following, we consider an ordered pair

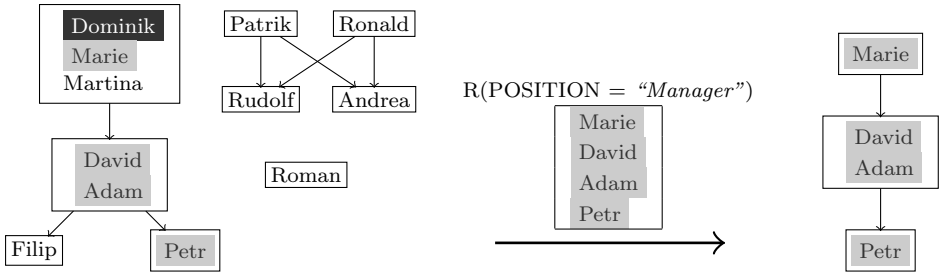
$$[R, \leq^R],$$

of a relation R with its preference relation (ordering) $\leq^R$.

3.1 Relational algebra operations

Restriction $R(\phi)$ returns the relation $\{r \in R \mid \phi(r)\}$ consisting of all tuples from a given relation R that satisfy a specified condition ϕ.

Example 3.1 (Ordering on a restriction)



We prefer Marie to David, Adam, and Petr and at the same time we also prefer David and Adam to Petr as all these preferences hold in the original – input – relation. □

In the case of an ordered relation $[R, \leq^R]$, we define:

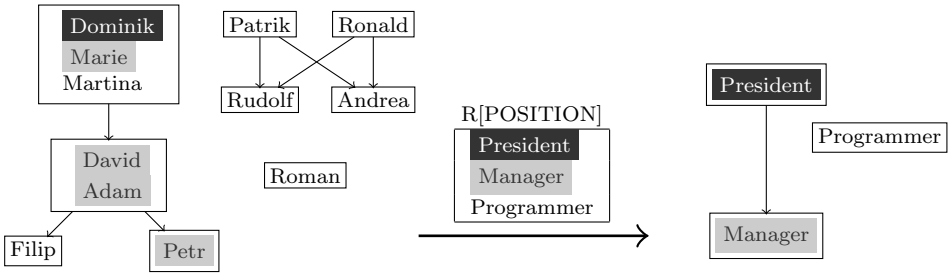
$$[R; \leq^R](\phi) = [R(\phi); \leq^R_{R(\phi)},$$

where

$$\leq^R_{R(\phi)} = R(\phi) \times R(\phi) \cap \leq^R$$

Projection $R[C]$ returns a relation consisting of all tuples that remain as (sub)tuples in a given relation after specified attributes have been eliminated.

Example 3.2 (Ordering on a projection)



We prefer the president to the manager as all presidents (which is in our case the unique element "Dominik") are preferred to all managers (Marie, David Adam, Petr) in the input ordering. However, we can not say anything about preference of the

programmer and the manager as we can find a programmer that is preferred to a manager and vice versa. Equally, we can not say anything about preference order of the programmer and the president as there are programmers that are "incomparable" to the president. □

In the case of an ordered relation $[R, \leq^R]$, we define:

$$[R; \leq^R][C] = \left[R[C]; \leq^{R[C]}\right],$$

where

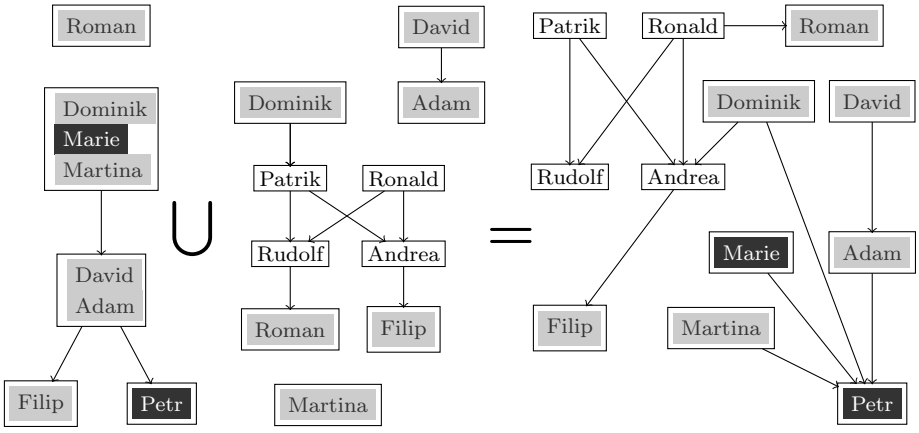
$$\leq^{R[C]} = \big\{(p_i, p_j) \big| (\exists r_i, r_j \in R)(r_i[C] = p_i \ \wedge \ r_j[C] = p_j) \wedge (\forall r_i, r_j \in R)(r_i[C] = p_i \ \wedge \ r_j[C] = p_j \Rightarrow r_i \leq^R r_j)\big\}$$

Union $R_1 \cup R_2$ returns a relation consisting of all tuples appearing in either of the specified relations.

Example 3.3 (Ordering on a union)
Let us consider two input relations $[R_1; \leq^{R_1}], [R_2; \leq^{R_2}]$. What will their union, $[R_1; \leq^{R_1}] \bigcup [R_2; \leq^{R_2}]$, look like? Clearly, we are looking for an ordering on $R_1 \cup R_2$.

First of all, we determine the preference order of the elements belonging to the intersection of the input relations and of the elements belonging to the symmetric difference of the input relations. The preference order is defined in compliance with the preference orders of the input relations elements (cf. the following figure, e.g. Dominik, Filip). If there is no preference order given between elements of a couple in some of the input relations (Dominik, Roman), or the preferences are different, we designate the preference of the couple elements as "incomparable" in the union.

This may seems to be paradoxical at first glance. However, realize we are looking for the overall preference ordering. The condition that allows us to determine the preference order of a pair of elements in the union occurs only when the preference orders of these two elements are identical in both of the input relations. Under all other circumstances, we postulate that we are unable to decide which one of the elements of a pair of elements in the union has preference over the other.



To illustrate this, imagine that we are studying recommendations for hiring new employees. We have received recommendations from independent subjects, and we want to hire the most suitable employee. Incidentally, recommendations mention the same pair of employees. While the first recommendation clearly states the preference order of the candidates, the other one just lists arguments for the candidates suitability. Unfortunately, we are not capable of determining the preference order based on these arguments. Consequently, we are not able to determine the overall preference order between the candidates. If we decide for the candidate preferred to the other by virtue of the first recommendation, we might get unconsciously in contradiction with the second recommendation.

After this first step of determining the preference order of the elements belonging to the intersection of the input relations, we have to determine the preference order of the couples in which one of the elements comes from the intersection and the other one from the symmetric difference of the input relations. To be consistent with the preference determined in the previous step, we have, with respect to the transitivity property of the ordering, to drop "some" preference ordering between couple of elements in the union and designate them as *incomparable*.

Thus we have to designate preference order of (Dominik, Rudolf) as *incomparable*, otherwise the preference order of (Rudolf, Roman) and the transitivity property would imply preference order of (Dominik, Roman), which is in contradiction with the preference designated in the first step. Recall that (Dominik, Roman) were designated as *incomparable*. □

In the case of two ordered relations $[R_1, \leq^{R_1}], [R_2, \leq^{R_2}]$, we define:

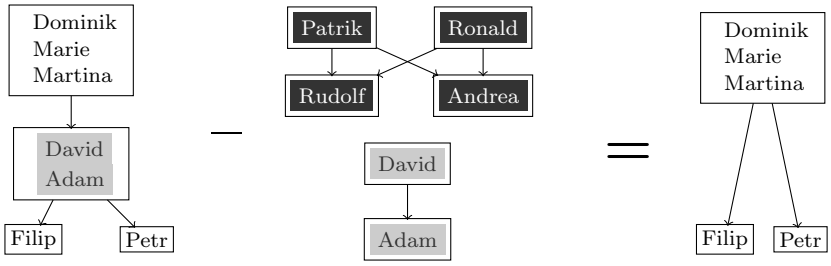
$$[R_1; \leq^{R_1}] \bigcup [R_2; \leq^{R_2}] = [R_1 \cup R_2; \leq^{R_1 \cup R_2}]$$

where

$$\leq^{R_1 \cup R_2} = \overline{\max\left\{R' \middle| \left((\leq^{R_1} \leq^{R'}) \cup (\leq^{R'} \leq^{R_1})\right) \subseteq (\leq^{R_1} \setminus R_2 \times R_2 \cup (\leq^{R_1} \cap \leq^{R_2})\right\} \bigcup}$$
$$\overline{\max\left\{R' \middle| \left((\leq^{R_2} \leq^{R'}) \cup (\leq^{R'} \leq^{R_2})\right) \subseteq (\leq^{R_2} \setminus R_1 \times R_1 \cup (\leq^{R_1} \cap \leq^{R_2})\right\}}$$

Difference $R_1 \setminus R_2$ returns a relation consisting of all tuples appearing in the first relation R_1 and not in the second relation R_2.

Example 3.4 (Ordering on a difference)
Again, consider two input relations. The difference ordering is the restriction of the input ordering on the the difference of the input relations.



□

In the case of ordered relations $[R, \leq^{R_1}], [R_2, \leq^{R_2}]$, we define:

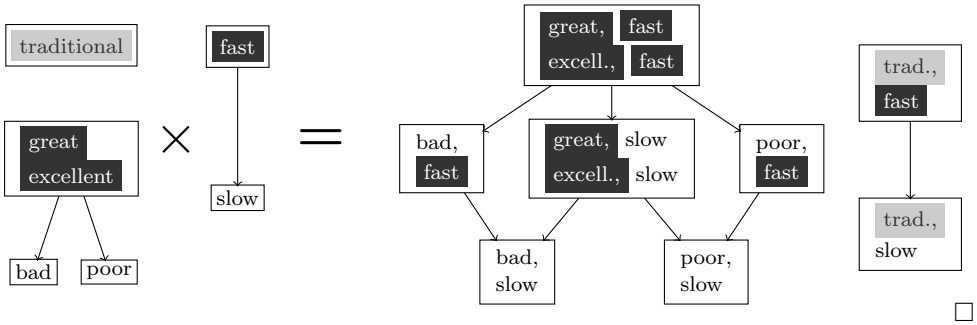
$$[R_1; \leq^{R_1}] \setminus [R_2; \leq^{R_2}] = [R_1 \setminus R_2; \leq^{R_1 \setminus R_2}]$$

where

$$\leq^{R_1 \setminus R_2} = \leq^{R_1} \cap (R_1 \setminus R_2) \times (R_1 \setminus R_2)$$

Cartesian product $R_1 \times R_2$ returns a relation consisting of all possible tuples that are a combination of two tuples, one from each of the specified relations R_1, R_2.

Example 3.5 (Ordering on a cartesian product)
The output ordering is defined as an ordering of ordered pairs.



□

In the case of ordered relations $[R, \leq^{R_1}], [R_2, \leq^{R_2}]$, we define:

$$[R_1; \leq^{R_1}] \times [R_2; \leq^{R_2}] = [R_1 \times R_2; \leq^{R_1 \times R_2}]$$

where

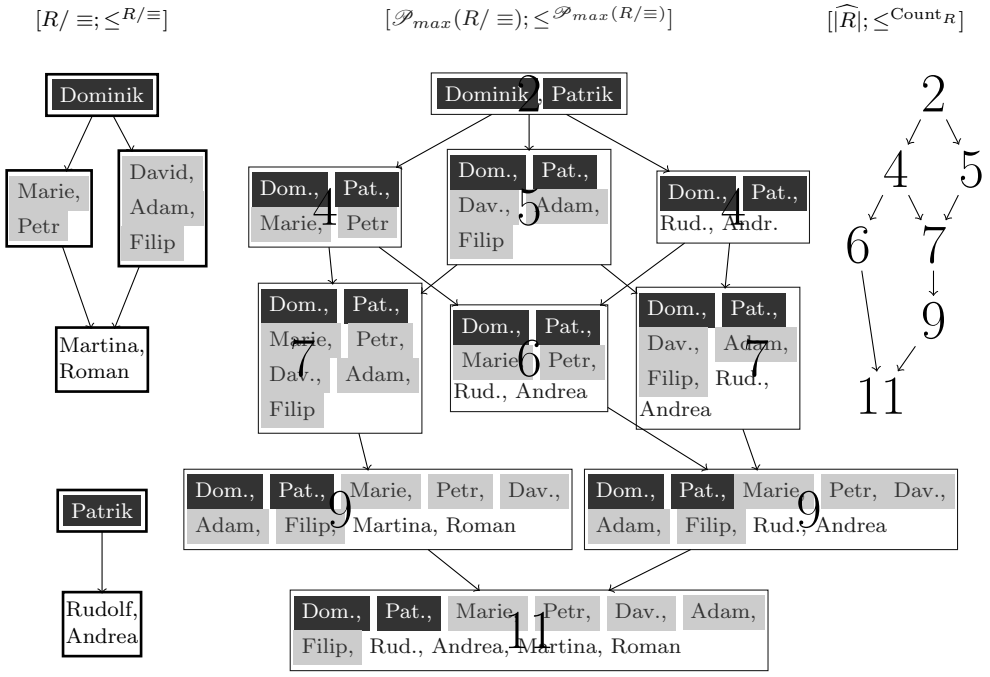
$$\leq^{R_1 \times R_2} = \left\{ \left((r_1, r_2), (r_1', r_2')\right) \middle| (r_1, r_1') \in \leq^{R_1} \wedge (r_2, r_2') \in \leq^{R_2} \right\}, \tag{3.1}$$

which is the direct product definition.

3.2 Aggregation Functions

Example 3.6 (Count on an ordered relation)
Again, let us start with the following illustrative and motivating example depicted on the next page. Consider the ordered pair $[R, \leq^R]$, i.e. a relation R with preference represented as the ordering $\leq^R$: Marie $\leq^R$ Dominik, Petr $\leq^R$ Dominik, David $\leq^R$ Dominik, ... Marie $\leq^R$ Petr, Petr $\leq^R$ Marie, ... Rudolf $\leq^R$ Patrik etc. Notice that not all the elements need to be in the relation $\leq^R$, i.e. some of the elements of R are "incomparable" with respect to $\leq^R$, e.g. Patrick and Roman.

- Next, let us construct equivalence classes $R/\equiv$ containing the elements that we prefer equally: $\{Dominik\}$, $\{Marie, Petr\}$, $\{David, Adam, Filip\}$ etc.
- We can then induce an ordering $\leq^{R/\equiv}$ on $R/\equiv$ of classes of equally preferred elements, which is depicted in the following figure on the left hand side.



- Next, following the definitions (3.2) and (3.3) (see next page), we get the ordered pair

$$[\mathscr{P}_{max}(R/\equiv); \leq^{\mathscr{P}_{max}(R/\equiv)}]$$

  This ordered pair expresses the fact that, first of all, we are interested in the most preferred elements. The less preferred elements are taken into consideration gradually with respect to their preference rank. The principle is never to add elements that are in the hierarchy of the input ordering below the elements that we have not counted yet. The rationale behind this is that one always chooses the best elements possible.

  In this way, we get a lattice ordering of sets containing the maximal number of elements with the preference higher or equal to a certain level.

- Finally, the count operation is performed, and using the definitions (3.4) and (3.5), we get the resulting ordered pair $[|\widehat{R}|; \leq^{\mathrm{Count}_R}]$.

  The semantic of this final ordering can be seen on the couple of 4 and 7 for instance: As we have got $7 \leq^{Count_R} 4$, we can conclude that for any set of 7 elements chosen as the most preferred ones, there is its subset containing 4, more or equally preferred, elements. The elements with an equal preference are always taken into account together. □

From the viewpoint of a formal description, we will consider in the following:

$$[R, \leq^R], \qquad \leq^R \text{ is a relation of a preference on R,}$$

$\leq^R$ is generally unsymmetrical pre-ordering on R as:

$$\begin{aligned} (\leq^R)^0 &\subset \leq^R \\ \leq^R \leq^R &= \leq^R \\ R \times R \setminus \leq^R \cap (\leq^R)^{-1} &\neq \emptyset \end{aligned}$$

We can introduce a relation of equivalence $\equiv$ on R:

$$[R; \equiv], \qquad \equiv \; = \; \leq^R \cap (\leq^R)^{-1}$$

This equivalence induces a factorization, and thus we get a set $[R/\equiv]$ of cosets R_a:

$$R_a = \{r \in R | r \equiv a\}$$

with induced ordering defined as:

$$(\forall R_a, R_b \in R/\equiv)(R_a \leq^{R/\equiv} R_b \iff a \leq^R b)$$

Next, we can define a set $\mathscr{P}_{max}(R/\equiv)$:

$$\begin{aligned} \mathscr{P}(R/\equiv) \supseteq \mathscr{P}_{max}(R/\equiv) = \Big\{ \tilde{R} \subseteq R/\equiv \Big| \\ (\forall R_a \in R/\equiv)\big((\forall R_b \in R/\equiv)(R_a \leq^{R/\equiv} R_b \Rightarrow R_a = R_b) \Rightarrow R_a \in \tilde{R}\big) \quad \wedge \\ (\forall R_a \in \tilde{R})(\forall R_b \in R/\equiv)(R_a \leq^{R/\equiv} R_b \Rightarrow R_b \in \tilde{R})\Big\} \end{aligned} \tag{3.2}$$

It is a set containing all the maximal elements of $[R/\equiv, \leq^{R/\equiv}]$ and all the end stretches of its own elements. Finally, we can define ordering $\leq^{\mathscr{P}_{max}(R/\equiv)}$ on $\mathscr{P}_{max}(R/\equiv)$ as:

$$\big(\forall \tilde{R}_i, \tilde{R}_j \in \mathscr{P}_{max}(R/\equiv)\big)(\tilde{R}_i \leq^{\mathscr{P}_{max}(R/\equiv)} \tilde{R}_j \iff \tilde{R}_i \supseteq \tilde{R}_j), \tag{3.3}$$

which is in fact a lattice ordering. Thus we have defined the ordered pair:

$$[\mathscr{P}_{max}(R/\equiv); \leq^{\mathscr{P}_{max}(R/\equiv)}]$$

An aggregation function in the classical relational data model is a function: $\mathscr{P}(R) \to \mathbb{R}$ (operating on sets and returning numbers). In the relational data model with ordered relations, we define aggregation function *Agg* generally as follows:

Agg: $[\mathscr{P}(R); \leq^R] \longrightarrow \big\{[\mathbb{R}; \leq^{\mathrm{Agg}_{R'}}] \big| R' \in \mathscr{P}(R)\big\}$, where

$$\begin{aligned} \big(\forall R' \in \mathscr{P}(R)\big)(\forall i, j \in \mathbb{R})\Big(i \leq^{\mathrm{Agg}(R')} j \iff \\ \big(\exists \tilde{R}_j \in \mathscr{P}_{max}(R'/\equiv)\big)\Big(g(\tilde{R}_j) = j \quad \wedge \\ \big(\forall \tilde{R}_i \in \mathscr{P}_{max}(R'/\equiv)\big)\big(g(\tilde{R}_i) = i \Rightarrow \tilde{R}_i \leq^{\mathscr{P}_{max}(R'/\equiv)} \tilde{R}_j\big)\Big)\Big), \end{aligned} \tag{3.4}$$

where

$$g : \mathscr{P}(R) \longrightarrow \mathbb{R}$$

is defined with respect to the specific aggregation function as:

Count: $[\mathscr{P}(R); \leq^R] \longrightarrow \big\{[|\widehat{R}|; \leq^{\mathrm{Count}_{R'}}] \big| R' \in \mathscr{P}(R)\big\}$

$$g : \big\{\mathscr{P}_{max}(R'/\!\equiv) \longrightarrow |\widehat{R}| \big| R' \in \mathscr{P}(R)\big\}$$

$$g(\tilde{R}) = \sum_{R_a \in \tilde{R}} |R_a| \tag{3.5}$$

In the following, A stands for attributes of the relation R.

Max: $[\mathscr{P}(R); \leq^R] \longrightarrow \Big\{ \big[(-\infty; \max\{r.A | r \in R\}\rangle; \leq^{\mathrm{Max}_{R'}}\big] \Big| R' \in \mathscr{P}(R)\Big\},$

$$g : \big\{\mathscr{P}_{max}(R'/\!\equiv) \longrightarrow (-\infty; \max\{r.A | r \in R\}\rangle \big| R' \in \mathscr{P}(R)\big\}$$

$$g(\tilde{R}) = \max\big\{r.A \big| \exists R_a \in R'/\!\equiv (r.A \in R_a \wedge R_a \in \tilde{R})\big\}$$

Min: $[\mathscr{P}(R); \leq^R] \longrightarrow \Big\{ \big[\langle \min\{r.A | r \in R\}; +\infty); \leq^{\mathrm{Max}_{R'}}\big] \Big| R' \in \mathscr{P}(R)\Big\},$

$$g : \big\{\mathscr{P}_{max}(R'/\!\equiv) \longrightarrow \langle \min\{r.A | r \in R\}; +\infty) \big| R' \in \mathscr{P}(R)\big\}$$

$$g(\tilde{R}) = \min\big\{r.A \big| \exists R_a \in R'/\!\equiv (r.A \in R_a \wedge R_a \in \tilde{R})\big\}$$

Sum: $[\mathscr{P}(R); \leq^R] \longrightarrow \big\{[\mathbb{R}; \leq^{\mathrm{Sum}_{R'}}] \big| R' \in \mathscr{P}(R)\big\},$

$$g : \big\{\mathscr{P}_{max}(R'/\!\equiv) \longrightarrow \mathbb{R} \big| R' \in \mathscr{P}(R)\big\}$$

$$g(\tilde{R}) = \sum_{\exists R_a \in R'/\equiv (r.A \in R_a \wedge R_a \in \tilde{R})} r.A,$$

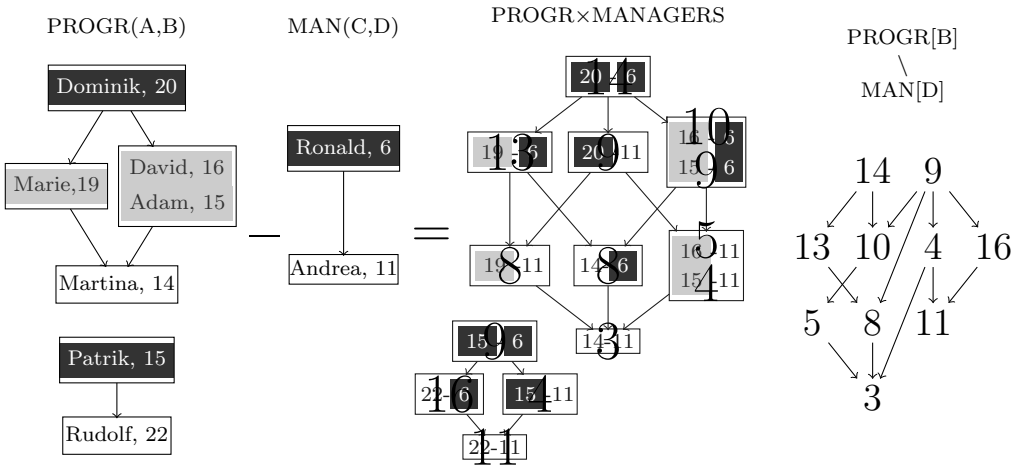
Average: $[\mathscr{P}(R); \leq^R] \longrightarrow \big\{[\mathbb{R}; \leq^{\mathrm{Avg}_{R'}}] \big| R' \in \mathscr{P}(R)\big\},$

$$g : \big\{\mathscr{P}_{max}(R'/\!\equiv) \longrightarrow \mathbb{R} \big| R' \in \mathscr{P}(R)\big\}$$

$$g(\tilde{R}) = \frac{\sum_{\exists R_a \in R'/\equiv (r.A \in R_a \wedge R_a \in \tilde{R})} r.A}{\sum_{R_a \in \tilde{R}} |R_a|},$$

3.3 Arithmetic

EXAMPLE 3.7 (Subtraction on ordered relations)
Let us consider two input relations PROGR(A,B) of programmers and managers MAN(C,D) respectively. We are interested in their names (attributes A,C) and years of practice (attributes B,D) only. The ordering reflects the preference based on their, say, proficiency. The question is: "What is the difference of years of practice between the most proficient programmers and managers?" We clearly need the arithmetic operation of subtraction.



To get the result, we have to consider all the possible couples of programmers and managers. The relation of these couples is ordered as cartesian product – see the equation (3.1). After performing the subtraction, the resulting ordering is determined in accordance with equation (3.6) – see below.

The semantic of this final determination can be seen on the couple of 9 and 8 for instance: For any couple of a programmer and a manager having the difference of years of practise 8, there is another couple of a programmer and a manager that is preferred to the former couple and whose difference of years of practice is 9. □

Let us consider a triplet $[R; \leq^R; \oplus]$ of a relation $R(A : \mathbb{R}, B : \mathbb{R})$, preference relation $\leq^R$ on R, and a binary arithmetic operation $\oplus^{(A,B)}$ on domains of attributes A and B:

$$\oplus^{(A,B)} : [R_1; \leq^{R_1}] \times [R_2; \leq^{R_2}] \to [\mathbb{R}; \leq^{R_1[A] \oplus R_2[B]}]$$

Then the resulting ordering $\leq^{R_1[A] \oplus R_2[B]}$ on $\mathbb{R}$ is defined as follows:

$$(\forall i, j \in \mathbb{R})\Big(i \leq^{R_1[A] \oplus R_2[B]} j \;\Leftrightarrow (\exists r_m \in R_1)(\exists r_n \in R_2)\Big(r_m.A \oplus r_n.B = j \quad \wedge$$
$$(\forall r_k \in R_1)(\forall r_l \in R_2)\big(r_k.A \oplus r_l.B = i \;\Rightarrow\; (r_k, r_l) \leq^{R \times R} (r_m, r_n)\big)\Big)\Big) \quad (3.6)$$

4 Conclusion

By redefinition of relational algebra operations, aggregation functions and arithmetic, we get operations of relational data model with preferences. These operations correspond to operations we have in the classical relational data model. Thus, on one side, we maintain the expressive power of the ordinary relational data model and at the same time, as the new operations operate on and return ordered relations, we are able to handle new information of preference represented by an ordering. This results in the ability to retrieve more accurate data.[2]

References

[1] C. J. Date, *An Introduction to Database Systems.* Pearson Education, 8th edition, 2004.

List of Symbols

$[a, b]$	an ordered pair of a and b
$R(\phi)$	a restriction of the relation R – the tuples satisfying a condition ϕ
$R[A]$	a projection of the relation R on the set of attributes A – subtuples of the relation R
$\leq^A$	an ordering relation with an index A (just a label)
$\leq_A$	a restriction of the ordering relation $\leq$ on the set A
$(\leq)^a$	a power a of the ordering relation $\leq$
$\equiv$	an equivalence relation
$R/\equiv$	$= \{R_a \mid R_a \subseteq R \wedge a \in R_a \wedge (\forall r \in R)(r \in R_a \Leftrightarrow r \equiv a)\}$
$\mathscr{P}(A)$	$= \{B \mid B \subseteq A\}$
$r.A$	the value that a tuple $r \in R$ acquires on an attribute A
$\oplus$	the general arithmetic operation $(+, -, \times, \div, \ldots)$
$\overline{A}$	transitive closure of the set A

Precedence	Operation	Symbol
higher	projection	$R[A]$
$\uparrow$	restriction	$R(\phi)$
$\uparrow$	product	$\times$
$\uparrow$	difference	$\setminus$
lower	union, intersection	$\cup$, $\cap$

TABLE 1. Precedence of relation algebra operations

Received 28 November 2004.

[2] To our best knowledge, there is no similar study described in the literature.