Doktorandský den ’04

Ústav informatiky

Akademie věd České republiky

Paseky nad Jizerou, 29. září – 1. říjen 2004

Obsah

Relational Databases with Ordered Relations

doktorand:
ING. RADIM NEDBAL
Ústav informatiky, Pod Vodárenskou věží 2, Praha 8
radned@seznam.cz

školitel:
ING. JÚLIUS ŠTULLER CSC.
Ústav informatiky, Pod Vodárenskou věží 2, Praha 8
stuller@cs.cas.cz

obor studia:
Mathematical Engineering
číselné označení: 39-10-9

Abstrakt

This paper[1] describes an option to express our preferences in the framework of relational databases. Preferences have usually a form of a partial ordering. Therefore the question is how to deliver the semantics of ordering to a database system. The answer is quite straightforward.

1. Introduction

When retrieving data, it is difficult for a user of a classical relational database to express various levels of preferences.

Example 1 (Preferences represented by an ordering) *How could we express our intention to find an employee with a good command of English, or at least a good command of German, or at worst a good command of Russian? At the same time, we may want the employee to belong to the salesmen department or with higher preference to the management department. To sum up, we have the following preferences:*

A language:	*B department:*
1. English,	*1. management department,*
2. German,	*2. salesman department,*
3. Russian,	

which can be formalized by an ordering, in general case by a partial ordering.

[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”

NAME	*LANGUAGE*	*DEPARTMENT*
Petr	*English*	*management*
Patrik	*German*	*management*
Pavel	*Russian*	*salesmen*
Dan	*Czech*	*clerk*
Robert	*English*	*president*
Martin	*German*	*management*
Marek	*Hungarian*	*clerk*

We can see that Petr is preferred to Pavel for instance. However, what we can say about Robert for example? In this case, we need cartesian product operating on ordered relations.

The aim of this paper is to incorporate semantics of partial ordering into all the primitive operations, i.e. those that can not be expressed by means of others operations, of relational data model. The resulting data model should be capable of providing users with the most, according to their preferences, relevant data.

2. Relational data model

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

As a relation is a set, we have all the **set operations** plus **aggregation functions**, which are unary operations on sets returning a number, plus **arithmetic** for performing all the usual operations on numbers. As for the relational data model, Codd has introduced eight relational algebra operations:

1. Cartesian product $\times$,
2. Union $\cup$,
3. Intersection $\cap$,
4. Difference $\setminus$,
5. Restriction,
6. Projection,
7. Join,
8. Divide

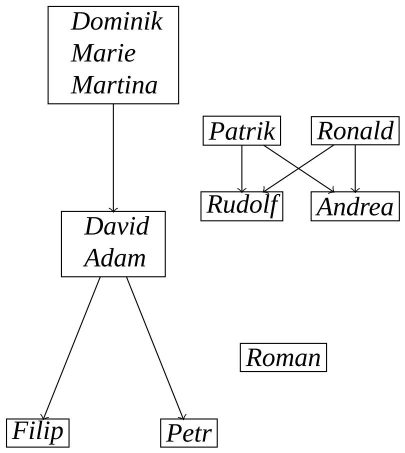
These operations are, however, not primitive [1] – they can be defined in terms of the others. In fact, of the set of eight, three (join, intersection and divide) can be defined in terms of the other five. Those other five operations (restriction, projection, cartesian product, union, and difference), by contrast, can be regarded as primitive, in the sense that none of them can be defined in terms of the other four. Thus, a minimal set of operations would be the set consisting of the five primitives – the *minimal set of relational algebra operations*.

3. Operations on Ordered Relations

Example 2 (Preferences of employees based on attribute values)

Relation scheme:
$R(\underline{NAME}, POSITION, LANGUAGE)$

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*



We prefer employees speaking English to those speaking German, and at the same time we prefer German speaking employees to those who speak other germanic language. Similarly, we prefer Spanish and French to any other romanic language. We have no other preference. [2]

The ordering represents an extra 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 $\leq^R$.

3.1. Relational algebra operations

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

In the case of 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 specified relation R after specified attributes have been eliminated.

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

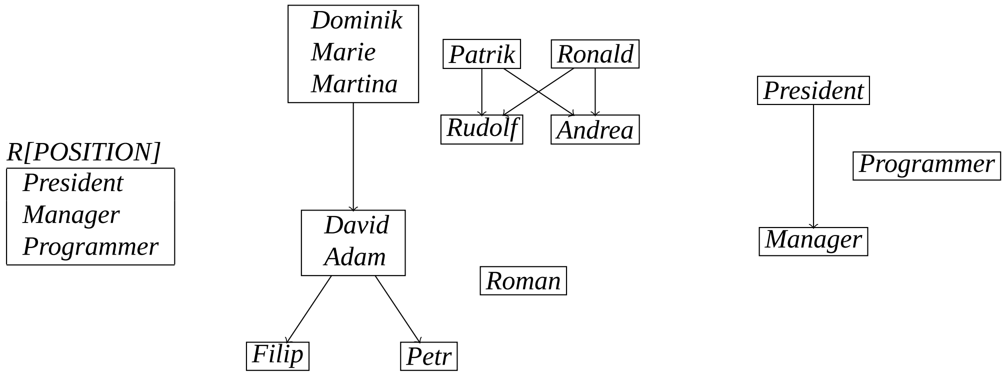
$$[R; \leq^R][C] = [R[C]; \leq^{R[C]}],$$

where

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

[2]The partial ordering is depicted using the standard Hasse diagram notation.

Example 3 (Ordering on a projection)



We prefer president to manager as all the presidents, which is in this case the only element, are preferred to all the managers in the input ordering. At the same time, we can say nothing about preferences of programmer and manager for instance as we can find incomparable couples of programmers and managers or those with contradictory preferences in the input relation.

Union $R_1 \cup R_2$ returns a relation consisting of all tuples appearing in either or both of two specified relations R_1, R_2.

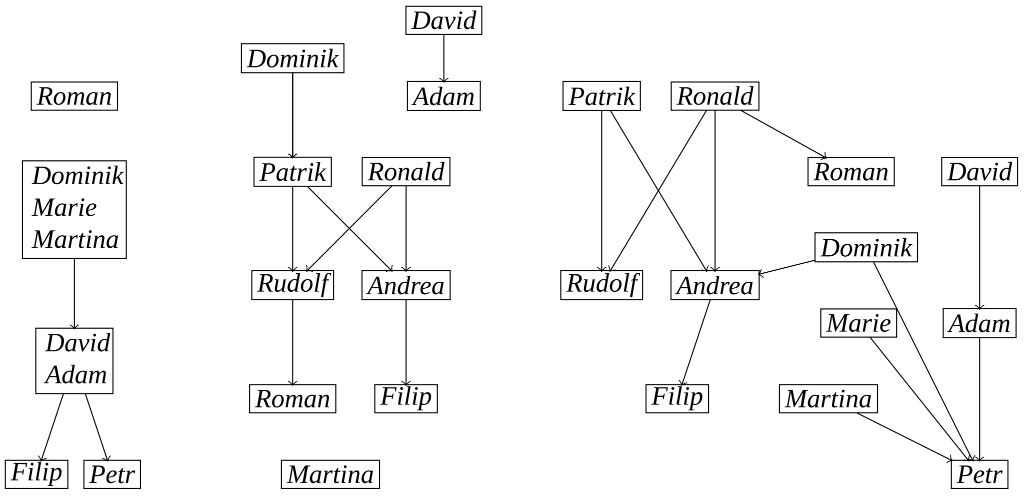
In the case of ordered relations $[R_1, \leq^{R_1}], [R_2, \leq^{R_2}]$, we define:

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

where

$$\begin{aligned}
&\forall r_1, r_2 \in (R_1 \cup R_2)(r_1 \leq^{R_1 \cup R_2} r_2 \iff \\
&\quad (r_1 \leq^{R_1} r_2 \quad \wedge \quad r_1 \leq^{R_2} r_2) \quad \vee \\
&\quad (r_1 \leq^{R_1} r_2 \quad \wedge \quad r_1, r_2 \notin R_2) \quad \vee \\
&\quad (r_1 \leq^{R_2} r_2 \quad \wedge \quad r_1, r_2 \notin R_1) \quad \vee \\
&\quad (\exists r_3 \in (R_1 \cup R_2)(r_1 \leq^{R_1} r_3 \quad \wedge \quad r_3 \leq^{R_2} r_2)) \quad \vee \\
&(r_1 \in R_1 \cap R_2 \quad \wedge \quad r_2 \in R_1 \setminus R_2 \quad \wedge \quad r_1 \leq^{R_1} r_2 \quad \wedge \quad \forall r_3 \in R_1 \cap R_2 (r_2 \leq^{R_1} r_3 \Rightarrow r_1 \leq^{R_2} r_3)) \quad \vee \\
&(r_2 \in R_1 \cap R_2 \quad \wedge \quad r_1 \in R_1 \setminus R_2 \quad \wedge \quad r_1 \leq^{R_1} r_2 \quad \wedge \quad \forall r_3 \in R_1 \cap R_2 (r_3 \leq^{R_1} r_1 \Rightarrow r_3 \leq^{R_2} r_2)) \quad \vee \\
&(r_1 \in R_1 \cap R_2 \quad \wedge \quad r_2 \in R_2 \setminus R_1 \quad \wedge \quad r_1 \leq^{R_2} r_2 \quad \wedge \quad \forall r_3 \in R_1 \cap R_2 (r_2 \leq^{R_2} r_3 \Rightarrow r_1 \leq^{R_1} r_3)) \quad \vee \\
&(r_2 \in R_1 \cap R_2 \quad \wedge \quad r_1 \in R_2 \setminus R_1 \quad \wedge \quad r_1 \leq^{R_2} r_2 \quad \wedge \quad \forall r_3 \in R_1 \cap R_2 (r_3 \leq^{R_2} r_1 \Rightarrow r_3 \leq^{R_1} r_2)))
\end{aligned}$$

Example 4 (Ordering on a union)



We can determine easily the ordering of the elements belonging to the intersection of the input relations and of the elements belonging to the symmetric difference of the input relations. Then we have to determine the ordering between elements from intersection and symmetric difference of the input relations. The possible contradictions following from the transitivity property of ordering have to be avoided.

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

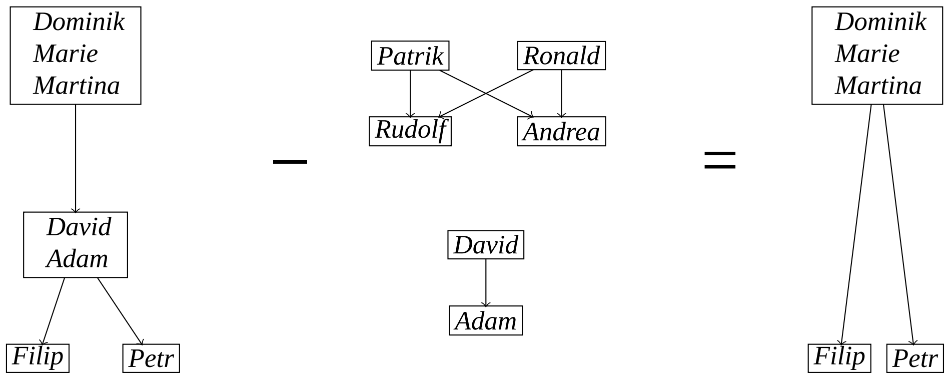
In the case of ordered relations $[R, \leq^{R_1}], [R_2, \leq^{R_2}]$, we define:

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

where

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

Example 5 (Ordering on a difference)



The difference ordering is the restriction of the input ordering on the the difference of the input relations.

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 two specified relations R_1, R_2.

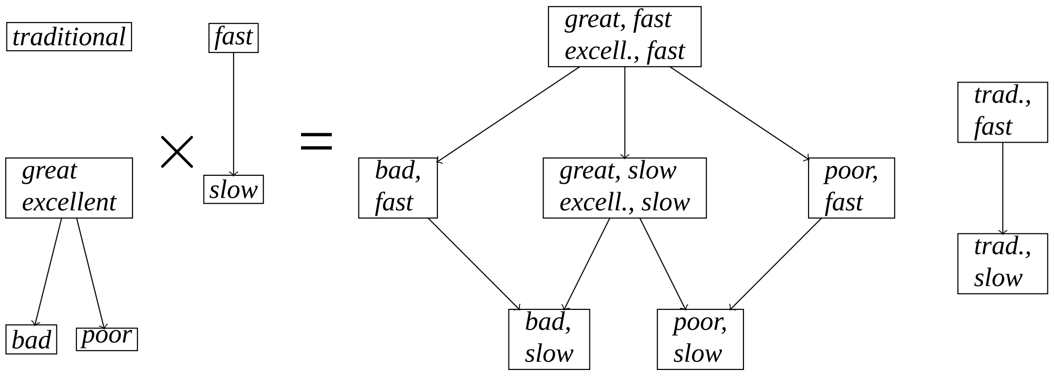
In the case of ordered relations $[R, \leq^{R_1}], [R_2, \leq^{R_2}]$, we define:

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

where

$$\leq^{R_1 \times R_2} = \{((r_1, r_2), (r'_1, r'_2)) | (r_1, r'_1) \in \leq^{R_1} \wedge (r_2, r'_2) \in \leq^{R_2}\}$$

Example 6 (Ordering on a cartesian product)



The output ordering is defined as a ordering of ordered pairs.

3.2. Aggregation Functions

In this subsection, we will extend the database relation R with a special element

$$\hat{r} = R(\phi), \quad \text{where} \quad \forall r \in R\ (\neg\phi(r))$$

In the following, the symbol R stands for this extended relation.

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

$$\leq^R \text{ is generally no ordering on } R \text{ because } \leq^R \cap (\leq^R)^{-1} \not\subseteq I = (\leq^R)^0,$$

$$\leq^R \cap (\leq^R)^{-1} \subseteq R \times R,$$

$$[R; \equiv], \qquad \equiv = \leq^R \cap (\leq^R)^{-1}, \qquad \equiv \text{ is a relation of equivalence}$$

$$[R/\!\equiv; \leq^{R/\equiv}], \qquad \forall R_a, R_b \in R/\!\equiv (R_a \leq^{R/\equiv} R_b \iff a \leq^R b),$$

where

$$a, b \in R, \quad R_a = \{r \in R | r \equiv a\}, \quad R_b = \{r \in R | r \equiv b\}$$

$$\leq^{R/\equiv} \text{ is an ordering on } R/\equiv$$

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

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

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

An aggregation function g in the classical relational data model is a function: $\mathscr{P}(R) \rightarrow \mathbb{R}$ operating on sets and returning numbers. In the case of relational databases with ordered relations, we define it as:

$$g: \quad \mathscr{P}([R; \leq^R]) \rightarrow [\mathbb{R}; \leq^{g(R)}],$$

where

$$\mathscr{P}([R; \leq^R]) = \{[R'; \leq^{R'}] | R' \subseteq R \wedge \leq^{R'} = \leq^R_{R'}\}$$

and

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

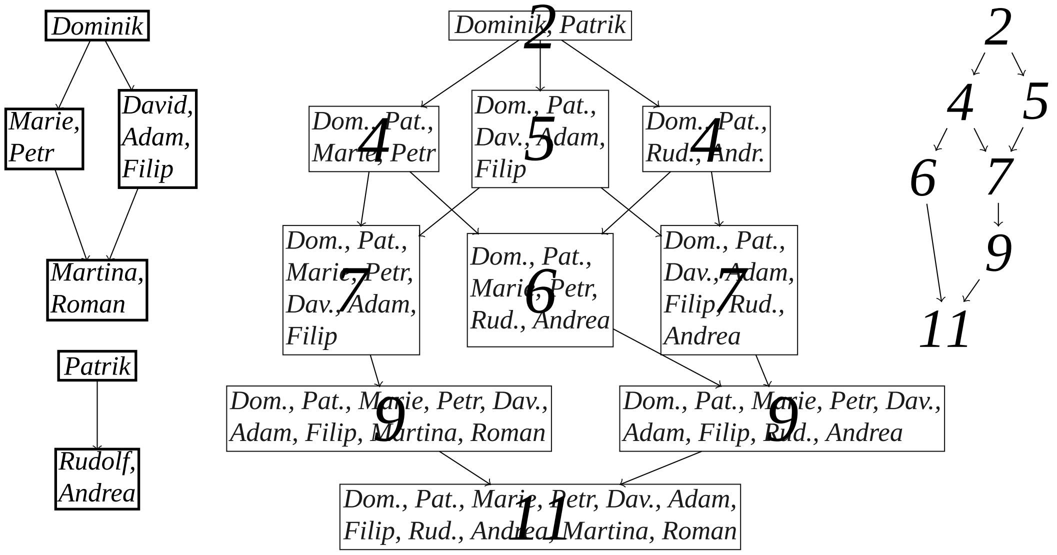
where g is defined with respect to the specific aggregation function as:

Count

$$[|R|; \leq^{|R|}]$$

$$g: \mathscr{P}_{max}(R/\equiv) \rightarrow |R|, \quad g(\tilde{R}) = \sum_{R_a \in \tilde{R}} |R_a|$$

Example 7 (Count on an ordered relation)


Dominik
Marie, Petr
David, Adam, Filip
Martina, Roman
Patrik
Rudolf, Andrea
Dominik, Patrik
2
Dom., Pat., Marie, Petr
4
Dom., Pat., Dav., Adam, Filip
5
Dom., Pat., Rud., Andr.
4
Dom., Pat., Marie, Petr, Dav., Adam, Filip
7
Dom., Pat., Marie, Petr, Rud., Andrea
6
Dom., Pat., Dav., Adam, Filip, Rud., Andrea
7
Dom., Pat., Marie, Petr, Dav., Adam, Filip, Martina, Roman
9
Dom., Pat., Marie, Petr, Dav., Adam, Filip, Rud., Andrea
9
Dom., Pat., Marie, Petr, Dav., Adam, Filip, Rud., Andrea, Martina, Roman
11
2
4
5
6
7
9
11

First we count the most preferred elements. Then the the less preferred elements are added. The rule is that we never add the elements that are in the hierarchy of the input ordering below the elements that we have not counted yet. The rationale behind this rule 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 certain level. Then the classical count operation is applied and finally the resulting ordering determined.

The semantic of this final determination can be seen on the couple of 4 and 7 for instance: For any set of 7 elements having been 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.

Max

$$[(-\infty; \max\{r.A|r \in R\}\rangle; \leq^{(-\infty;\max\{r.A|r\in R\}\rangle}]$$

$$g : \mathscr{P}_{max}(R/\!\!\equiv) \to (-\infty; \max\{r.A|r \in R\}\rangle, \quad g(\tilde{R}) = \max\{r.A|\exists R_a \in R/\!\!\equiv (r.A \in R_a \wedge R_a \in \tilde{R})\}$$

Min

$$[\langle\min\{r.A|r \in R\}; \infty); \leq^{\langle\min\{r.A|r\in R\};\infty)}]$$

$$g : \mathscr{P}_{max}(R/\!\!\equiv) \to \langle\min\{r.A|r \in R\}; \infty), \quad g(\tilde{R}) = \min\{r.A|\exists R_a \in R/\!\!\equiv (r.A \in R_a \wedge R_a \in \tilde{R})\}$$

Sum

$$[\mathbb{R}; \leq^{\text{Sum}(R)}],$$

$$g : \mathscr{P}_{max}(R/\!\!\equiv) \to \mathbb{R}, \quad g(\tilde{R}) = \sum_{\exists R_a\in R/\equiv(r.A\in R_a\wedge R_a\in\tilde{R})} r.A$$

Average

$$[\mathbb{R}; \leq^{\text{Avg}(R)}],$$

$$g : \mathscr{P}_{max}(R/\!\!\equiv) \to \mathbb{R}, \quad g(\tilde{R}) = \frac{\sum_{\exists R_a\in R/\equiv(r.A\in R_a\wedge R_a\in\tilde{R})} r.A}{\sum_{R_a\in\tilde{R}} |R_a|}$$

3.3. Arithmetic

We will consider a triplet of a relation R with a preference relation $\leq^R$ and basic arithmetic operations – denoted $\oplus$:

$$[R; \leq^R; \oplus]$$

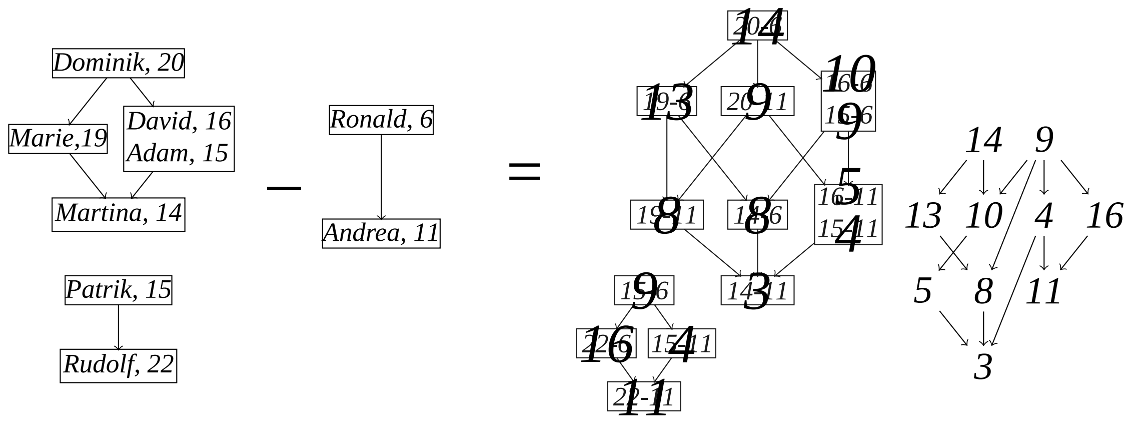
$$\oplus : [R_1; \leq^{R_1}][A] \times [R_2; \leq^{R_2}][B] \to [\mathbb{R}; \leq^{R_1[A]\oplus R_2[B]}]$$

where

$$\forall i, j \in \mathbb{R}\Big(i \leq^{R_1[A]\oplus R_2[B]} j \;\Leftrightarrow$$

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

Example 8 (Subtraction on an ordered relation) *Let's consider two input relations of programmers and managers respectively. We are interested in their names and years of practice 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.*



We have to consider all the possible couples of programmers and managers. The relation of these couples is ordered as ordered pairs of numbers. After performing the subtraction, the resulting ordering is determined.

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 above this couple in the hierarchy of preference and whose difference of years of practice is 9.

4. Conclusion

By means of redefinition of the *minimal set of relational algebra operations*, aggregation functions and arithmetic, we get operations corresponding to all the operations that we have in the relational database framework. Thus we maintain the expressive power of the classical relational model. As the new operations operate on and return ordered relations, we are able to handle an extra information of preference represented by an ordering. The result is the ability to retrieve more accurate data.

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
$\leq^A_B$	a restriction of the ordering relation $\leq^A$ on the set B
$(\leq)^a$	a power a of the ordering relation $\leq$
$\equiv$	an equivalence relation
$R/\equiv$	$= \{R_a \vert R_a \subseteq R \wedge a \in R_a \wedge \forall r \in R(r \in R_a \Leftrightarrow r \equiv a)\}$
$\mathscr{P}(A)$	$= \{B \vert B \subseteq A\}$
$r.A$	the value that a tuple $r \in R$ acquires on an attribute A
$\oplus$	the general arithmetic operation $(+, -, \times, \div, \ldots)$

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

Tabulka 1: Precedence of relation algebra operations

References

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