Jaroslav Janák

The calculus of an unnormalized relational model of data

Kybernetika, Vol. 20 (1984), No. 3, 231--239

Persistent URL: http://dml.cz/dmlcz/125582

KYBERNETIKA — VOLUME 20 (1984), NUMBER 3

THE CALCULUS OF AN UNNORMALIZED RELATIONAL MODEL OF DATA

JAROSLAV JANÁK

Recently, much attention has been paid to the semantic extension of the relation data model. A contribution factor to this field is the calculus that can be interpreted on a semantically extended relational model of data. This calculus allows the objects of relation and attribute to be used in a more general sense that it has been the case in the calculuses reported so far in the literature.

0. INTRODUCTION

The relational model of data was first defined by E. F. Codd [2] in the early seventies. In the years following, this model was given considerable theoretical [5] and also practical attention [1, 6]. The theoretical elaboration of the model essentially aimed at the definition and the description of properties that are not explicitly contained in the relational model, but that follow from relationally organized data (various types of interdependencies of data).

Already in its very beginnings, the relational model of data was commonly restricted to relations in the so-called first normal form. The domain of the attribute was demanded not to consist of relations. As far as the creations of the relations were concerned, this requirement presented no serious problems. It has always been practical and also advantageous to create only normalized relations.

However, the first normal form of the relations brings about restrictions in the communication with the relational model of data. Since, in this case, the relation cannot be an element of the attribute domain of an other relation, it cannot appear at the place of the argument of the predicate in the language of predicate calculus, nor can it be the value of the function in this language. For this reason, it is not possible, for instance, in the language of the predicate calculus, to identify a relation with the aid of the attribute of this relation or to express a function that assigns the relation itself to an element of the domain of the relation attribute, etc.

Other restrictions in communicating with the relational model of data are due

to the position of the attribute in the relational model of data, be it normalized or unnormalized. In the relational model of data, the attribute is thought to be an objects distinct from other objects.

This distinctness leads to two complications. Since the attribute cannot be an element of an other attribute domain of any relation, it does not enable us, in the relational model of data, to define the relation describing the properties of the attributes or to write, in the language of the predicate calculus, a predicate having the attribute as its argument (the corresponding predicate symbol would lack an interpretation in the relational model). Similarly as in the case of the relation, thus, the attribute cannot be identified, in the predicate calculus language, with the aid of attributes of its own and it is not possible to express the function assigning the attribute itself to an element of the attribute domain, etc.

The domain-oriented relational calculus [8, 9] and the n-tuple oriented relational calculus [4, 9] are examples of such types of calculuses. The domain-oriented relational calculus employs the attribute in so-called model predicates, while the n-tuple oriented relational calculus does so in what we call n-tuple variables. However, in both calculuses the attribute is not expressed by an independent symbol type and both calculuses do not allow the attribute to be used in its full universality. For instance, they do not allow the attribute to be expressed as a variable in the formulas.

This article describes the language of a domain-oriented calculus in which the symbol of the relation or that of the attribute can make up the term in the same way as any other symbol interpreted by an element of the attribute domain (Section 3). The rules governing the syntactic and the semantic definitions of this language respect the rules of the syntactic and semantic definitions of the language of the predicate calculus of the order k. However, they have been extended to the rules defining the attribute object. The interpretation structure consists of an un-normalized relational model of data where the elements of the domain can also be attributes of the relations.

The basic concepts of the relational model of data are defined in Section 1. The initial predicate language of the order k is described in Section 2, while Section 4 demonstrates the potential application of the language to a semantically extended un-normalized relational data base.

1. DEFINITION OF THE RELATIONAL MODEL OF DATA

At the present time, the relational model of data is a theoretically highly developed field in data base design. In the paper presented, the definitions of the basic concepts start from the literature [2, 5, 10].

The relation concept is the basic concept of the relational model of data.

Definition. Let $\boldsymbol{A} = \{A_1, \ldots, A_n\}$ be the non-empty set of attributes. Let dom (A_i)

be the set of values, the so-called attribute domain of the attribute A_i, and let it be uniquely assigned to every attribute $A_i \in \boldsymbol{A}$. Then, the structure defined by:

– a set of attributes $\boldsymbol{A}$,

– a subset of cartesian product $\mathrm{dom}(A_1) \times \ldots \times \mathrm{dom}(A_n)$

we shall call the *relation* of degree n over the set of attributes $\boldsymbol{A}$. Such a structure we shall denote $\mathbf{R}(A_1, \ldots, A_n)$ or shorter $\mathbf{R}$.

It follows from the definition that the attributes of a relation are unique, that means that $A_i \neq A_j$ holds for $i \neq j$.

The attribute domain can also be a set the elements of which are relations. Such a domain is called a non-simple domain. A non-simple domain can comprise relations defined on other non-simple domains, etc. The relation is considered to be normalized or of the first normal form, if it is the subset of a cartesian product of simple domains only. We also admit that the element of the attribute domain may be attributes of the relations.

The set of all possible relations is called the *relational model of data*. If the relational data model comprises only all normalized relations it is considered to be a normalized relational model of data. We shall call the subset of the relational model the relational data base.

2. THE PREDICATE CALCULUS OF HIGHER ORDER

The language basing upon the predicate calculus of first order [4, 8] can be interpreted on the normalized relational model of data. However, when admitting unnormalized relations, we should start from the predicate calculus of higher order in defining the predicate language.

The types and their ordering

Let $\boldsymbol{T}$ be the set with the following properties:

1. $t_0^1, t_0^2, \ldots \in \boldsymbol{T}$,
2. if $t_1, \ldots, t_n \in \boldsymbol{T}$, then the ordered n-tuple $(t_1, \ldots, t_n) \in \boldsymbol{T}$,
3. the element of the set $\boldsymbol{T}$ are only elements determined by rules 1 and 2.

In this case, the set $\boldsymbol{T}$ is called the set of types and its elements are called *types*.

The elements $t_0^1, t_0^2, \ldots$ form the base of the set of types. Later, the type t_0 will denote any element of the base. As follows immediately from the definition, every type $t \neq t_0$ (i.e. $t \neq t_0^1, t \neq t_0^2, \ldots$) uniquely determines its constituting ordered n-tuple $t = (t_1, \ldots, t_n)$.

A number called the *rank of the type* can be assigned to every type. Let us put number 0 as the rank of the type t_0. The rank of the type $t \neq t_0$ is represented by the

natural number k which is the length of the longest sequence of types $t_1, \ldots, t_k$ with the following properties:

1. $t_k = t$, $t_1 \neq t_0$,
2. for $1 \leqq i < k$, t_i is one of the elements of the ordered n-tuple forming the type t_{i+1}.

The order of types determines the map r from the set of types to the set of integers. This map defines, on the set $\boldsymbol{T} \times \boldsymbol{T}$, the binary relation denoted $\leqq^T (t_1 \leqq^T t_2 \Leftrightarrow$ $\Leftrightarrow r(t_1) \leqq r(t_2))$. The relation $\leqq^T$ is obviously reflexive, antisymmetric, and transitive and, therefore, it is the total ordering of the set of types $\boldsymbol{T}$.

The syntax of the language of the predicate calculus of order k

The language of the predicate calculus is determined by the set of the basic symbols and the elements of the language are called formulas. A formula is defined by way of induction using the auxiliary concepts terms and the atomic formula (n-ary predicate).

We shall denote $\boldsymbol{L}_k$ the language of the predicate calculus of order k. Every basic symbol of the language $\boldsymbol{L}_k$ has a type assigned to it. We shall admit several types to be assigned to the predicate and functional symbols. The basic symbols of the language $\boldsymbol{L}_k$ involve the following kinds of symbols:

1. constants $A, a, A_1, a_1, \ldots$ of the type t_0,
2. constants $\mathbf{P}, \mathbf{P}_1, \ldots$ of the types t, where $1 \leqq r(t) \leqq k$,
3. variables $x, x_1, \ldots$ of the types t, where $0 \leqq r(t) \leqq k$,
4. n-ary functional constants $\mathbf{F}, \mathbf{F}_1, \ldots$ of the types $(t, t_1, \ldots, t_n)$ where $0 \leqq r(t)$, $r(t_1), \ldots, r(t_n) \leqq k$, $n \geqq 1$. The value of the interpreting function is of the type t, the n of the arguments is of the types $t_1, \ldots, t_n$.
5. logical joints $\neg \wedge \vee \Rightarrow \Leftrightarrow$
   quantifiers $\forall \exists$
   delimiters , ( )

The sets of all kinds and types of the basic symbols of the language $\boldsymbol{L}_k$ should be pairwise disjoint. They can be empty or infinite. The variable or constants of the type t_0 will be called individual variables or individual constants. The variables or constants of the type $(t_1, \ldots, t_n)$ will be called n-ary predicate variables or n-ary predicate constants. The unary predicate symbols are called sets.

The set of the terms of the language $\boldsymbol{L}_k$ is divided into sets of terms of various types. The terms of the type t $(0 \leqq r(t) \leqq k)$ consist of:

1. constants of the type t and variables of the type t,
2. the expression of the form $\mathbf{F}(T_1, \ldots, T_n)$, where $T_1, \ldots, T_n$ are terms of the types $t_1, \ldots, t_n$ and $\mathbf{F}$ is the n-ary functional constant of the type $(t, t_1, \ldots, t_n)$.

The atomic formula (or the n-ary predicate) of the language $\boldsymbol{L}_k$ is an expression of one of the following forms:

1. $\mathbf{P}(T_1, \ldots, T_n)$ where $T_1, \ldots, T_n$ are terms of the types $t_1, \ldots, t_n$ and $\mathbf{P}$ is the n-ary predicate constant of the type $(t_1, \ldots, t_n)$,
2. $x(T_1, \ldots, T_n)$, where $T_1, \ldots, T_n$ are terms of the types $t_1, \ldots, t_n$ and x is the n-ary predicate variable of the type $(t_1, \ldots, t_n)$.

The formulas of the language $\boldsymbol{L}_k$ are defined with the aid of atomic formulas. The latter are constituted by:

1. any atomic formula,
2. the expressions $\neg\mathscr{F}, (\mathscr{F} \wedge \mathscr{F}_1 \wedge \ldots \wedge \mathscr{F}_n), (\mathscr{F} \vee \mathscr{F}_1 \vee \ldots \vee \mathscr{F}_n), (\mathscr{F} \Rightarrow \mathscr{F}_1)$, $(\mathscr{F} \Leftrightarrow \mathscr{F}_1)$ where $\mathscr{F}, \mathscr{F}_1, \ldots, \mathscr{F}_n$ are formulas,
3. the expressions $\forall x\mathscr{F}, \exists x\mathscr{F}$ where $\mathscr{F}$ is the formula and x is a variable.

The semantics of the language of the predicate calculus of the order k

The semantic of the language of the predicate calculus imply the interpretation of all kinds and types of constants and variables in a certain structure, the interpretation of all functional symbols in this structure and the unique assignment of values from the set $\{TRUE, FALSE\}$ to all formulas of the language.

The interpretation structure $\boldsymbol{S}_k$ of the language $\boldsymbol{L}_k$ is composed of:

1. non-empty, pairwise disjoint sets $U^{t_1}, U^{t_2}, \ldots$, the so-called interpretation domains of the types $t_1, t_2, \ldots$,
2. n-ary functions mapping $U^{t_1} \times \ldots \times U^{t_n}$ to U^t for every functional constant of the type $(t, t_1, \ldots, t_n)$.

The functional constant of the language $\boldsymbol{L}_k$ is interpreted, on the structure $\boldsymbol{S}_k$, by the corresponding function of the structure $\boldsymbol{S}_k$. The predicate constant of the type $t(1 \leqq r(t) \leqq k)$ is interpreted as an element of the interpretation domain U^t of the structure $\boldsymbol{S}_k$, the so-called relation, and the individual constant is interpreted as an element from the interpretation domain U^{t_0}. The variables of the type t $(0 \leqq r(t) \leqq k)$ are interpreted by the map h from the set of the variables of the type t to the interpretation domain of type t, that means again by relations or elements of the interpretation domain U^{t_0}.

The interpretation of terms given by the map h makes possible, in the ordinary way, to assign a value to every formula (with a certain interpretation of the variables h). For instance, the value $TRUE$ is assigned to the atomic formula $\mathbf{P}(T_1, \ldots, T_n)$ if the ordered n-tuple of the interpretations of the terms $(I(T_1), \ldots, I(T_n))$ is an element of the relation interpreting the predicate symbol $\mathbf{P}$. In the opposite case, the value $FALSE$ is assigned to the formula $\mathbf{P}(T_1, \ldots, T_n)$. Thus, the value assigned to the formula depends not only on the formula itself, but also on the interpretation of the terms and, therefore, it can also depend on the map h of the variables of this formula. A detailed description of how to assign values to formulas is given in the literature (see [7, 9], for instance).

Both the interpretation of predicate constants and that of predicate variables given by the map h should comply with the axiom of extensionality. According

to this axiom, the interpretation of the terms of the atomic formula is determined by means of the interpretation of the predicate symbol in this formula. The axiom of extensionality is applied with the aid of the predicates ω of the type $((t_1, \ldots, t_n), t_1, \ldots, t_n)$, where $(t_1, \ldots, t_n)$ is the type of the predicate symbol in the atomic formula and $t_1, \ldots, t_n$ are the types of the terms in this formula. Any interpretation of the constants and variables should assign the *TRUE* value to the corresponding predicates ω.

3. THE CALCULUS OF THE UNNORMALIZED RELATIONAL MODEL OF DATA

The interpretation of the language $\boldsymbol{L}_k$ on an unnormalized relational model of data requires to restrict the rules of the inductive formula definitions. We shall call this restructed language the language of the calculus of the unnormalized relational data model and denote it $\boldsymbol{L}_R$. The intended interpretation of the individual variables and constants on the attribute domains can characterize this calculus as being domain-oriented.

The definition of the calculus of the unnormalized relational data model does not comprise n-ary functional constants. We can completely take over the other rules defining the predicate calculus of the order k but permitted only for the following types:

1. the type of the elements of the simple domains t_0,
2. the type of the relations $t_R = (t_X, \ldots, t_X)$, where $t_X \in \{t_0, t_R\}$.

The following interpretation domains correspond to the types of the terms of this calculus in the relational model of data:

1. the interpretation domain U^{t_0} of the type t_0 which is the union of simple attribute domains $\bigcup_i \bigcup_{j=1}^{n_i} \mathrm{dom}(A_{ij})$ where the index i passes through all relations of the relational data model and where n_i is the degree of the relation $\mathbf{R}_i$,
2. the interpretation domain U^{t_R} of the type t_R which consists of the relations of the relational data model.

In the language of the predicate calculus, the location of the term in the n-ary predicate is conditioned by its position j $(1 \leqq j \leqq n)$. In query languages basing on the predicate calculus of the first order (cf. [8], for instance), the term position in the predicate can be expressed by means of the attribute in accordance with certain rules. For instance, the n-ary predicate $\mathbf{P}(T_1, x, T_2, x, \ldots, x)$ interpreted on a relation whose first attribute is A and the third attribute is B, can be written as $\mathbf{P}(A : T_1, B : T_2)$ according to these rules.

In the calculus of the un-normalized relational model of data, however, the attribute

can appear as a more general object. The following re-arrangements will be sufficient:

1. adding the attribute type t_0^A,
2. re-defining the relational type as $t_R = (t_0^A : t_X, \ldots, t_0^A : t_X)$, where $t_X \in \{t_0, t_0^A, t_R\}$.

All attributes of the relational data model make up the interpretation domain U^A of the type t_0^A. The syntax and the semantics of the language L_R remain fully valid, but the extensionality axiom has to be applied when interpreting. The interpretation of the terms (of the type t_X) should be determined by the interpretation of the attributes (of the type t_0^A), and the interpretation of the attribute should be determined by the interpretation of the predicate symbol.

Conventions simplifying the writing of atomic formulas may be accepted. When writing a formula, the position of the pair $t_0^A : t_X$ within the predicate is of no importance. The validity of the extensionality axiom and the uniqueness of the attribute in the relation allows this position to be determined during the interpretation only. The axiom of extensionality allows to accept another convention to shorten the writting. Due to this convention it is not necessary to write all free variables. Missing free variables can automatically be added during the interpretation. If, for instance, the n-ary predicate symbol **P** is interpreted as the n-ary relation $\mathbf{R}(A_1, \ldots, A_n)$, then the following notations of the atomic formula are equivalent:

$$\mathbf{P}(\ldots, A_i : a_i, A_j : a_j, \ldots) \equiv \mathbf{P}(\ldots, A_i : a_i, A_{i+1}^* : x_{i+1}^*, \ldots, A_{j-1}^* : x_{j-1}^*, A_j : a_j, \ldots)$$

where $i < j$ and the asterisk symbols can be added during the interpretation.

4. SEMANTIC EXTENSION AND QUERIES ON THE RELATION DATA BASE

The possible semantic extension of the relational data base is going to be exemplified for the catalogue relations taken over in free form from the model RM/T [3]. Let us assume that, in addition to other relation, there exist relations defined on relations, attributes and domains:

CATR (*R*, *RELRANGE*)
CATA (*A*, *R*, *USERKEY*)
CATD (*D*, *VTYPE*)

The data base relations are the domain of the attribute *R*, the ranges concerned with the relations are the domain of the attribute *RELRANGE*, the data base attributes are the domain of the attribute *A*, the values *YES*|*NO* are the domain of the attribute *USERKEY* depending upon whether the attribute is or is not a keyword attribute, the attribute domains are the domain of the attribute *D*, and the domain types are the domain of the attribute *VTYPE*.

The relations interpreting the predicates ensuring the validity of the extensionality axiom have to be defined on every relational data base. In the given case, let this be all relations determining the reference between a relation and its attributes, and the relation

CATRAD (R, A, D)
determining the reference between an attribute and its attribute domain.

In order to simplify the queries, the binary relations $=$, $<$, $\geqq$, $>$, $\leqq$, $\neq$ defined on the cartesian product $U^X \times U^X$ where $U^X \in \{U^{t_0}, U^{t_0^A}, U^{t_R}\}$, are commonly defined in the relational data base.

In the language of the predicate calculus, the query can be of the form $\mathscr{F}(x_1, \ldots, x_n)$, where $\mathscr{F}(x_1, \ldots, x_n)$ is the formula with the free variables $x_1, \ldots, x_n$. The response to this query is given by the set of all ordered n-tuples of the interpretations $(I(x_1), \ldots$ $\ldots, I(x_n))$ for which the formula $\mathscr{F}(x_1, \ldots, x_n)$ is true (in the case of existentially bound variables the response can also comprise the interpretation of these variables). If the formula $\mathscr{F}(x_1, \ldots, x_n)$ lacks free variables, that menas $n = 0$, then the response is of the *YES*|*NO* type.

The relational data base described above aids in illustrating the application of the language $\boldsymbol{L}_R$. Instead of the constants of this language, we shall directly write their interpretation in the formulas of the query. The potentials of the language $\boldsymbol{L}_R$ can be demonstrated, for instance, for the query that would read as follows in the natural language: "Does any key attribute of the relation in the *PERSONALITY* range contain a key the value of which is a? If yes, write all the data concerning this key!" This query is expressed as follows in the language $\boldsymbol{L}_R$:

$$(\textbf{CATR}\ (R : x_R,\ RELRANGE : PERSONALITY) \wedge$$
$$\textbf{CATA}\ (A : x_A,\ R : x_R,\ USERKEY : YES) \wedge x_R(x_A : a))$$

or shorter:

$$(\textbf{CATR}\ (: x_R,\ RELRANGE : PERSONALITY) \wedge$$
$$\textbf{CATA}\ (: x_A, : x_R,\ USERKEY : YES) \wedge x_R(x_A : a))$$

or even shorter:

$$((: x_R,\ RELRANGE : PERSONALITY) \wedge (: x_A, : x_R,\ USERKEY : YES) \wedge x_R(x_A : a))$$

A further reduction in writing the query could substantially increase the possibility of an unexpected interpretation of the query.

5. CONCLUSION

The article describes the language $\boldsymbol{L}_R$ of the calculus of the un-normalized relational model of data. The contribution of this language consists in the more general concept of the objects of the structure of the relational data base. Every relation or attribute can be appear as a variable in this language, it can be quantified or even qualified by other properties of its own. This is an extension if compared to the calculuses known from the literature [4, 8] where these objects are denoted only by their names.

The predicate calculus of the order $k > 1$ is the theoretical starting point of the calculus of the un-normalized relational data model. Contrary to the predicate calculus of the first order, the predicate calculuses of higher order have not yet been

paid systematic attention. Therefore, the basic rules of such a calculus had to be defined in the paper presented. As compared to the definition reported in the literature [7], this paper defines other extending concepts and rules. The base of types was introduced, the assignment of several types to certain symbols was permitted, and the conception of the function was generalized.

The calculus of the un-normalized relational model of data fully respect the rules of the predicate calculus of the order $k > 1$. It was found necessary, however, to extend it to the rules defining a new type – the attribute type. In this calculus, the attribute type is comprehended in a more general way than it does in the calculuses known from the literature [4, 8].

The formulas of the language $\boldsymbol{L}_R$ are interpreted on an un-normalized relational model of data. The structure of the un-normalized relational model of data was fully suited for this purpose, it was only necessary to permit the attributes to be elements of the attribute domain.

The language $\boldsymbol{L}_R$ is a formal language suitable to the interpretation on a semantically extended relational data base. Thus, it also is a contribution to the topical problems associated with the relational model that have also been investigated by the Research Institute of Mathematical Machines.

(Received May 18, 1983.)

REFERENCES

[1] D. D. Chamberlin, A. M. Gilbert and R. A. Yost: A history of System R and SQL/data system. In: Proceedings of Very Large Data-Bases Conf., IEEE Computer Society Press 1981, pp. 70–94.

[2] E. F. Codd: A relational model of data for very large shared data banks. Comm. ACM *13* (1970), 6, 377–387.

[3] E. F. Codd: Extending the database relational model to capture more meaning. ACM Trans. Database Systems *4* (1979), 4, 397–434.

[4] E. F. Codd: Relational completeness of database relational model. In: Courant Computer Science Symposium 6 in Data Base Systems. Prentice Hall, Englewood Cliffs 1972, pp. 65–98.

[5] C. Delobel: An overview of the relational data theory. In: Information Processing, North-Holland, Amsterdam 1980, pp. 413–426.

[6] J. Janák: Relační systém řízení báze dat – Systém R (Relational Data Base Management Systém – System R). Informačné systémy *11* (1982), 4, 345–358.

[7] G. Kreisel and J. L. Krivine: Elements of Mathematical Logic. North Holland, Amsterdam 1967.

[8] A. Pirotte: Explicit Description of Entities and their Manipulation in Languages for the Relational Data Base Model. Report R 336, M. B. L. E. 1976.

[9] J. Pokorný: Dotazy a odpovědi v databázových systémech (Queries and Responses in Database Systems). ČVUT, Praha 1980.

[10] J. Pokorný and S. Machová: Databázové modely (Database models). Výpočetní centrum Univerzity Karlovy, Praha 1982.

RNDr. Jaroslav Janák, Výzkumný ústav matematických strojů, koncernová účelová organizace (Research Institute of Mathematical Machines), Loretánské nám. 3, 100 000 Praha 1. Czechoslovakia.