

Data Functional Feature Sets and their Adaptability

Petras G. Adomenas

Vilnius Gediminas Technical University, Lithuania
adomenas@fa.vtu.lt

Abstract. This paper presents a data functional feature set whose subsets may be applied a required number of times and in the order necessary for processing data in relationship sets. Data functional features are considered as autonomous and intercompatible data processing algorithms and program modules of their realization. Thus, with a change in data structures, data contents, and algorithms of problems, it is possible to change subsets of the sets of data functional feature so that one could solve problem considering a different situation. This is called data a functional feature set adaptability. For solving new and different problems, using special changing methods, the set may be supplemented with new functional features that would embrace and realize algorithms of new problems. Three classes of data functional features are presented and illustrated here by proper examples typical of those classes. The above statements apply to the case only when initial data of problem solution and its results can be presented as relational sets.

Keywords: data functional feature, functional feature set adaptability, relationship set.

1 Introduction

So far the structure of a data relationship model, its integrity, limitations, etc. have been approached differently by various researchers. [1], [2], [3], [4]. In some papers efforts were made to co-ordinate and unify the above distinct views, e.g. [5]. This paper deals with the sets of functional features of data functions and their adaptability which have not been considered earlier in the frame of the relationship model.

The to key objective of the present paper is to demonstrate feasibility of sets of data functions as well as show their advantages in adapting data processing problems to varying data structures, contents, and problem solution algorithms.

For the sake of simplicity, the main structures of the data relationship model are represented in a formalized way more suitable for defining and solving the above problems.

The formal expressions

$$r = \frac{A_j}{c_{ij}} = \begin{array}{c|cccc} r & A_1 & A_2 & \dots & A_n \\ \hline & c_{11} & c_{12} & \dots & c_{1n} \\ & c_{21} & c_{22} & \dots & c_{2n} \\ & \dots & \dots & \dots & \dots \\ & c_{m1} & c_{m2} & \dots & c_{mn} \end{array}$$

are called here as a relationship set. In this case, $1 \leq i \leq m, 1 \leq j \leq n$.

$R(A_j)$ is a scheme of the relationship set diagram made up of the names of attributes. Let us call the attribute A_k and its values c_{ik} of the same $j = k$ as an attribute domain $dom(A_k)$. An attribute value series of the same value $i = l$ will be called a tuple l . All attribute values c_{ij} form the relationship domain $dom(c_{ij})$. The values c_{ik} of the attributes $A_k \subset A_j$, where any $c_{ik} \neq c_{dl}$, $d \in l$ and $d \neq i$ are called as the tuple keys. In our case, the functional features of attribute values represent data functional features, while every functional feature is a kind of algorithm determining the mode of processing attribute values of the same relationship set. It is evident that a set of functional features also includes the features allowing sets of data that belong to the same or some other diagram, to be transformed into a single set or integrate various sets in a different way to process all the data c_{ij} needed for solving the problem simultaneously.

Let us consider sample processing of c_{ij} . For this purpose, classes of functional data features, their sufficiency and compatibility as well as extensibility by adding some other functional features not breaking the integrity of the functional features system should be determined. The capability to manipulate the same set of functional features when solving various data processing problems allows us to use different applied problem algorithms as well to adapt to the altered data structures or contents without extending the range of functional features. In case the need for such an extension arises, a feasible mechanism to do it may be easily traced from the formalized expression of functional features.

Based on the above considerations, this phenomenon is called data processing adaptability. The variety of functional data features and their inter correlation as well as the ability of extending a set by adding some new features make it possible to embrace essential peculiarities and distinctive features of functional dependencies and functional independencies by functional features algorithm.

2 Composite attributes and relationship sets of fixed capability

A set of functional features is aimed at data, i.e. c_{ij} , processing. Therefore, the more elements make up a set and the more diverse classes of these elements for the set, the more detailed and universal data processing may be achieved, without extending the set of functional features by adding some new features. To determine the peculiar features of a particular problem solution, it is sufficient to analyse the semantics of c_{ij} attributes and the relations of c_{ij} characteristic

of other relationship sets. If the attributes are composite forming more than one independent statement, then the expression of one and the same attribute referring to two parts, A and B, of the relationship set ω' will be as follows:

u'	A_1, A_2, \dots, A_n
B_1	$c_{11}, c_{12}, \dots, c_{1n}$
B_2	$c_{21}, c_{22}, \dots, c_{2n}$
\vdots	$\dots\dots\dots$
B_m	$c_{m1}, c_{m2}, \dots, c_{mn}$

In this case, attributes consist of:

$B_1 A_1$	$B_1 A_2$	\dots	$B_1 A_n$
$B_2 A_1$	$B_2 A_2$	\dots	$B_2 A_n$
\dots	\dots	\dots	\dots
$B_m A_1$	$B_m A_2$	\dots	$B_m A_n$

Thus, it may be stated that the relationship set ω' is the same as the set ω , as it is interpreted by many authors except for the fact that each attribute of the set ω' has a single value c_{ij} :

u'	$B_1 A_1, B_1 A_2, \dots, B_1 A_n, \dots, B_m A_1, B_m A_2, \dots, B_m A_n$
	$c_{11}, c_{12}, \dots, c_{1n}, \dots, c_{m1}, c_{m2}, \dots, c_{mn}$

Therefore, we may say that there are no formal differences between ω and ω' , however by presenting ω' possesses composite attributes as ω by a single tuple c_{ij} , an inverse action i.e., decomposition of the attributes into independent propositions is hardly possible without making errors. E.g. let us have propositions d and e , where

$$d, e = A$$

while h and g belong to

$$h, g = B.$$

Then by integrating the attributes into a single proposition we get:

$$dehg = AB$$

However, when dividing $dehg$ into two parts again an error can be easily made, e.g. by joining together dh with g' and eg with dh .

In the latter case, certain values of the attributes A_k would acquire wrong properties not typical of them if at least one AB in the complete relationship set diagram was reconstructed in the wrong way. Therefore, the structure of ω' allows us to get a clear idea of A_k features and, consequently, to design more detailed and comprehensive data processing based on their functional properties as seen from theoretical and practical standpoint.

The principal difference to be emphasized referring to the keys of the tuple of π is that they do not make a part of an attribute but merely represent the attribute values. However, E_1, E_2, \dots, E_n are a part of the respective attribute of $a_{1j}, a_{2j}, \dots, a_{nj}$ or its sub attribute.

3 Presentation of attribute values (structural approach)

Relationship sets may be identified in a different way in order to facilitate gathering the proper sets to solve a particular problem as well as queuing them in required order [1]. However, the above identification is not sufficient to define an adaptive general data processing problem solution since it requires the positioning of any attribute value A_k in the structure of a relationship set. This makes it possible to use the same problem solution algorithm for processing sets of a different domain belonging to the same diagram. Therefore, a necessary condition is that π and π' be ordered, while A_k be given concrete addresses in a relationship set. The address of any A_k of the set π may be denoted by a digit j designating the current number of an attribute domain since, as a rule, the values of the same attribute possess the same functional features. In the case of the set π' , the address is denoted by i/j , which means that the current numbers of a tuple and a domain should be given. It follows that any changes of the diagrams of π and π' lead to treating a relationship set as a different one requiring a different identifier for its identification. The addresses of attribute values in a relationship set may be also identified in other ways. For example, tuple keys may be used for this purpose. Another method is to define the distance from the beginning of a complete ordered set to a particular A_k by a natural number, etc.

4 The structure of a set of data functional features

In order to process data of the sets π and π' , a set of functional features F should be formed:

F	\bar{A}	\bar{A}	...	\bar{A}
	f	f	...	f
	f	f	...	f

	f	f	...	f

All the attributes \bar{A} of the set refer to the same thing, i.e. functional features, but every f consists of an ordered set of functional features:

$$f = (f_1, f_2, \dots, f_k),$$

where k may vary from 1 to k , with k acquiring different values for different f .

The fact that one and the same value of an attribute may be applied different f when solving different data processing problems might be assumed as one of the adaptability features.

Prior to considering some particular functional features it should be noted that, in essence, functional features of the attribute values c_{ij} represent their processing algorithms which technically may be implemented as independent but intercompatible program modules, i.e., able to act operate together and in any order.

5 Functional data features

The attribute values c_{ij} may have three levels of functional features. First, these are unary functional features that determine the processing of one $c = c_{ij}$. Binary functional features establish a logical or arithmetical relation between two separate c belonging to the same relationship set. Multiprocessing functional features operate with the amount of data c_{ij} needed to be processed in solving a particular problem, which may belong either to the same or to different relationship sets.

In the formal expressions of functional features given below, they will be denoted by f^x , where x refers to a functional feature code that identifies each functional feature thus differentiating it from others. Every c_{ij} in the relationship set will be identified by an address, expressed as mentioned above, in the sets ω and ω' by the components j , and i/j , respectively. The address j is sufficient for the set ω though it refers to all the values of the same attribute belonging to the same attribute domain. It is evident, that the values of the same attribute from the same set possess the same functional features.

As concerns unary functional features, their formal expressions will include a datum itself instead of an identifying the datum address

$$c = c_{ij}.$$

The functional feature

$$f^x \left(c = \begin{Bmatrix} T \\ y \end{Bmatrix} \right)$$

represents a test whereby it is possible to determine whether c is a digit, when

$$f^x (c = y),$$

or whether c is text, if

$$f^x (c = T).$$

This is important since if, for some reason, i.e. because of an error, the text is found in an address instead of a digit, further processing will be disturbed because the performing of arithmetical operations will be impossible.

Another unary functional feature limits the numerical value of c

$$f^x (c_1 \leq c \leq c_2)$$

by the numbers c_1 and c_2 . The limitation from both sides may result not only in the equality, or only in more or less but also in any other form of limitation: $<, >, \dots$

If c refers to a small amount of different values, then all the values may be presented as a formalized expression of a functional feature:

$$f^c(c \vee (c_1, c_2, \dots, c_k)).$$

This means that c should be equal to $(\vee) c_1$ or c_2 , and so on.

If we have to determine the equivalence of c to at least one of the many attribute values of the same domain (not necessarily in a single relationship set of the same intentional), then

$$f^c(c \vee \{j\}),$$

where j is the current number of the domain or an address in the set $\{n\}$.

If a certain amount of data has to be substituted for a c , then

$$f^c(c \vee j(j, j, \dots, j)),$$

where c , in case of its coincidence with a certain datum in the attribute domain j , is replaced by data of the addresses (j, j, \dots, j) from the same tuple which had c equal to the j address content.

Obviously, it is possible to construct the required amount of functional features that enable us to manipulate one attribute value (i.e. to control or change it, etc.).

Binary functional features should be analysed if the relationship set ω' is available and by performing some arithmetical operations with the attribute values of this set.

Let the relationship set ω' be given:

ω'	A_1	A_2	A_3	A_4	A_5
B_1	a_1	a_2	a_3	a_4	a_5
B_2	a_6	a_7	a_8	a_9	a_{10}

Where digit attribute values are arithmetically related as follows:

$$a_1 + a_2 \times [a_3 - a_4 \times (a_5 - a_6)] + a_7 : (a_8 - a_9 \times a_{10}).$$

The above arithmetical operations can be replaced by references to the ω' address contents as follows:

$$a_6 - a_8, 1/5 - 1/6 \rightarrow 1/5.$$

Since the content of the address $1/5$ will no longer be used in operations, we may write the result of subtraction in the address $1/5$. Other operations are performed analogously and the content of an address is written in brackets like $(1/5)$:

$$\begin{array}{ll}
e_{r_6} \times e_{r_{10}}, & 2/4 \times 2/5 \rightarrow 2/4; \\
e_{r_3} \times (1/5), & 1/4 \times 1/5 \rightarrow 1/4; \\
e_{r_9} - (2/4), & 2/3 - 2/4 \rightarrow 2/3; \\
e_{r_2} - (1/4), & 1/3 - 1/4 \rightarrow 1/3; \\
e_{r_7} : (2/3), & 2/2 : 2/3 \rightarrow 2/2; \\
e_{r_8} \times (1/3), & 1/2 \times 1/3 \rightarrow 1/3; \\
e_{r_1} + (1/3) + (2/2), & 1/1 + 1/3 + 2/3 \rightarrow 2/2.
\end{array}$$

One can see from the above example that the result of all operations will be written into the address 2/2, and, what is even more important, all the functional features at a certain moment of their processing are binary, i.e. are performed with two attribute values. Moreover, a conclusion can be definitely drawn that any arithmetical relationships between the attribute values may be presented in terms of addresses interconnected by arithmetical operation signs:

$$L \psi L \dots \psi L \rightarrow L,$$

where L is n address j or n address i/j , while ψ is a sign of addition (+), subtraction (-), multiplication (\times), division ($:$) or refers to some other arithmetical operation. The capability to manipulate the attribute values can be considerably extended if a logical comparison of the above operation is introduced:

$$\begin{array}{l}
L_1 \psi L_1 \psi \dots \psi L_1 \rightarrow L_1; \\
L_2 \psi L_2 \psi \dots \psi L_2 \rightarrow L_2; \\
L_1 \theta L_2,
\end{array}$$

where θ is one of =, \neq , <, >, \leq , \geq , ... Thus, the comparison of arithmetical and logical operations as well as their diverse multiple application allows us to solve various mathematical problems, applying the same functional features. In this case, the adaptation manifests itself by a varying amount and order of the same functional features available for solving various mathematical problems.

Multiprocessing functional features can be quite diverse. In this paper, some major types, i.e. the functional features of relationship set transformations have been considered.

A formal expression of the transformations of relationship sets is of the form:

$$f^r(A_1 A_2 A_3 (A_4))_{\theta \theta'}$$

This expression may contain at most four sets of addresses coded by 1, 2, 3, 4. Moreover, the fourth set, or a data source, may have the required number of the addresses of data sets belonging to the same diagram. Thus, A_1, A_2, A_3 and A_4 actually are not data sets, being merely the internal addresses of the above sets, the content of which is compared under the condition

$$r \theta r', \text{ where } \theta \text{ may be: } =, \neq, >, <, \wedge \dots$$

The content of the set addresses is compared as follows

$$A_1 \sim A_3 |_{\theta \theta'}$$

while, if the condition is satisfied, the data are transferred from A_4 to A_2 set addresses

$$A_2 \leftarrow A_4 \mid_{\#}.$$

The transformation of data in a single operation for the sets ω and ω' having the same or different structure is performed (if the condition is fulfilled) as follows:

- from tuple to tuple, as $\omega_2 \leftarrow \omega_4$;
- set domain to set domain, as $\omega'_2 \leftarrow \omega'_4$;
- from domain to tuple, as $\omega_2 \leftarrow \omega'_4$;
- from tuple to domain, as $\omega'_2 \leftarrow \omega_4$.

It should be noted, that some cases represent partial transformation, where the formalized expression of a functional feature includes three or two sets. It follows that the condition $\#$ is fulfilled in one or both relationship sets in which transformations are performed:

$$\begin{aligned} & f^{\#}(A_1 A_2 A_4)_{\#}; \\ & f^{\#}(A_2 A_3 A_4)_{\#}; \\ & f^{\#}(A_2 A_4)_{\#}; \end{aligned}$$

If a condition is not stated, then a condition less transformation is made:

$$f^{\#}(A_2 \leftarrow A_4).$$

A different case is where a condition is given, with no indication of sets, from the addresses of which data transformations and data transfers should be made, i.e.

In this case, a comparison of some data needed for their processing should be made.

The transformations may be divided into several classes. These are conditional and conditional continuous transformations.

Conditional transformations are being performed only until the condition $\#$ has been satisfied for the first time.

A continuous conditional transformation is being performed until the whole source of data (A_4) run short.

Moreover, all sorts of transformations may be performed in a tuple, a domain and in a combined tuple. Transformations in a tuple will result in:

$$\frac{\omega_4}{\begin{array}{|c|} \hline \alpha_1 \ \alpha_2 \\ \hline \alpha_3 \ \alpha_4 \\ \hline \end{array}} \rightarrow \frac{\omega_2}{\begin{array}{|c|} \hline \beta_1 \ \beta_2 \\ \hline \beta_3 \ \beta_4 \\ \hline \end{array}} = \frac{\omega_2}{\begin{array}{|c|} \hline \alpha_1 \ \alpha_2 \ \beta_1 \ \beta_2 \\ \hline \alpha_3 \ \alpha_4 \ \beta_3 \ \beta_4 \\ \hline \end{array}}$$

The result of transformations made in the domains for the same ω_4 and ω_2 , will be of the form:

a_2	
	$b_1 \quad b_2$
	$b_1 \quad b_2$
	$a_1 \quad a_2$
	$a_1 \quad a_2$

The transformation in a combined tuple could be performed in the following way. Let the addresses of A_3 be 1&3&3, and the addresses of A_4 be 2 and 4, then $f = \langle 1, \emptyset, 3, \emptyset \rangle \leftarrow \langle 2, 4 \rangle$ will yield a result given below if a_3 and a_4 have not changed:

a_2			
	b_1	a_1	b_2
	b_1	a_1	b_2
	b_2	a_2	a_2
	b_2	a_2	a_2

It is evident, that only a part of capability of the transformation method is demonstrated here. In general, speaking about multiprocessing functional data features, it should be mentioned that they enable a great many of diverse diagrams of relationship set attribute values to be presented as a single relationship series of either a_2 or a_2' which considerably simplifies the perception of an algorithm used allowing to design a better solution of an applied problem with the help of various functional features.

6 Conclusion

The problems considered and the results obtained can considerably facilitate designing efficient solutions of problems as well as adapting the systems of functional sets to varying conditions.

Important conclusions can be drawn referring adaptability which, in general, may be considered as a means allowing to solve various problems or those, with altered algorithms, data structures or contents by manipulating the same functional features, with no need for developing new original programs. If, in some cases, available functional features used for data processing operations are not sufficient, the ways of creating new functional features to extend a set of functional features can be easily seen. The latter, when created, could be used to perform impossible so far procedures, without violating the adaptability of functional sets and the integrity of a particular set.

References

- [1] J. D. Ullman: Principles of Database and Knowledge-Base Systems, Computer Science Press, Rockville, 1986.

- [2] H. Thalheim: *Dependencies in Relational Database*, Teubner, Leipzig, 1991.
- [3] J. H. Baker: *Semantic Data Modeling*, Prentice Hall, 1992.
- [4] A. Haener, G. Szala: *Datenbanken, Konzepte und Sprachen (in German)*, International Thomson Publishing Group, 1995.
- [5] A. Eisenmann-Zschauwitz: *Current Issues in Database and Information System*, 2000.