INFORMATICA, 2002, Vol. 13, No. 4, 381–392

Data Aggregation Sets in Adaptive Data Model

Petras Gailutis ADOMĖNAS, Algirdas ČIUČELIS
Vilnius Gediminas Technical University
Saulėtekio 11, LT-2040 Vilnius, Lithuania
e-mail: adomenas@fm.vtu.lt, fmf@fm.vtu.lt

Received: January 2002

Abstract. This article presents the ways of identification, selection and transformation of the data into other structures. Relation selection and transformation may change data quantity and order of laying out. As a result the data are aggregated to the structure needed for application problem algorithm. Data aggregation makes possible to adapt data structure presentation order and quantity for any application problem. Naturally, there must be enough necessary data in the relation sets for any application problem.

Key words: input data, application problem, data sets, relation.

1. Introduction

The adaptive data model consists of two parts. These are aggregation data sets and data processing. This article investigates the first part only, i.e. data aggregation problems. The data aggregation model is composed of input data identification and data transformation. Input data are relations and input data identification means identifier formation of general characteristics for all attribute meanings in one relation. This identifier lets select needed relation from the others. Transformation is laying out of different relations in a necessary order and quantity. Besides, it is possible to change structure by removing into one aggregate data set that is necessary for solving a concrete application problem. An application problem is regarded here as a data processing problem with the only requirement: its input data are relational type data sets (DS) defined in (Adomenas, 2001). The correctness of adaptability is understood in this case as laying out of formal expression of input data identifiers into separate DS identifiers by which the sets identified have to provide any application problem with input data, i.e., with the required DS schemes, their subsets, extensions or their parts, the necessary amounts, as well as all the input data presentation sequence or order.

There exist different approaches to the structure, integrity, and constraints of the relational data model so far (Ullman, 1988; Thalheim, 1991; Bekke, 1992; Haeuer and Saake, 1995). There is an attempt to unify some of the main attitudes and terminology (Binemann-Zdanowicz, 2000). The relational data model is extended by new conceptions and mechanisms (Adomenas, 2001). In such a situation, it is indispensable to present some of the reasoning and their formal expressions so as they are treated (Adomėnas, 2001).

Let there exist a data set (DS)

$$u = \frac{A_j}{c_{ij}},$$

where A_j are attributes of the DS that make up its scheme, c_{ij} ($1 \leqslant i \leqslant m$, $1 \leqslant j \leqslant n$) are attribute values, n is the rank of the DS, and m is the order or power DS. All the attribute values that have fixed (the same) i are called a tuple, and those having the same j are called a domain of attribute A_j. Let u_k be a set of relational sets which contain all the attribute values c_{ij} or input data that are necessary to solve an application problem P_k.

Suppose every set u_k is identified by the identifier W_k of the same k which is going to be defined later. Then the aim of this article can be concretized as follows. Let $< N >$ be a data source and it includes u_k necessary to solve the problem P_k. It remains to prove that there exists an algorithm that is able to arrange DS identifiers W_k in such a way that, it were possible to present u_k according to these identifiers to solve any P_k:

$$u_k \xleftarrow{w_k} < N > .$$

This arrow show the directions of data removing from $< N >$ to u_k.

The most general idea of the formation theorem is formulated in this way. Since there is no need to have all the attribute values c_{ij} of the sets u_k to solve P_k, one has to prove that there exists an algorithm of transformtion T^l such that makes it possible to transfer from the sets u_k to the set u only those c_{ij} which are necessary and sufficient to solve P_k:

$$u \xleftarrow{T^l} < u_k > .$$

Here l are kinds of transformations, whose realization enables us to transfer the necessary c_{ij} from u_k to u.

Assume that, to solve a concrete problem P_k we have to present input data, i.e., the necessary amount, different schemes of DS, and in a proper order:

$$u_k = \left(u_1^1, u_2^1, \ldots, u_{n_1}^1, u_1^2, u_2^2, \ldots, u_{n_2}^2, \ldots, u_1^t, u_2^t, \ldots, u_{n_p}^t\right) \tag{1}$$

Here indices $1, 2, \ldots, t$ of u denote different schemes of DS, $1, 2, \ldots, n_1, \ldots, 1, 2, \ldots$, $n_2, \ldots, 1, 2, \ldots, n_p$ stand for extensions (copies of DS of the same scheme, filled with data that are characterized by different parameters of identifiers). Thus $n_1, n_2, \ldots, n_p$ indicate amounts of DS extensions of each different scheme. This formal way of input data presentation is in fact impossible to be used in the model of an application problem. One of the reasons why is that, if a certain problem has great amounts of DS, it is impossible to indicate all of them in a model. Besides, some of the problems require the completeness of input data (each of the schemes $1, 2, 3, \ldots, t$ has $n_1, n_2, \ldots, n_p$ copies of extensions, respectively), while for other problems it is not necessary. In addition, if we wish to use the same data (only in different amounts and another order) for solving other or different problems, it is indispensable to have a universal method for their identification or a model for identifying all DS as a composite part of the mathematical model of an application problem.

2. Data Presentation

Let identifiers of DS have an analogous scheme as DS themselves composed of the attributes A, B, D, where A is the attribute of a scheme identifier, B is the attribute of the extension identifier and D is the attribute of the extension identifier of the time factor. As a rule, the three attribute value codes mentioned are sufficient to identity DS, though it is easy to notice from a further identifier analysis that another number (larger or smaller) of identifiers does not change the essence of their formal expressions.

Suppose attribute A values are a_x, where x are codes of concrete schemes, attribute B values are b_y, where y is the code of a subject which the data belong to, D is the code z of the time factor attribute d_z.

Thus, any DS is identified by a relational set where a_x, b_y, and d_z are the values of attributes A, B, and D, and W_k is the relational set of identifiers:

$$W_k = \left[\frac{A}{a_x}\frac{B}{b_y}\frac{D}{d_z}\right]. \tag{2}$$

Obviously, for any application problem, it is always possible, to arrange input data for its algorithm in the proper sequence, as shown in (1), because otherwise the existence of the problem itself would be impossible, however, in the mathematical model of this problem sequence (1) has to be replaced by the set of identifiers:

$$a_x b_y d_z,$$

Otherwise the data presentation model would not exist and it would be necessary to present input data that may differ for each particular problem. Apparently it is reasonable to present the set of identifiers (2) in the application problem model in a concise mathematical form. Then, for input data of a particular problem, we indicate the necessary parameters typical only of this problem. In such a case, we have to extend the indicated mathematical data presentation form by programming means into a sequence of identifiers of type (2) before solving the problem, later on replacing it by the sequence of input data (1). But then we have to prove that it is possible to extend, by the same ways, the mathematical identifier model with different parameters to a sequence of input data identifiers, correct to any application problem.

Let there any of identifier indices x, y, z be denoted as w and expressed by natural numbers, and any value of identifier attributes a, b, d are denoted by g.

Theorem 1. *One can express any attribute value g_w by the following modifications of the indices w presentation*

$$g_{w=}g_{1-4} \quad \textit{and} \quad g_{20,4-1,8,7,11-13,6}, \tag{3}$$

and by the ways of extension of these modifications:

$$g_{1-4} = g_1, g_2, g_3, g_4 \quad \textit{and}$$
$$g_{20,1-4,8,7,11-13,6} = g_{20}, g_4, g_3, g_2, g_1, g_8, g_7, g_{11}, g_{12}, g_{13}, g_6, \tag{4}$$

that would satisfy the requirements of any application problem algorithm to the ordering of input data identifiers and amounts of DS extensions for each different but necessary scheme of this problem.

In the case $g_w = g_{1-4}$, extension of identifier indices is very explicit, if we have in mind that not only these four indices of identifiers can be involved in the extension, but also some other amount, e.,g.,

$$g_w = g_{1-66} = g_{1,2,3,\ldots,65,66}.$$

Besides, there is no need whatsoever that indices should start from one or that they be of increasing order:

$$g_{12-33} = g_{12,13,14,\ldots,32,33}; \quad g_{12-3} = g_{12,11,10,\ldots,4,3}.$$

It means that all the data identifier presentation and extension variants mentioned are correct and they embrace any variants of concise data presentation and their extension, where identifier indices are presented in successive increasing or decreasing natural numbers. In addition, the number of groups in one g_w of thoseindices is not limited:

$$g_w = g_{2-5}, \;\; g_{66-61}, \ldots, g_{33-38} = g_{2,3,4,5}, g_{66,65,64,63,62,61}, \ldots, g_{33,34,35,36,37,38}.$$

The second case, shown in expression (3), differs from the considered one only in that individual indices or their groups, such as $g_{8,7,33}$ in expansion (4), can be inserted between the given expressions and expansions in any order and quantity. So, the latter case increases and facilitates the possibilities of data presentation variety even more than the first case considered. After estimating all the given cases, it is evident that the indicated expansion ways allow DS identifiers, included in expression (2), to present input data in any order and quantities, necessary for a particular application problem. First of all, the identifiers are extended according to a indices, then b indices, and at the end by d indices. This makes up, however, but one sixth of all the possible expansion ways, since $a_w b_w d_w$ arranged in any order identify one and the same DS. Then we can select, for a particular application problem, one of the suitable expansion ways with a different identifier arrangement ordering in the expansion of the same DS identifiers:

$$a_w d_w b_w \equiv a_w b_w d_w \equiv b_w a_w d_w \equiv b_w d_w a_w \equiv d_w a_w b_w \equiv d_w b_w a_w. \tag{5}$$

The latter identities show all the possible arrangement both in the presentation and extension of an identifier. When analyzing expression (3) we have shown all the possible variants, indicating DS quantities and their ordering inside of each permutation in expression (5). Consequently, the ability of satisfying the requirements to input data for any application problem algorithm has been proved.

3. Data Transformations

Let input data to solve a particular problem be collected in data sets u. We regard data transformation to be data transference from one and the same scheme DS u_k to a set u, having a possibility to change the structure and content of DS during the transformations. If we transfer data of other DS schemes to u, then we need other kinds of transformation.

Possible variants of transformations from one and the same scheme of DS are as follows:

a) all c_{ij} are necessary from all DS without changing the ordering of c_{ij};
b) all c_{ij} are necessary from all DS changing the inner ordering of c_{ij};
c) a certain part of tuples is necessary from each DS;
d) a certain part of domains is necessary from each DS;
e) certain separate c_{ij} are necessary from each DS;
f) all the possible combinations of variants b, c, and d are necessary from different DS schemes: b, c, d; b, c; b, d; c, d.

The most general formula of transformation will be:

$$u \xleftarrow{T^l} < u_k > .$$

Dependent on the transformtation formula parameters, one can transfer DS data from sets to the same set u, thus forming u of quite a different structure and content than those of (1). Obviously, we have to extend the abovementioned most general transformation formula up to the reference to inner addresses of DS c_{ij} having a possibility of selecting necessary tuples, domains, individual c_{ij} and all possible their combinations in such a way that it were possible to construct u such that would enable us to choose input data in their content, structure and quantities suitable for any application problem:

$$T^l\Big(\{i/j\}^1[i/j]^2 \xleftarrow{q} \{i/j\}^3[i/j]^4\Big). \tag{6}$$

Here l denotes the code of the kind of transformation to be defined later on. $\{i/j\}^1$, $[i/j]^2$, $\{i/j\}^3$ and $[i/j]^4$) stand for attribute value addresses in the sets identifier by upper indices 1 and 3, 2 and 4, all the four, three or two of which can be involved in the transformation, dependent on the transformation code l. If all the four sets are involved in the transformation, then the contents of addresses $\{i/j\}^1$ and $\{i/j\}^3$ are mutually compared to verify if they satisfy the transformation condition q, and the data transfer from $[i/j]^4$ to $[i/j]^2$ is performed only in case the condition q has been satisfied. The address of the attribute value c_{ij} in the set is shown by i and j, which can have different values for different sets at one and the same moment of transformation, despite the fact that they are identical in formula (6).

If four sets are involved in the transformation, then in some of them (say 1 and 3) we look for data that satisfy q, while the transformation itself is performed in the other sets (2 and 4). If three sets are involved in the transformation, then

$$T^l\Big(\{i/j\}^1[i/j]^2 \xleftarrow{q} [i/j]^3\Big) \tag{7}$$

In this case, set 1 whose data are directly compared to that of the data source which is identified in the latter formula as set 3, since set 4 is not involved in the transformation. If only two sets are involved in the transformation, then the data are both compared and transformed in the same sets:

$$T^l\Big([i/j]^1 \overset{q}{\longleftarrow} [i/j]^2\Big).$$

The transformation condition q may have different values: $=$, $\neq$, $<$, $>$, $\forall$, $\exists$, etc., or not be present in the transformation formula at all (unconditional transformation). In the case of building up input data, it suffices for q to be $=$, and also two sets are enough: that of the data source and a set that accepts data. The equality and other values of q as well as the whole formula (6) are used in those cases where transformations are directly involved in data processing but not in the formation of the data set u.

The kinds of transformations expressed by the code l may be as follows:

- unconditional transformation ($l = 0$);
- transformation when q is satisfied first and thus the transformation is completed by one and the same transformation formula ($l = 1$);
- transformation when q is satisfied, and after the transformation, other cases to satisfy q are sought as long as the whole data source has been looked through, i.e., all the DS of one and the same scheme from u_k, after which the transformation is completed ($l = 2$);
- transformation when q is satisfied, and after it other cases to satisfy q are sought as long as the whole data source has been looked through, i.e., all the DS of one and the same scheme from u_k, returning afterwards to the beginning of the data source and seeking cases to satisfy q, after changing the contents of address $\{i/j\}^1$, and completing the transformation only when all the data in the addresses $\{i/j\}^1$, necessary for comparison have been exhausted ($l = 3$).

In case we need a sub kind of the kind transformation, after the first digit which means a kind of transformation the next digit is indicated, which stands for a sub kind.

Theorem 2. *For any application problem the data transformation*

$$T^l\Big(\{i/j\}^1[i/j]^1 \overset{=}{\longleftarrow} [i/j]^2\Big),$$

can form such an input data set u that it were possible to solve that problem if the data were indicated in the sets u_k, i.e., in the data source.

The addresses $\{i/j\}^1$ and $[i/j]^1$ are addresses of the same set, only the former $\{i/j\}^1$ show the contents of which address is compared with the address contents of the set $[i/j]^2$, while the latter $[i/j]^1$ indicate to which addresses the contents of $[i/j]^2$ are transferred.

If, for each kind of transformation l logically compatible with each variant of the transformation from a to f, there exists a single-valued transformation algorithm expressed by the transformation formula with different parameters, then one can consider this theorem proved.

If, for a particular application problem, all c_{ij} from all the DS (1) are necessary without changing the ordering of c_{ij} (variant a), obviously it is unconditional transformation ($l = 0$)

$$T^0\Big([i/j]^1 \longleftarrow [i/j]^2\Big),$$

where 1, or the structure of the set u and that of the same scheme sets 2 from u_k are identical. Since the contents c_{ij} of the respective addresses of the mentioned sets are also the values the of the same attributes, any expansion of sets in the set u has no sense. In other words, after every transformation we solve the problem of processing u and perform the next transformation of c_{ij} as long as all the sets of the same scheme in the data source have been exhausted.

If, for a certain application problem, all c_{ij} from all the DS (1) are necessary changing the inner ordering of c_{ij}, then we divide the transformation procedure into two sub kinds of $l = 0$: namely, $l = 01$ and $l = 02$. As $l = 01$, the transformation procedure is just like in the previous case except that the addresses $[i/j]^1$ are ordered other than that of $[i/j]^2$. There can be $x = [(m \times n)! - 1]$ different orderings, where m is the order of the set, and n is its rank. Thus, we subtract a unit from all the possible transformations of addresses, since the ordering of $[i/j]^2$ cannot by repeated and the transformation has the necessary condition, in this case, to change the ordering of c_{ij}. As $l = 02$, all the sets of one scheme in the resulting set u are accumulated in the row of sets

$$T^{02}\Big([i + n/j]^1 \longleftarrow [i/j]^2\Big),$$

or the column of sets

$$T^{02}\Big([i/j + m]^1 \longleftarrow [i/j]^2\Big).$$

In both cases transformations are performed by inserting one set of the same scheme into u each time distinguishing the first transformation, and increasing the current address components by the rank $i + n$ or power $j + m$ of the set.

If there is a need for a certain part of tuples from each of the same scheme DS (variant c), then

$$T^2\Big(\{i/j\}^1[i/j]^1 \overset{=}{\longleftarrow} [i/j]^2\Big).$$

The set that accepts data and the one indicating tuple identifiers is the same set, though, if we have to preserve identifiers, we may construct a special set of tuple identifiers $\{i/j\}^1$:

$$T^2\Big(\{i/j\}^1[i/j]^2 \overset{=}{\longleftarrow} [i/j]^3\Big).$$

Ordering of data transferred to u, in the set $[i/j]^2$, into tuples or domains following successively one after another has no difference in principle, i.e., has no influence of principle on the algorithm of an application problem, therefore it is not considered as a separate case. Consequently, as a result of transformation, u will consist of all the tuples of sets that have one and the same key of one and the same scheme. When

$$T^3\Big(\{i/j\}^1[i/j]^2 \overset{=}{\longleftarrow} [i/j]^3\Big),$$

u will contain all the tuples of the indicated keys of all $\{i/j\}^1$.

If we need a certain part of domains from one and the same DS, then the transformation is unconditional, because domains are fixed in a scheme by one component i of an address and that address must not change in set 2 since, in the opposite case, we should have a set of another schemes. However, ordering of addresses i in the set u, i.e., set 1, in this case, can be any, in accordance with the requirements of the algorithm of a particular application problem:

$$T^0\Big([i]^1 \longleftarrow [i]^2\Big).$$

Attribute values of one and the same domain are transferred to the same domain as long as the sets of the same scheme

$$T^{03}\Big([2,4,1,6]^1 \longleftarrow [1,2,3,4,5,6]^2\Big)$$

have been exhausted in the data source. With such a transformation formula, the data source consists of two sets:

$u_k =$

A	*B*	*C*	*D*	*E*	*F*
1	2	3	4	5	6
a	2	4	77	3	4
d	11	5	4	1	2
b	34	3	5	7	9
e	4	6	8	3	2

,

A	*B*	*C*	*D*	*E*	*F*
1	2	3	4	5	6
c	5	66	2	6	7
f	3	23	1	2	4
a	3	6	8	8	6
d	1	9	5	4	8

Here numbers 1, 2, 3, 4, 5, 6 denote addresses i of the domains, *A, B, C, D, E, F* are attributes, and c_{ij} are attribute values. The set u is of the shape

$u =$

B	*D*	*A*	*F*
2	4	1	6
2	77	*a*	4
11	4	*d*	2
34	5	*b*	9
4	8	*e*	2
5	2	*c*	7
3	1	*f*	4
3	8	*a*	6
1	5	*d*	8

We can transform the domains by means of conditional transformations:

$$T^{31}\Big(\{i\}^1[i]^2 \overset{=}{\longleftarrow} [i]^3\Big).$$

In this case, we indicate the necessary attribute values in the addresses $\{i/j\}^1$, and if a domain has at least one the same value, then it is transformed.

For instance, if $f, 3 \in \{i/j\}^1$, then

$u =$

C	E	A	B
3	5	1	2
4	3	c	5
5	1	f	3
3	7	a	3
6	3	d	1

If certain individual c_{ij} from each DS of the same scheme are necessary, then this case is analogous to the previous one, only we transform not all the domains indicated, but only those c_{ij} that are indicated for two addresses. Therefore the formulas must also contain the component of addresses j:

$$T^{04}\Big([i/j]^1 \longleftarrow [i/j]^2\Big),$$
$$T^{32}\Big(\{i/j\}^1[i/j]^2 \longleftarrow [i/j]^3\Big).$$

Mixed variants of transformations presented in case f: b, c, d; b, c; b, d; c, d, are evident since they are performed using the abovementioned formulas, each case presenting a corresponding formula. Before that, one has to properly adjust the ordering of sets of various schemes in the data source u_k. It means that the sequence of transformation formula ordering has to correspond to the order of DS extensions of the same schemes assigned to those formulas. The transformation is performed by the same formula until the extensions of the set a_x of the same scheme$x$have been exhausted in the set u_k. Then we go over to another transformation formula and take in turn DS extensions from another scheme in the data source u_k, whose data have to be transformed by the latter formula.

Example. Suppose a_1 is the goods transportation relation the scheme of which is formed of attributes: A_1 as the quantity of transported goods, A_2 as denomination of transported goods, A_3 as the transportation distance, A_4 as the vehicle identification numbers. Then goods transportation DS is:

$$a_1 = \left[\frac{A_1}{c_{i1}}\,\frac{A_2}{c_{i2}}\,\frac{A_3A_4}{c_{i3}c_{i4}}\right],$$

where i shows the number of runs.

The names of transportation enterprises are b_1, b_2, b_3; d_1, d_2 mean the transportation dates.

It is obvious that every a_1 must be identified by one of b and d. Thus, to solve application problems of every transporter it is necessary to get all their DS as input data of a concrete problem:

$$W_1 = a_1 b_1 d_{1,2}; \quad W_2 = a_1 b_2 d_{1,2}; \quad W_3 = a_1 b_3 d_{1,2}.$$

To determine the number of transported goods a day the input data should be:

$$W_4 = a_1 b_{1-3} d_1; \quad W_5 = a_1 b_{1-3} d_2.$$

Naturally, W_k aggregates the input data only u_k, where k may vary from 1 to 5. To solve transportation problems resulting in numbers certain data transportations should be made. An example of such transportation may be data moving from one structure into another if we want to know how many goods are transported by the vehicle c_{34}. Then out of all transportation data

$$W_k = a_1 b_{1-3} d_{1,2},$$

the c_{i1} with c_{34} in the fourth column of its tuple is transformed:

$$T^2\big(\{1\}^1[2]^1 \xleftarrow{=} \{4\}^2[1]^2\big).$$

In the address $\{1\}^1$ of the first column c_{34} is presented. It is compared with the contents of input data, i.e., with the contents of the fourth column of the second DS $\{4\}^2$. When the data coincide, the number indicating the quantity of transported goods is moved from $[1]^2$ to $[2]^1$. Then at the end of transformation all the quantity of goods taken by c_{34} will be in $[2]^1$. By similar selection and transformation of input data it is possible to aggregate necessary input data for solving any application problem if there are sufficient data in W_k.

4. Conclusions

The theorems proved allow us to assert that a part of the adaptive data model – input data presentation model – is correct. For any application problem, if only its input data are of defined DS structure, it can provide with input data of the necessary quantity, structure, contents, and in the proper order. We confirm thereby a possibility of the existence of an adaptive data model in terms of data presentation.

References

Ullman, J.D. (1988). *Principles of Database and Knowledge-Base Systems*. Computer Science Press, Rockville.
Thalheim, B. (1991). *Dependencies in Relational Database*. Teubner, Leipzig.
Bekke, J.H. (1992). *Semantic Data Modeling*. Prentice Hall.

Haeuer, A., G. Saake (1995). *Datenbanken, Konzepte und Sprachen*. International Thomson Publishing Group (in German).

Binemann-Zdanowicz, A. (2000). Current issues in databases and information system. In *East-European Conference on Advances in Databases and Information Systems Held Jointly with International Conference on Databases Systems for Advanced Applications*. Prague, Czech Republic. pp. 307–314.

Adomenas, P.G. (2001). Data functional feature sets and their adaptability. In *The 5th East-European Conference on Advances in Databases and Information Systems*. Vilnius, Lithuania. pp. 131–140.

P.G. Adomėnas is a head of Information Systems Department in Vilnius Gediminas Technical University. His research interests include investigation of relation data model adaptability.

A. Čiučelis is a dean of Fundamental Sciences Faculty in Vilnius Gediminas Technical University. His research interests include application of optimization theory and methods in modern technologies.

Duomenų agregavimo aibės adaptyviajame duomenų modelyje

Petras Gailutis ADOMĖNAS, Algirdas ČIUČELIS

Tiriami reliacinio duomenų modelio adaptyvumo ypatumai. Agreguojamos pirminių duomenų aibės taip, kad jos tiktų bet kokiam taikomajam uždaviniui spręsti, jeigu tik duomenų šaltinyje (bazėje) jų pakanka. Įrodoma galimybė duomenis agreguoti dviem metodais: reikiamų duomenų aibių atrinkimu ir reikiamų duomenų transformacija į agreguojamas aibes. Agreguotos aibės turi taikomajam uždaviniui reikiamą struktūrą, duomenų kiekį ir turinį.