The Main Adaptability Principles for the Process of Handling Relational Data Sets

BIRUTĖ PLIUSKUVIENĖ, PETRAS ADOMĖNAS
Information Systems Department
Vilnius Gediminas Technical University
Saulėtekio al. 11, Vilnius
LITHUANIA

Abstract: In this paper an adaptive data processing technology is presented that implements solutions to applied problems in two stages. The first stage consists of data aggregation, i.e., data identification and transformation into the primary relational data set for an applied problem, and the second stage deals with algorithmic dependencies and the generation of implementing program modules into an application for an applied problem. The intent of this technology is to lessen the faults in solving a problem as data and the algorithms for their processing change. In all instances this is achieved not by creating a new program but by adapting certain subset of the program module set into a particular program for solving a problem.

Key words: relational sets, algorithmic dependencies, data aggregation, program generation.

1 Introduction

As an application domain changes, it often becomes necessary to change the structure and contents of primary data as well as the algorithm for solving problems, so applied problems have to be designed, programmed and included into already functioning systems anew. In the latter case not only many problems arise in lessening the faults of problem solving, but the time and expenses for solving the problem increase as well. Taking this into account an adaptive data processing technology has been created that implements the solutions of applied problems in two stages (Fig. 1).

In the first stage, data selection is performed out of the totality of primary data supplied as relational sets using identification methods. The data required for the solution of a particular problem are selected and provided in the order needed. Through further data transformations the data selected are placed into one common relational set (RS), which can also be composed of result data, since during transformations both arithmetic and logical operation can be performed. The structure and contents of primary data RS can be changed with transformation, too.

In the second stage, having one common RS composed of the data needed for the solution of a particular problem, this problem is solved as a certain implementation of algorithmic dependencies among data because the algorithm for solving a problem can be considered as a sequence of algorithmic dependencies. To implement these algorithmic dependencies program modules are created whose codes are provided in the RS. Algorithmic dependencies and implementing program modules are physically matched with one another in any combinations, and their logical placement is set by the designer according to a particular problem-solving algorithm.

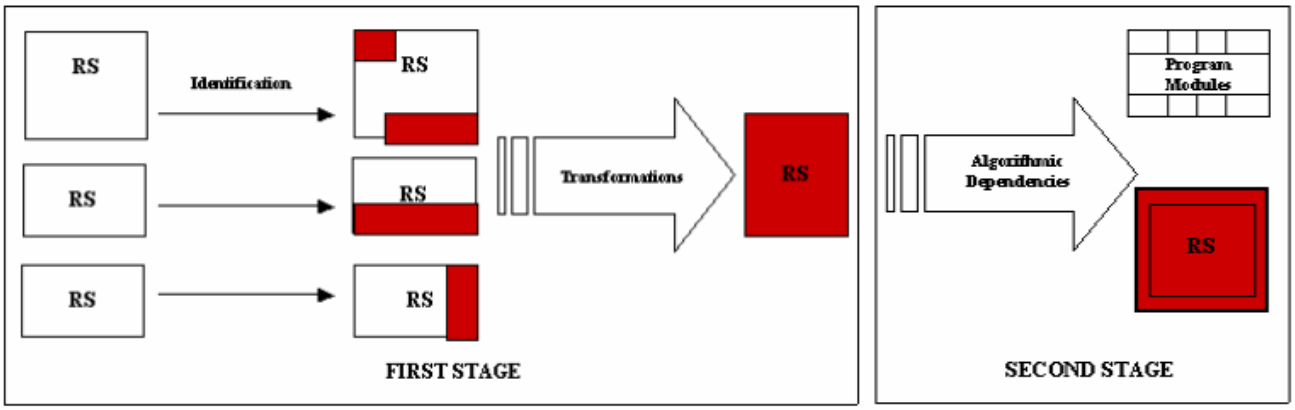

Fig. 1. Two stages of applied problems solutions

Adaptive data processing technology can be used only with data structures expressed as relational sets. Primary and result data on real world objects, events, processes, etc. can be expressed as relational sets. This form of expression has been chosen because: RS can be easily presented in data tables; data in data tables can be considered sets, and the set theory in mathematics is well-developed; the information presented in table is easy to transform into a mathematical construct, relational sets (the latter also have a flat quadrangular shape); the expression of relational set possesses a large portion of technical, technological, economic, scientific and other information; a table is a collection of data very evident and simple for humans to understanding.

To this day different viewpoints exits to the structure, integrity and limitations of a relational data model [1], [2], [3], [4]. Efforts are made to unify some basic tenets and terminology [5]. As the things stand, it is necessary to provide some concepts and their formal expressions as they are interpreted.

2 Relational Sets and Their Elements

The basis of the data model described is composed of relational sets whose structure is as follows:

A_1	A_2	…	A_j	…	A_n
c_{11}	c_{12}	…	c_{1j}	…	c_{1n}
c_{21}	c_{22}	…	c_{2j}	…	c_{2n}
…	…	…	…	…	…
c_{i1}	c_{i2}	…	c_{ij}	…	c_{in}
…	…	…	…	…	…
c_{m1}	c_{m2}	…	c_{mj}	…	c_{mn}

Set elements A_j (where $1 \leq i \leq m$, $1 \leq j \leq n$) are called attributes, c_{ij} – attribute values. All these elements are connected among themselves both by various parameters and semantically. Relational set is a data set that expresses correspondence between attribute names

A_1	A_2	…	A_j	…	A_n

and their values

c_{11}	c_{12}	…	c_{1j}	…	c_{1n}
c_{21}	c_{22}	…	c_{2j}	…	c_{2n}
…	…	…	…	…	…
c_{i1}	c_{i2}	…	c_{ij}	…	c_{in}
…	…	…	…	…	…
c_{m1}	c_{m2}	…	c_{mj}	…	c_{mn}

The set of attribute names is called an RS schema:

$$R = R(A_j) = R(A_1, A_2, \ldots, A_j, \ldots A_n) \quad (1)$$

The set of RS attribute values $< c_{ij} >$ is called an RS domain:

$$D = < c_{ij} > \quad (2)$$

A subset of the RS domain $< c_{ij} >$ composed of attribute values where i fixed and $i=a$ is called a tuple of attribute values:

$$t_a = t(A_a) = c_{a1}, c_{a2}, \ldots, c_{aj}, \ldots, c_{an} \quad (3)$$

A certain subset R of RS schema is called an internal key, where $K \varepsilon R$:

$$K = \{A_y\}, y \varepsilon j \quad (4)$$

Then the values $\{ c'_{ij} \} \varepsilon \{ c_{ij} \}$ of K attributes are tuple keys or their identifiers.

It must be noted that the latter RS has a fixed n for a particular schema but can have a different m. The set parameter n (the number of attributes) is called a rank. The RS parameter m (the number of tuples) is called the order or cardinality of a set.

When RS are processed by software, data can be addressed in different ways:

- to a separate tuple;
- to a separate domain;
- to a separate value using its coordinate in a set table i / j.

3 Data Provision

For solving many applied problems it is necessary to supply some quantity of data in certain order for processing. Also, in exceptional cases all primary data, i.e. all attribute values, are needed. Taking that into account data provision is organized in the defined adaptive data processing technology by implementing data identification model and transformations. The data identification model can easily provide primary data for various applied problems at the quantity and order required by its algorithm. It also allows to control the completeness of data [6]. Out of the data sets marked by identifiers attribute values are transformed into a primary data set for a particular applied problem. That allows to change data structure and contents.

Data structure identifiers have the schema analogous to that of a relational data set (RDS) composed of attributes: *A, B, D*, where *A* is a code attribute for a RDS schema identifier, *B* – a code attribute for an identifier of the dependence of an extensional to a subject, *D* – a code attribute for an identifier of the dependence of an extensional to a time factor. As a rule, three said codes are enough to identify an RDS; however, after further identifier analysis it is easy to notice that another amount of identifiers in essence does not change their formal expressions.

The values of attribute *A* are a_x, where x are codes for particular schemas, the values of attribute *B* are b_y , where y are the codes of the subject to which data belong and *D* is the code z of the time factor attribute d_z . Therefore, any RS identified by RDS, where a_x b_y and d_z are the values of attributes *A*, *B* and *D*. Then using an identifier set for any applied problem we can select the primary data needed and order them in the sequence required.

Let‘s assume for the problem being solved two independent RDS, three subjects and the same moment in time are needed. In this case the identifier set is:

$$\begin{bmatrix} a_1 & b_1 & d_1 \\ a_1 & b_2 & d_1 \\ a_1 & b_3 & d_1 \\ a_2 & b_1 & d_1 \\ a_2 & b_2 & d_1 \\ a_2 & b_3 & d_1 \end{bmatrix}$$

The provision of the identifiers needed can be organized in a more universal fashion:

$$a_{1,2}\ b_{1\text{-}3}\ d_1 = a_1b_1d_1,\ a_1b_2d_1,\ a_1b_3d_1,\ a_2b_1d_1,\ a_2b_2d_1,\ a_2b_3d_1. \quad (5)$$

Although the results of both provision methods are the same, it is easy to note that the latter method makes it easy to provide large quantities of RDS identifiers in a compact form, and the identifiers can be automatically expanded for solving the problem and calling the RDS needed. In the compact form for providing identifiers a_x b_y and d_z can be reordered freely:

$$a_x b_y d_z,\ a_x d_z b_y,\ b_y a_x d_z,\ b_y d_z a_x,\ d_z a_x b_y,\ d_z b_y a_x. \quad (6)$$

It is also possible to provide index identifiers in parts and not necessarily in sequence, e.g.:

$$a_{1\text{-}100} = a_{1\text{-}20},\ a_{21\text{-}56},\ a_{84\text{-}100},\ a_{79\text{-}83},\ a_{57\text{-}78}. \quad (7)$$

This way we can guarantee the provision of any quantity of RDS required and in any order for solving a particular problem [7]. It is evident that if no RDS whose identifiers are in the identifier sequence of a particular problem is found, it is determined what RDS are missing for solving the problem.

While checking the data completeness in every RDS provided for the solution, it is determined if all tuples are present in every RDS provided, since one of the attributes is marked as a key attribute, and its values become tuple keys or identifiers. So it is necessary to add to the RDS schema the key attribute *K* (4) in a fixed order and the values of that attribute:

$$c'_{1j},\ c'_{2j},\ c'_{3j},\ldots,\ c'_{mj}. \quad (8)$$

Therefore, during attribute value transformation the tuples not needed for solving a particular problem are not touched, but if no tuple out of sequence (5) is found in the RDS, it is easy to determine what data are missing and exactly what tuples are not provided in a particular RDS.

RDS transformations enable to transfer the attribute values needed to one common set out of the already-formed primary data sets for solving a particular problem.

Transformations can be classified into unconditional, conditional, conditional continuous, conditional cyclical and arithmetic according to semantic algorithms. All transformations except for arithmetic serve for forming the RS of the primary data needed for solving a particular problem. Using an arithmetic transformation we can browse a data source and change their contents because it is

possible to perform addition, subtraction and other arithmetic operations.

In all transformation mentioned except for unconditional the following symbols are used to indicate the conditions for comparing data: $=, \neq, >, <, \geq, \leq, \wedge, \vee, \neg$.

In conditional transformations address contents are compared and the transformation is performed only if the condition is satisfied.

Transformation addresses in RS can be of several kinds:

- the addresses out of which data is taken $< k / l >$;
- the addresses into which data is put $< i / j >$;
- the addresses whose contents are compared according to the set condition $\{ k / l \}, \{ i / j \}$.

These addresses are merged into simple not ordered sets because the numbers of addresses being compared can be vary widely. For example, the contents of one address can be compared with a large quantity of source addresses, and the transformation performed if the condition is satisfied. In this case the order of positioning source addresses has no significance whatsoever.

Hence, the address structure of the transformation formula is as follows:

$$f^t(\{ k / l \}^n < k / l >^n \xrightarrow{\Sigma} < i / j >^n \{ i / j \}^n). \quad (9)$$

Here the contents of addresses $\{ i / j \}$ and $\{ k / l \}$ are being compared; Σ is defined as a symbol for one of the comparison conditions mentioned above. When the condition is satisfied, data are transferred from address $< k / l >$ to addresses $< i / j >$. The arrow indicates the direction of data transformation. Index n means the number of set order. At the same time these indices can be all different or all the same. If n values are different, then the comparison of data to satisfy the condition Σ is performed in some groups and the transformation of data in the others. If n values are the same, then data are compared and transformed in the same RS; i.e., the change (data positioning and/or their quantity) of one RS structure is performed.

In case of unconditional transformation condition Σ is not in the formula and neither are the addresses in the parentheses. Hence, the formula of unconditional transformation:

$$f^t(< k / l >^n \longrightarrow < i / j >). \quad (10)$$

In expression $< i / j >$ index n has no meaning because the RS receiving data is always a single one.

In all transformation data can be transferred from one set to another in three ways:

- performing Cartesian transformation;
- performing transformation in tuples;
- performing transformation in domains.

Therefore, set transformations are expressed by the formula that can be substituted for operations of relational algebra and exceed their capabilities in many instances. Since arithmetic and logical operations can be performed during transformations, the RS receiving data can be composed completely or partially of new attribute values c_{ij} not present in the data source. It means that during transformations we can change the structure and contents of data sets if desired.

4 Algorithmic Data Dependencies

Having formed a set of primary data for solving an applied problem, a problem can be solved as a certain implementation of algorithmic dependencies between data because algorithmic dependencies appear between attributes. In other words, while solving this particular problem the attributes selected are associated; i.e., in the space of this problem they depend on each other. From a theoretical standpoint, data dependency is considered an algorithm that describes how data in the same RS are processed while solving a problem in question [6]. Data in this system are actually the values c_{ij} of RS attributes. For each attribute value or for any subset of RS values an ordered set of algorithmic dependencies is selected, the program modules of algorithm implementation of its elements perform data processing:

$$<p_{ij}> \rightarrow <c_{ij}> \rightarrow <c'_{ij}>, \quad (11)$$

where $<p_{ij}>$ is a set of algorithmic dependencies, $<c_{ij}>$ – a set of the primary data selected and $<c'_{ij}>$ – a result of the solution.

Algorithmic dependencies (AD), depending on the nature of operations performed, can be subdivided into six classes.

- *The computational - infological* ADs are subdivided into computational p^+ and infological $p^=$. Computational ADs are used to describe various arithmetic operations also getting the results of these operations. Infological ADs are used to compare the results of various arithmetic operations while checking the compatibility and correctness of various processes.
- *The computational AD in domains* – p^s. Computation-infological ADs are applied to attribute values in RS tuples, so computational ADs are defined that are applied only to RS domains. If arithmetic operations have to be performed in domains, an additional domain of special tuple keys and the sums of various attribute values is

introduced. In this case, the distribution of attribute values and the number of tuples can vary.

- Taking into account that a particular attribute value can change as it is used in solving different problems, *internal algorithm dependencies p^a of attribute values*. This class is composed of markedly different algorithmic dependencies that only rather reservedly can be considered permanent.
- *The AD of program steps – p^p*. Many algorithmic dependencies use data from various RS addresses so it is necessary to start a program step at the required address and continue in the required direction. Otherwise, the process of data processing becomes incorrect or impossible. All program steps in algorithmic RS (ARS) are divided into successive and non-successive. An ARS is a set composed of the identifiers for algorithmic dependencies of attribute values that unambiguously determines the use of attribute values in a certain operation of data processing.
- *The external AD – p^i*. The external dependencies of algorithmic RS regulate the interrelation of the ARS of the data being processed with the adjacent ARS. These links cannot be separated from the RS of the data being processed.

Algorithmic dependencies can change. For one planned processing they can be of one kind between the same attributes, and for another they might need be selected similarly. If for a data processing problem being solved new algorithmic dependencies and their implementing modules are needed because their existing totality cannot perform certain operations for solving a problem, then a new algorithmic dependency and its implementing module are created that are included into existing sets of algorithmic dependencies and program modules. A new algorithmic dependency together with other encompass the implementation of solving a new problem. So it can be said that the set of all algorithmic dependency classes and the number of algorithmic dependencies in a class are open.

To develop the concept of this paper algorithmic dependencies could be illustrated by examples. That would enable one to comprehend the validity and rationality of the process more convincingly.

5 Conclusions

The ability to aggregate data into primary data for any relational problem enables to interpret a complex provision of data uniformly, and that simplifies the understanding of primary data aggregation, since aggregation principles for each problem vary only through technological formula parameters.

A program for an applied problem is formed as a subset of all implementation modules of algorithmic dependencies, so no new programs appear. That enables to solve various problems with the same set of program modules for algorithmic dependencies without creating any new programs.

The adaptive data processing technology is improving or learning in that aggregation and generation codes and their ordered set accumulate while more problems are being solved and that results in more complex procedures.

The limitation of this technology is that both primary data and result data have to be expressed as an RS.

References:

[1] J.D.Ullman. *Principles of Database and Knowledge – Base Systems*. Computer Science Press, Rockville, 1988.

[2] B.Thalheim, *Dependencies in relational database*, Teubner, 1991.

[3] J.H.Bekke. *Semantic Data Modeling*. Prentice Hall, 1992.

[4] A.Haeuer, G.Saake. *Datenbanken, Konzepte und Sprachen*. Internation Thomson Publishing Group, German, 1995.

[5] A.Binemann-Zdanowicz, Current Issues in Databases and Information Systems, *Proc. East-European Conference on Advances in Databases and Information Systems*, Prague, 2000, pp. 307-314.

[6] B.Pliuskuvienė, P.Adomėnas, The Aggregation of Relational Sets and the Algorithmic Dependencies of Data, *International Journal of WSEAS Transactions on Systems*, Issue 4, Volume 5, April 2006, p. 793-798.

[7] P.G.Adomėnas, A.Čiučelis, Data aggregation sets in adaptive data model, *Informatica*, Vol. 13, No 4, 2002, p. 381-392.

[8] P.G.Adomėnas, Data Functional Feature Sets and their Adaptability, *Proc. V East–European Conference on Advances in Databases and Information Systems*, Vilnius, 2001, pp. 131-140.

[9] B.Pakalniškytė, P.Adomėnas, The Identification Model of Data Structuries, *Proc. IV Conference on Information Technology,* Alytus, 2005, pp. 157-162.

[10] К.Дейт. *Введение в системы баз данных*, Москва, 1999.

[11] Г.Ханнсен, Д. Ханнсен. *Базы данных: раэработка управление*, Москва, 1999.