Conclusion

The designed device is a composite part of the automated information system for control, reporting and documenting the quantity of produced glass bottles, which is introduced in the factory for glass processing in town of Elena. The device enables the counting of empty bottles, discolored or of different coloring, of different form and size. It also may be successfully applied for counting of full bottles regardless the content and it level. These qualities of device provide its comparatively wide application in different branches of industry.

Bibliography

[Draganov, 2006] Draganov, V.D. , G.P. Toshkov, D.T. hristov. System for reading and documenting of the ready production quantity, Acta Universitatis Pontica, Volume 6, Number VII, 2006

[Solid Count, 2006] www.cornerstoneautosys.com/solidcount.htm - Solid Count

[Fast Counts, 2006] http://www.accusort.com/applications/counting.html - Fast Counts

[Patent 0050111724, 2005] United States Patent Application: 0050111724, A1, May 26, 2005. Method and apparatus for programmable zoned array coundter

[Bergmann, 1980] Bergmann H. Fotoelektrische Schalter, Radio fernsehen elektronik, 12/1980, pp. 769-770, in German

[Marinov, 1980] Marinov, Ju., E. Rangelova, V. Dimitrov. Technical diagnostics of radio-electronic systems and devices, Tehnika, Sofia, 1980, in Bulgarian

Authors' Information

Ventseslav Draganov – e-mail: v2draganov@abv.bg

Georgi Toshkov – e-mail: g_toshkov2006@abv.bg

Daniela Toshkova – e-mail: daniela_toshkova@abv.bg

Technical University of Varna, 1, Studentska Str, Varna, 9010

Dimcho Draganov – "Technotrade", Varna, 9000.

ABOUT METHODS OF MATHEMATICAL MODELLING IN THE DEVELOPMENT OF INFORMATION SYSTEMS

Maria Eremina

Abstract: This article describes the approach, which allows to develop information systems without taking into consideration details of physical storage of the relational model and type database management system. Described in terms of graph model, this approach allows to construct several algorithms, for example, for verification application domain. This theory was introduced into operation testing as a part of CASE-system METAS.

Keywords: information system, database, metadata, mathematical model, graph.

ACM Classification Keywords: H.2.4 Systems - Relational databases; D.2.2 Design Tools and Techniques - Computer-aided software engineering (CASE).

Introduction

The necessity in development of large information heterogeneous system is essentially increasing now. As is well known, *information system* is a complex of information recourses, technologies of getting and processing data, and keeping it in actual and consistent state. This definition is formulated from the point of user's view, but from the realization point of view information system is a complex combination management and technological solutions, hardware and software, and also information content.

The basic part of any information system is *database*, which complexity and capacity constantly increasing, leading to new difficulties in its operation. In particular, in many situations different nodes are controlling by different database management system. Such information systems are called *heterogeneous* [2].

Present paper is devoted to the analysis of special approach for describing database in large heterogeneous information systems. In particular, the publication demonstrates the method of working with such databases in natural for users terms, giving them possibility not to take into consideration the details of physical storage data in the relational model and type of database management system as well.

In order to provide the formulated above features the new approach is offered. It is found on additional metadata about information system, stored in the base. This approach is formally described in terms of graph model, that allows to formalize some algorithms of data processing, for example of verification problems.

The results of work are used in project of development CASE-system METAS, being a part of it.

METAS technology

CASE-system METAS (METAdata System) is a foundation for creation of system, which is controlled by metadata. This technology is destined for decreasing labour-intensiveness of the development of information systems and increasing their flexibility, scalability and adaptability in exploitation. The key feature of the discussion technology is usage interrelated metadata, which describes information enterprise system [7].

The majority of existing CASE-system generates code on some programming language according to definite specifications, which describe application domain. The main difference of METAS consists in usage of this metadata during its work. So the process of application's creation is reduced to writing necessary metadata.

The usage of metadata gives possibility of flexible application's adjustment and functionality. This creates the necessary prerequisites for development of "intellectual" system, which can automatically adapt to user's needs and to changing service conditions.

All metadata in METAS are divided into separate models (or levels), which of them describes the definite part of information system. Some model can describe the same part of information system, but from the different points of view. All models are interrelated, because one model can be based on another, being higher-level description of information system [8].

This article is devoted to the consideration of *logical model (LM)* of metadata. It directly founds on *physical model (PM)*, which describes all objects of database in information system: tables and their fields, tables relations, indexes and others. The detailed description of PM is presented in work [8].

There are some other models except listed above. Foe example the *presentational model* describes visual user's interface [6], the *reporting model* – queries and reports in information system and *security model* – right of users for execution operations with different objects in information system. This publication doesn't consider these models.

The main aims of logical model's creation

Some difficulties appear in usage of any relational databases. They are connected with characteristics of data keeping in relation tables and necessity of normalization. The structures of data, storing in database, often different from those, that user want to see. Let us consider this situation in more detailed way.

The first difficulty appears on the stage of application domain's analyze during designing database. The main method for this process is building the different entity-relationship diagrams or class diagrams in UML. In all these approaches the specificity of relational data model is necessary to take into consideration. The specialist, who dealing with analyses and designing, must work with physical tables and their relation and also must know the main principles of normalization. At the same time he must deeply understand the application domain of information system. This makes high demands of developer qualification.

Another difficulty appears in further work with database. In the process of SQL-query's construction it is necessary to remember all specificity of information storage in relational tables and to consider it.

For overcoming these difficulties the following approach was used in METAS. On design stage, as before, it is necessary to build diagram, which consists of entities and their relations. However in this case the term "*Entity*" means not one, but several related tables in database. This conception making entity closer to real life, simplifies the process of design.

The essential advantage of this approach is the possibility to work in natural for user terminology, abstracting from physical method of information storage. In particular, this slant provides to METAS independence from database type and support of heterogeneous systems.

For practical realization of discussed above approach the component called "*Logical model*" was developed. It is based on metadata of logical level. All other components that apply LM may work with database in logical level terms that strongly simplifies their development.

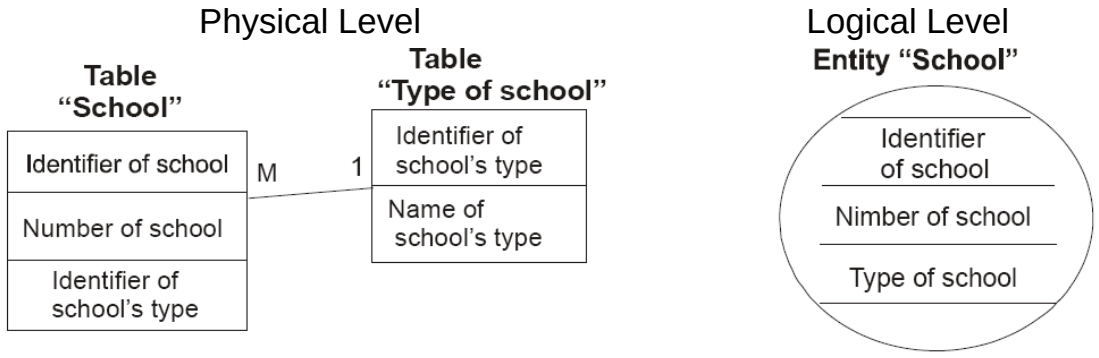
In this publication we will consider only main principles of LM. More detailed description may be found for example in [3, 4].

The main conception of logical model

The really existing objects in LM are presented by set of characteristics (attributes). These attributes can be physically stored in different tables of PM. So let as introduce some generalize construction called entity that is a set of these attributes.

For illustration we shell consider the fragment of database with information about schools: their numbers and types (for example lyceum, gymnasia). Obviously in relational database it is necessary to create two tables for presentation of this information. At that the first tables must contain foreign key to another. This tables form a physical level of metadata.

To organize the logical level for this fragment we may define the entity "*School*" with attributes "*Number*" and "*Type*". In this case "*Type*" becomes the same attribute like other. The only difference is a presence of mark that attribute's value is choosing from dictionary. Entity itself provides all work with this dictionary transparent for users. Thus every entity corresponds to set of physical tables. One of these tables is main, other are related with it as "1:M", in other word, they are *dictionaries* for it.



The main task of entity is to provide users possibility to work in term of application domain, without thinking about physical information storage. In other words, there is no necessity on logical level to take care about distribution of fields between physical tables. Now user may work in terms of entity, simply calling its name in combination with the name of attribute. Entity builds SQL query itself and either executes it itself (operations of inserting, deleting and updating information in database) or returns ready SQL expression.

Sometimes it is necessary to represent the main information about real object of some entity in the short form. For example let two entities are related as "1:M". In this case entity on side "M" has an attribute representing another entity (*foreign attribute*). In the discussing example entity "*Pupil*" has foreign attribute "*School*". It is not necessary to represent in this attribute all information about pupil's school, but only the main, which is enough for its identification. For example it may be school number. Such main information for entity is determined by the special *presentational expression*.

Presentational expression looks like SQL expression and may contain any attributes of corresponding entity. Any operation and function of SQL language are allowed. All attributes must be enclosed in square brackets, for example, for entity "*Pupil*" the following presentational expression is acceptable: *[Last name] + [First name] + [Middle name]*.

It was mentioned before that all entities are a set of attributes. There are some types of attributes listed below.

- *Own attribute*. It is any attribute from the main table of entity.

- *Dictionary attribute*. Any entity's dictionary has corresponding attribute. When we work with it we may choose necessary values from the list or add a new value in dictionary.
- *External attribute*. If the entity relates to another as "M:1", then it must contain additional attribute with information about this relation. For example entity "*Pupil*" has foreign attribute representing entity "*School*". The work with external attributes is similar to work with dictionary attributes, but its values may be chosen but not changed or added.
- *Key attribute*. In fact it is an ordinary own attribute. It contains a link to key field of entity's main table, so it unambiguously determines values of all the rest attributes. This term is introduced only for handy work.

We must not mix terms "presentational attribute" with "entity's attribute". The first one is not an attribute in the ordinary sense, because its value doesn't store in database. This is some value, which is calculated according to presentational expression and represent the main information about this entity. It is similar to attributes, because it is stored in entity together the rest attributes.

Let us introduce the concept of relation between entities. It is analogy with table's relation, but on logical level, in terms of application domain. For example entity "*School*" may be related with entity "*City*", which defines its location.

There are two basic relation types.

- 1:M. This relation corresponds to physical relation "1:M" between entities' main tables.
- M:M. This relation corresponds to physical relation "M:M", which is realized using intermediate table in relational database management system.

For any entity relation we may concretely define the count of entity on both sides. They have minimal and maximal number of entity, which may participate in relation. For example the following subtype for "1:M" are possible: «1 : 2..3», «0..1 : 0..1», «0..1 : 1» and other.

Building mathematical model

For building algorithms for METAS work it is necessary to formalize the description of using metadata for all models. Let us consider the formal definition of metadata systems for physical and logical levels in terms of graph model. As an example of using this model we will describe one of verification algorithm.

Physical level

Let us assume that *Fields* is some abstract set with elements of the same name. Let us also define the set *Tables* as reciprocal overlapping subsets of fields. For all tables the set of its fields we will name $F(t) \subset$ *Fields.*

Let us define the binary relation $R \subset Fields \times Tables$, and name it the set of relation between tables. The following notation will be used for it:

$$r \in R \Leftrightarrow r=(f,t),\ f \in Fields,\ t \in Tables.$$

Thus any relation $r \in R$ is built between some field $f \in Fields$, which belongs to the first table and the second table wholly (i.e. subset of a set *Fields)*. At the same time it is possible that relation connects the field of some table with the same table, i.e. $r \in R$, $r=(f,t)$, $t \in Tables$, $f \in F(t)$.

Logical level

Let us assume that *Attr* is some abstract set with elements of the same name. Let us divide all set *Attr* into groups, named entities. The set of entities called *Ent* must be *disjunctive*. This means from one side that all attributes in a set *Attr* must belong some entity, and from other side entities are not overlapped. If we name the set of attribute belonging entity $e \in Ent$ as $A(e) \subset Attr$, then the following expressions are correct: $\bigcap_{e \in Ent} A(e) = \varnothing,\ \bigcup_{e \in Ent} A(e) = Attr$.

All attributes in any entity we divide into three non-overlapping set. For any $e \in Ent$ we will denote *the set of own entity's attributes* as $O(e) \subset A(e) \subset Attr$, *the set of dictionary attributes* as $S(e) \subset A(e) \subset Attr$ and *the set of external attributes* as $V(e) \subset A(e) \subset Attr$. So conditions of non-overlapping of these sets and mandatory including of any attribute into one of these sets both take place:

$$\forall e \in Ent\ \ O(e) \cap S(e) = \varnothing,\ V(e) \cap S(e) = \varnothing,\ O(e) \cap V(e) = \varnothing,$$

$$\forall e \in Ent \;\; \forall a \in A(e) \Rightarrow a \in O(e) \vee a \in S(e) \vee a \in V(e).$$

The subset of entity's attributes including in presentational attribute we name P(e). It will contain attributes from any set O(e), S(e), V(e). So,

$$\forall e \in Ent \;\; P(e) \subset A(e),\ P(e) \neq \varnothing.$$

Strict definition of concept entity follows from the above theses. The *entity* is define by five

$$e = (A, O, S, V, P),$$

where all of the last four sets are subsets of the set A with described above properties.

At all pictures below we will use following graphical symbol for introduced concept. We will denote as circle any member of set *Attr*, and any member of set *Ent* as big circle. If attribute belongs some entity then it will be on corresponding circle. The attribute's color of filling means, to which set O(e), S(e) or V(e) it belongs. Corresponding colors for these sets are white, gray and black. The set of attributes including into presentational attribute we will outline dotted line.

There are many different means to define entities, which satisfy the condition of disjointness, if we have fixed the set of attributes. For all entities there are many different ways to define sets O(e), S(e), V(e), P(e). It is not conflict with discussed model, but in practice the model must correspond to real application domain. So all divisions of attributes into entities and divisions attributes by types are unambiguously determined. That's why we can think, that all divisions are defined unambiguously in our model.

Let us name as Y the set of all external attributes of all entities. Then $Y = \bigcup_{e \in Ent} V(e)$. Let us define *W* as single-valued transformation, which any $v \in Y$ associates with some $e \in Ent$ by means of one-for-one relation. Another word

$$w \in W \Leftrightarrow w=(v,e),\ v \in V(e_1),\ e, e_1 \in Ent.$$

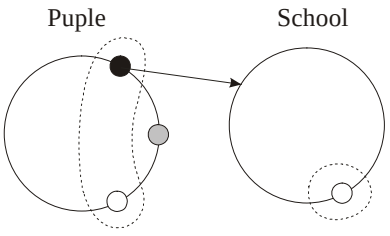
Thus any relation $w \in W$ connects one of external attributes of the first entity with the second entity. At the same time it is possible that relation connects the attribute of some entity with the same entity, i.e $w \in W$, $w=(v,e)$, $e \in Ent$, $v \in V(e)$. Graphically we will denote such relation as directed arc from external attribute to entity.

The second binary relation $M \subset Ent \times Ent$ we will name as set of relations "M:M" ("many-to-many")between entities. Graphically such relation we will denote as undirected arc between entities. Since there are many relation "M:M" between two entities in application domain, our graph will be of *multigraph* type or a *graph with parallel edges* [9]. For edges identification we will mark any such edge by unique name. This name will be used further in realization, for example in building the tree of objects serving for user's navigation inside the system [6]. Hence any relation "M:M" represents triplet (e_1,e_2,*name*), where e_1,e_2 interrelating entities, and *name* is unique relation name. We will use the following notation:

$$m \in M \Leftrightarrow m=(e_1,e_2,\text{name}),\ e_1,e_2 \in Ent,\ \text{name} \in String.$$

Thus any relation $m \in M$ connects two entities wholly.

Now we will describe the example. Let us suppose that the entity "*School*" has own attribute "*Number*", and the entity "*Pupil*" has attributes "*Name*" (own attribute), "*Sex*" (dictionary attribute) and "*Pupil's school*" (external attribute). These entities are related as "1:M" by means of the attribute "*Pupil's school*".
For "*Pupil*" as presentational attribute we may use expression "*Name*" + " " + "*Pupil's school*", and for "*School*" – its attribute "*Number*."



In this example the set Attr consists of four elements and the set Ent consists of two elements. Let us denote the entity "Pupil" as $e_1 \in Ent$, and the entity "School" as $e_2 \in Ent$. Then $A(e_1)$ consists of three elements and $A(e_2)$ consists of one element. The sets $O(e_1)$, $S(e_1)$, $V(e_1)$ contain one at a time attributes, and set $P(e_1)$ contains two attributes: "Name" and "Pupil's school". For entity "School" sets $S(e_2)$ and $V(e_2)$ are empty, and sets $O(e_1)$ and $P(e_1)$ contain only one attribute "Number".

Triplet L=(*Ent*, *W*, *M*) is called complete graph of application domain. Under the term "complete" we mean, that the model contains all information about application domain. Further we will describe reductive graphs, which are

building from complete graph using some transformation. For example if in a task we don't interested in relation "M:M", then in complete graph we may replace the set M on empty set. This reductive graph will be more suitable for solving this task.

As it follows from the above, the complete graph is a *mixed* graph (that is it has both directed and undirected arcs) with parallel edges and loops.

In terms of discussed graph models we may build algorithms for solution different tasks in CASE-system METAS. For example there are algorithms for separation subschema, reengineering and migration of data [1]. In the following part we describe another example using this model.

Using graph model for verification metadata

Metadata in METAS is using for generation of all functionality for information systems. If developer mistakes in building metadata, he may not observe this, because the system of metadata may be very complex. That's way it is necessary to realize algorithms, which analyze structural static features of model and discover some types of mistakes.

As it was mention above, reductive graphs of application domain may be often used for description of some algorithms in terms of graph model. They may be built from complete graph using some transformation, which is specific for every algorithm. For example for some of them we must remove all relation "M:M" from complete graph, in other case we must remove all attributes.

Let us describe the example of algorithm, which checks the application domain on availability of cycles.

Under the term "*cycle*" we mean the following. Let there are two entities and both of them have external attribute to each other. Let both of these attributes be included in presentational attribute of their entity. In this case when we try to ask information about the first entity, we must know the value of presentational attribute of other entity. But it contains external attribute, that's why we must ask information about presentational attribute of the first entity. As a result we have circular reference and the system will not be able to execute this query. Obviously such cycle may include more than two entities and be less evident. That's why we must exclude such cycles on the design stage.

Let consider the formal model of this situation.

In complete graph of application domain we don't interested in such objects as relation "M:M", own and dictionary attributes. That's why the set *Attr* has only foreign attributes in reduced graph. The sets of entities and their presentational attributes are not changed. The set W (relations "1:M") is not changed too. Although condition $P(e)\neq\varnothing$ in reduced graph is broken , but it is not essential for our task.

As a result we get a graph $L_1=(Ent_1,W,\varnothing)$, где $Ent_1=\{e_i\}$, $e_i=(Attr(e_i) \setminus O(e_i) \setminus S(e_i), \varnothing, \varnothing, V(e_i), P(e_i))$, i.e.

$L_1=(Ent_1,W,\varnothing)$, $Ent_1=\{e_i\}$, $e_i=(V(e_i), \varnothing, \varnothing, V(e_i), P(e_i))$.

The formally definition of cycle is introduced below.

Let $RV(e_1,e_2)$ is set of attributes, which satisfies the following expression:

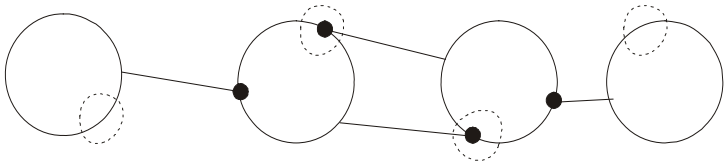
$$\forall a\in RV(e_1,e_2),\ a\in V(e_1),\ a\in P(e_1),\ \exists w_1\in W,\ w=(a,e_2),\ e_1,e_2\in Ent.$$

We suppose that the following condition is correct:

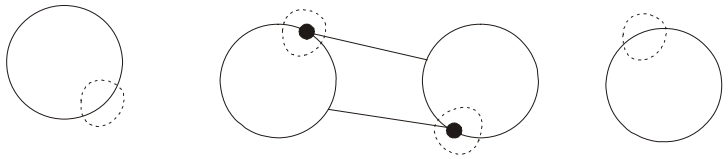
$$\exists a_1\in RV(e_1,e_2),\ \exists a_2\in RV(e_1,e_2),\ \dots,\ \exists a_{n-1}\in RV(e_{n-1},e_n),\ \exists a_n\in RV(e_n,e_1),\ e_i\in Ent,\ i=1..n.$$

In this case we will say that graph of application domain has a cycle.

The example of graph with cycle is demonstrated in the following picture.



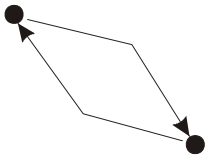
Obviously if external attribute is not a part of presentational attribute, then it doesn't affect on cycle's availability. That's why we may reduce our graph more when we remove all such attributes. Also it is necessary to remove all corresponding relation "1:M". We will have the following graph:



$L_2=(Ent_2,W_2,\varnothing)$, $Ent_2=\{e_i\}$, $W_2=(v_{ij},e)$, $e\in Ent$, $e_i=(\{v_{ij}\}, \varnothing, \varnothing, \{v_{ij}\}, P(e_i))$, $v_{ij}\in V(e_i)$, $v_{ij}\in P(e_i)$.

Further we may reduce this graph to classical directed graph. For this purpose we consider that all external attributes are graph's nodes, and edges from one node to another are contained in graph if only external attribute, corresponding the first node, is related with entity, which contains external attribute, corresponding the second node.

$G=(GV,GE)$, $GV=\{v_i\}$, $v_i\in Attr_2$, $GD=\{(v_i,v_j)\}$, $\forall v_j\in A_2(e)$, $w=(v_i,e)\in W_2$, $e\in Ent_2$



Here $Attr_2$ is a set of all attributes of graph L_2, and $A_2(e)$, $e\in Ent_2$ is its subset for entity e.

Further we may use any standard algorithm of finding cycles in graph.

Conclusion

Publication is devoted to the approach to the description of database, which allows to work in naturally for user terms and don't take into consideration details of physical storage of the relational model and type of database management system. Formally described in terms of graph model, this approach allows to develop several algorithms, for example, for verification of application domain.

The developed theory has a practical realization as a part of CASE-system METAS. Based on the last one the following information systems were developed: "The educational system of Perm region" and "Interdepartmental information system of personified registration of children in Perm region". These systems were successfully introduced into operation testing.

The results of paper permit generalization on distributed database case, when all information may be divided between different network nodes. This provides guidelines for future work [5].

Bibliography

[1] D.A. Borisova. The component of restructurization of CASE-system METAS. In: Mathematics of program systems. Perm: PSU , 2003. (in Russian)

[2] C. J. Date. An introduction to database system. Eighth Edition. Addison-Wesley, 2003.

[3] M. E. Eremina. Generation of SQL-expressions based on metadata. In: Mathematics of program systems. Perm: PSU , 2003. (in Russian)

[4] M. E. Eremina. Using metadata for generation of SQL-queries to database. In: Modern problems of mechanics and applied mathematics. Voronezh: VSU, 2004. (in Russian)

[5] M. E. Eremina. Realization of queries in distributed heterogeneous systems, controlled by metadata. In: Youth. Education. Economics. Part 4. Yaroslavl: Remder, 2004. (in Russian)

[6] E. Yu. Kudelko. Generation and adjustment of screen forms based on metadata. In: Mathematics of program systems. Perm: PSU, 2003. (in Russian)

[7] L. N. Lyadova, S.A. Ryzhkov. CASE-technology METAS. In: Mathematics of program systems. Perm: PSU, 2003. (in Russian)

[8] S.A. Ryzhkov. Conception of metadata in the development of information systems. In: Mathematics of program systems. Perm: PSU, 2002. (in Russian)

[9] M. Svami, K. Thulasiraman. Graphs, networks and algorithms. Mir, 1984. (in Russian)

Authors' Information

Maria Evgenevna Eremina – Perm State University; 614990, Russia, Perm Bukireva st. 15; e-mail: erm6@mail.ru