# *Meta-database for the information systems development platform*

Yury Rogozov, Alexander Sviridov, Sergey Kucheov
System analysis and Telecommunications dept.
Taganrog Institute of Technology, Southern Federal University
Taganrog, Russian Federation
rogozov@tsure.ru, sviridov@tsure.ru s.a.kucherov@gmail.com

***Abstract* — For today conditions of using information systems presuppose availability of tools, by which developers can quickly adjust their products in accordance with the updated requirements. In connection with this arises the problem of automating the information systems development and reducing the share of labor expenditures for writing the source code. The paper presents a solution for this problem, which is based on the modernization of data and knowledge storing technologies. The concept of the meta-database is proposed. The requirements for the meta-database are formulated. The formal and the graphical model are given; features of the meta-database implementation with using relational technology are described. The questions of constructing and utilization a development platform based on the meta-database are examined**

***Keywords – metamodel; metamodelling; meta-database; information systems development platform; automation***

## I. INTRODUCTION

The problem of automating the information systems development and reducing the share of labor expenditures for writing the source code actively discussed in scientific and research works recently. This is evidenced by the emergence and development of Model-driven-architecture (MDA) [1]. It is worth noting that each author sees the problem through the prism of their own knowledge. Considerable imprint on the vision of the problem leave and subject area in which the researchers works. Thus, for example, experts in the field of requirements management [2,3] trying to automate the process of user requirements fixation through specialized tools and use them for generating of application source code. Specialists in the field of software design and development [4] concentrate their efforts on creating of universal abstract application model, and then realize information system through configuring and customization of this abstract model.

Analysis of these papers shows that common idea of authors consist in an attempt to capture the acquirements about specific domain in the form of some models, that underlie the static structure of a development platform.

Due to this realization created models have generalizing character and serve as a means of constructing models specific information system, i.e. they are situated at a higher level of abstraction - a meta-level. Therefore, they can be called metamodels. There is some examples of the meta-model's definition:

- Metamodel is a model that defines the language for expressing a model [5]
- Metamodel is a model of model [6]
- In MDSD, a metamodel is a “model of the modeling language.” [7]
- Metamodel describes the possible structure of models written in that language, i.e., the “constructs of the language and their relationships, as well as constraints and modeling rules” [8]

The concept of metamodeling can also be correlated with data storage technologies. This will open up new possibilities for design and development of information systems in general.

In this paper the meta-database – the tool of storing description of the subject domain at the different levels of detailing – from domain metamodel to user data are proposed. In the first section of this paper we describe features of development environments based on meta-database, in the second section – meta-database concept. The third and fourth sections of this article describe models of meta-database.

## II. META-DATABASE BASED DEVELOPMENT PLATFORM

From the viewpoint of information systems development, the metamodel is primarily the knowledge ontology of the developers about particular domain with its activities and tasks. Also, the metamodel is a tool for describing information systems and data models.

Today the process of creating and using a development environment consists of the following steps:

1. Formation of the metamodel based on the acquired experience and knowledge. Metamodel, as a rule, is represented as a set of classes, and by this reasons it presents the static structure of the environment.
2. Static structure of classes is supplemented by components that will allow to work with the metamodel. Such components can be tools for interface specification, configuration metamodel, input/output data, etc.
3. Metamodel, supplemented by components, implemented into the finished form - development environment.
4. The metamodel is supplemented by description of specific business processes by means of the appropriate

components, configuration of this metamodel occurs. The complete product (information system) is formed in the result.

Visually, the process is as follows (fig. 1):

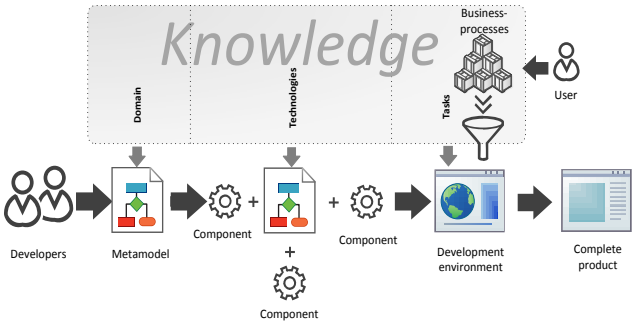


Figure 1. Metamodel-based development environment building process

Approaches that use a metamodel as a static structure of the development environment have capabilities to reducing the share of labor expenditures for writing the application source code. However, fig. 1 shows that the universality of these environments is limited by the range of problems from a particular subject domain, for example in the field of corporate web applications [4].

The problem of limitation this approach is fixing the metamodel as a static structure of a development environment. Creating more versatile metamodel - the task that is solved for today only in the development environment for object-oriented programming languages. Object-oriented paradigm allows to specify the elements of subject domain on abstract level, but it does not solve problem that is noted at the beginning of this paper - writing source code in such environments is the base component of the development process. Multiple approaches, such as code reuse and patterns are also oriented on source code, and it is not consistent with the direction of research outlined in the beginning of the paper.

In our opinion, the solution consists in empowering development platform by adding abilities to create new or modify existing therein metamodels. For this meta-model should be separated from the static structure of the development environment and is represented as object with complex structure stored in permanent memory (for example, like records in database), and the environment itself - have a means to describe and store metamodels (for example, a set of mechanisms for manipulating data and their structure). Such database, giving developers tools for describing metamodels of any structure, similarly to the metamodel can be called the *meta-database*. For the first time this term was formulated by Cheng Hsu et al. in 1990 [9], in those work were designated common meta-database properties, which is the integration into one the following aspects:

- knowledge about the company or the information system (i.e., operating, control, and decision);
- All models obtained at each stage of the development life cycle, ranging from multistage analysis to design and implementation;
- Data obtained in the process of using information system.

In our understanding, in terms of designated problem of automating the information systems developing process, *meta-database* is a tool that allows to represent a permanent memory descriptions of subject domains at various levels of detailing (the various meta-levels) - from metamodels to configuration of information system and user data.

In using the meta-database there could appears more abstract development tool - development platform. Development platform is a mechanism that reflects metamodels into the meta-database and thus it can allow to produce a less abstract development environments. These environments, in turn, give a complete product via concretization of metamodel to the level of current tasks (business processes) (fig. 2).

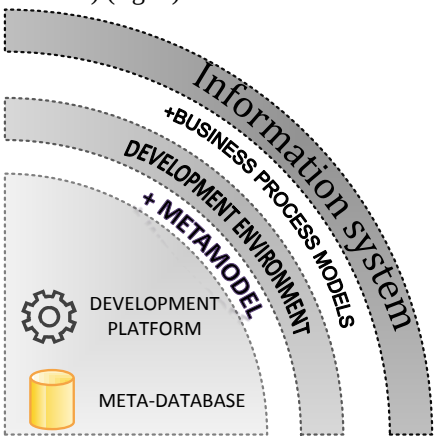


Figure 2. The concept of meta-database based development environment

Taking into account that the metamodel can be changed dynamically, without alteration the information system development platform, the process of building a development environment should look like this:

1. Existing meta-database based development platform that has a predefined set of basic components, if necessary, depending on the technologies knowledge complemented by missing components;
2. Domain experts (platform's users) placed metamodel into the development platform's meta-database, which is detailed to the level of selected implementation technologies.
3. The platform is configured to the level of the development environment which is oriented to concrete technology and a certain subject domain metamodel.
4. Users put the knowledge about current tasks in the form of business processes models into the development environment, and then they can try complete product (information system).

Visually, this process of creating a development environment may be shown as follows (fig. 3).

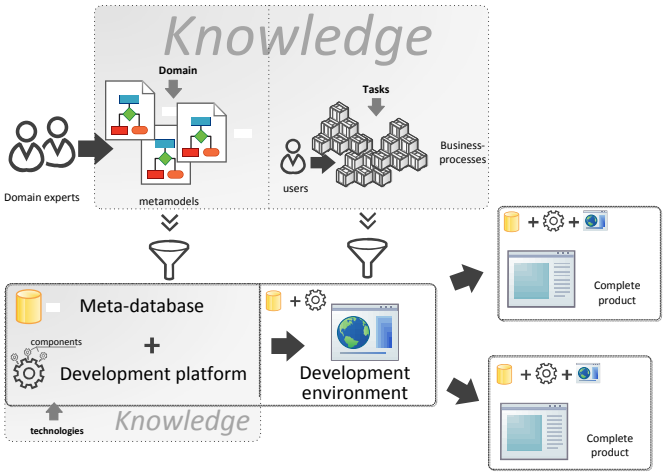


Figure 3. Using of meta-database to construct a development environment

Fig. 3 shows that the development platform can exist without any metamodel (in fact, the development platform in a static structure is described by the meta-metamodel, i.e., metamodel of higher level of abstraction). Knowledge of the developers from a particular subject domain, represented in the form of metamodel, are sort of configuration file for the platform. There can be several of such configuration files and in one platform we may have multiple development environments. In this way the development environment is constructed in accordance with the principles of meta-design and socio-technical environments [10,11].

This statement allows us to call the fig. 3 a meta-design model and to formulate the meta-database concept.

.

## III. THE META-DATABASE CONCEPT

So, the meta-database is a tool that allows a unified strict way to represent in permanent memory knowledge about subject domains and results of operation their business processes. These two big aspects are divided into:

- metamodels of information system' classes for different range of tasks in any domain;
- logics of subject domain business processes;
- structure of the information system that is described in terms of the metamodel;
- user data collected as a result of business processes.

Thus unified representation may be entities (the objects with some structure information about which should be stored), attributes (characteristics of an entity) and relations (between entities), which may be a special kind of entity. Entities are relevant to the classes, entity instances - to the objects of classes, etc.

Such properties set up the number of requirements to the meta-database:

5. Existence of manipulating mechanisms not only for data but also for their structure. Since on the upper levels of detailing the domain structure is described primarily, and the data appears at the lowest level;
6. Standard format for querying the meta-database. Since the meta-database is integrated with a development platform, then regardless of its content should be standard ways to access data and their structure;
7. Independence from the structure and composition of stored information. As can be seen from fig. 3, the meta-database is constantly updated by some information; structure of whose can not be determined in advance that in the case of depending the meta-database structure on stored information will lead to malfunction of platform as a whole.

Implementation of these requirements depends on separation methods, which will be applied to divide storing instruments on the meta-levels. For example, in database technology, there is one meta-level - metadata. This meta-level strictly define the access path to the data and data structure. Often, metadata is a declarative tool that is supported on DBMS level. As a result, the metadata becomes static element of database and its changing leads to redesign software that uses data from database.

Meta-database should provide means of storing metadata, which can take an arbitrary structure, depending on the metamodel. By this reason, structure of the meta-levels described above is unusable - it has to be modified. To ensure the creation of meta-database the metadata, which is a tool for fixing the knowledge about data, should be descriptive and be contained inside the database (fig. 4).

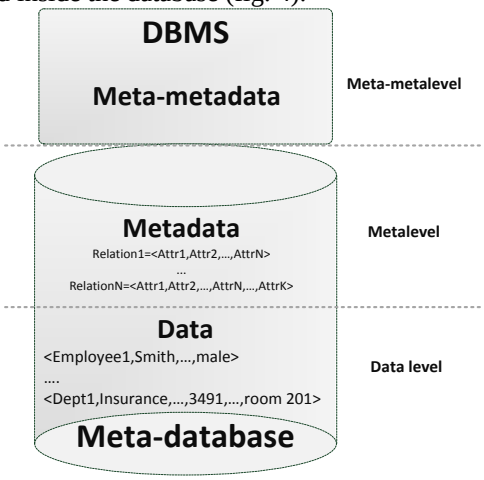


Figure 4. Meta-database concept

Fig. 4 shows that in process of using meta-database appear two meta-levels:

- Metalevel. Metadata - a means of representing knowledge about the database data. They are descriptive and are stored in the meta-database in a similar way with the data.
- Meta-metalevel. Meta-metadata - a means of describing the various metadata. It is a declarative tool that contains elements of the language for describing arbitrary metadata.

We can assert that the meta-database concept showed in fig. 3 is applicable for creating the information systems development platform since the metamodel, placed in the

permanent memory (in the meta-database ), it is nothing else than the metadata, and configuration of metamodel reflecting particular application – the data in the database. In this case, there is a meta-metalevel – meta-metadata reflecting the meta-database structure and defining thereby a means of describing the various metamodels.

## IV. THE META-DATABASE MODEL

Now turn to a more detailed presentation of the meta-database - its model. Here the key role is played by the strict representation in permanent memory of such aspects as domain knowledge and results of business processes operation. This strictness can be achieved by creating a formal meta-database representation model at the level of meta-metadata structure. Such formal representation will also allow to define accessing and manipulating methods for structure of stored metamodels and data.

As we told in second section of this paper, representation of any aspect from subject domain can be made with entities, attributes and relationships. Therefore, a formal tool for describing meta-database model may be of Codd's relational algebra [12], in which exist most similar concepts: entity - relation, attribute - attribute, connection - equal values of key attributes for two entities.

On the basis of the proposed concept, meta-database should combine in itself the lower level of representation - data of domain, and the first metalevel - metadata that store knowledge about data. When we create a formal model, this should be a fundamental requirement.

After analyzing several metamodels, described by the authors [4,13,14,15], we can define a unified set of their components:

8. Metamodel alphabet - entities reflecting the main aspects of information system produced by a specific metamodel. For example, use cases, data elements, reports, business logic, etc.
9. Entity attributes, adding of which leads to specification of metamodel.
10. Relationships between entities that define the structure of metamodel.
11. Hierarchy of entities, determining the order and direction of relationships.
12. Entity instances, reflecting the certain components of finished information systems.
13. Attribute values of entities by which metamodel is configured to the level of complete product.
14. Links between instances that represent structure of finished information system.

On the base of this classification meta-database model can be defined in terms of Codd's algebra by following system of relations:

$$M \equiv \langle E, A, S, L, R, V \rangle$$

where:

E – relation «Entities»

$$E = \{e_i\}$$

A – component «Attributes»

$$A = \{< a_i, t_j >\},\ t_i \in T$$

S – component «Entity structure»

$$S = (E, A, F), \pi_1(F) = E', E' \subseteq E,\ \pi_2(F) = A', A' \subseteq A$$

L – component «Relationship structure»

$$L = (E, A, F),\ \pi_1(F) = E', E' \subseteq E, \pi_2(F) = E'', E'' \subseteq E,\ \pi_3(F) = A', A' \subseteq A$$

R – component «Entity instances»

$$R = (E, I, F),\ \pi_1(F) = E', E' \subseteq E,\ \pi_2(F) = I$$

V – component «Attribute values»

$$V_{t_i} = (I, A, D, F),\ \pi_1(F) = I'', I'' \subseteq I,$$
$$\pi_2(F) = A', A' \subseteq A, \pi_3(F) = D'', D'' \subseteq D', D' \subseteq C_i$$

To prove the applicability of this model, consider the Ecore metamodel – the base for Eclipse Modelling Framework [14].

For example, we take two objects form Ecore: ENamedEement and ETypedElement, which is connected by hierarchical relationship without inherited characteristics (fig. 5).

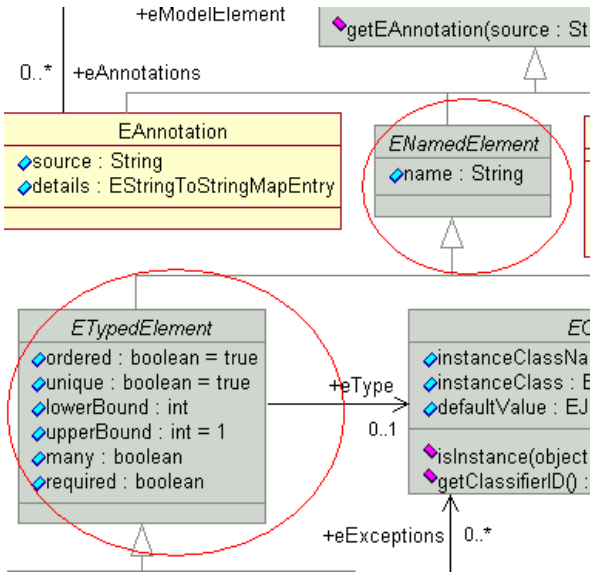


Figure 5. Ecore metamodel elements

Selecting entities:

E = { ENamedEement, ETypedElement}

Selecting the attributes and matching them with data types:

A = {<name,t3>,<ordered,t1>,<unique,t1>,<lowerBound,t2>, <upperBound,t2>,<many,t1>,<required,t1>}, где t1=’boolean’, t2=’integer’, t3=’string’

Matching the attributes and entities, defining the structure of entities:

S = {<ENamedEement,name>,<ETypedElement, ordered>,

<ETypedElement, unique>,<ETypedElement, lowerBound>, <ETypedElement, upperBound>,<ETypedElement, many>,

<ETypedElement, required>}.

Defining link structure:

L = {<ENamedEement, ETypedElement,0>}

Thereby, we described two objects of Ecore metamodel [14] by using of proposed formal model.

Let us suppose that in process of metamodel configuration appears certain instance of ENamedEement class and related to it instance of ETypedElement class. In our formal model this will be represented by next way:

Fixing of class instances:

R = {<ENamedEement,instanceID1>,<ETypedElement,instanceID2>}

Filling in values of attributes:

V = {<instanceID1, name, MyObject>,<instanceID2,ordered,true>,

<instanceID2,unique,true>,<instanceID2,lowerBound,-1>,

<instanceID2,upperBound,1><instanceID2,many,false>,<instanceID2,required,true>}

Fixing of relationship between class instances:

V = {<instanceID1, 0, instanceID2>}

Thereby, we showed creating of two class instances that appear in the process of metamodel configuring by using of proposed formal model. Such class instances could represent certain aspects of information systems or data. The given example and analysis of other known metamodels shows applicability of proposed formal model for its unified representation. More detailed representation of meta-database formal model and its utilization is described in [16].

## V. THE META-DATABASE REALIZATION AND UTILIZATION

Guided by the fact that relations in Codd's algebra may be mapped in corresponding database tables and their attributes - in the fields of these tables, we implemented meta-database on the base of relational DBMS.

The principal difference between a simple relational database and meta-database is showed by three-dimensional representation in basis "entity-attribute-value" (fig. 6,7).

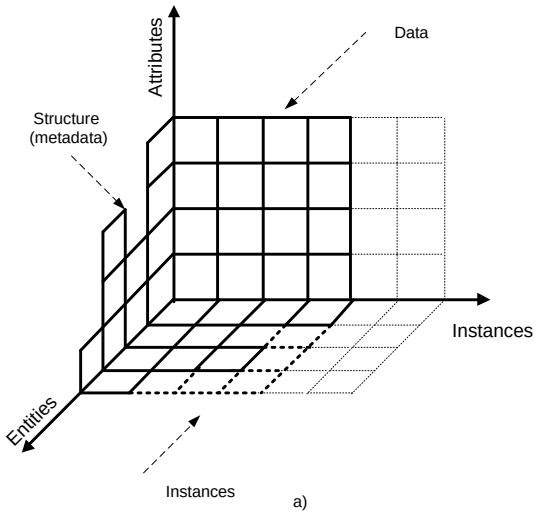


Figure 6. three-dimensional representation of a relational database in basis "entity-attribute-value"

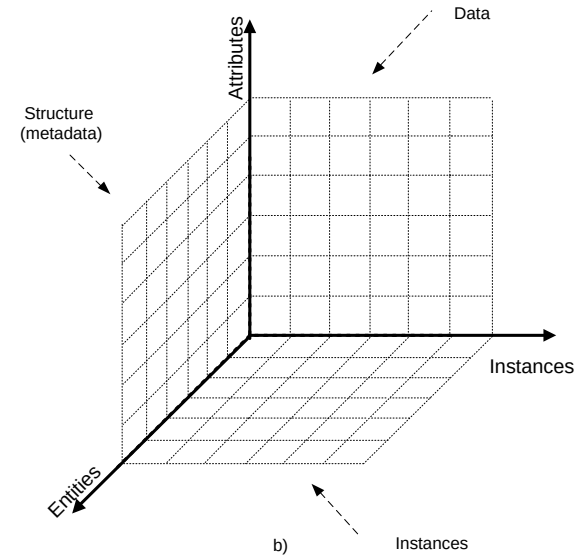


Figure 7. three-dimensional representation of a meta-database in basis "entity-attribute-value"

Fig. 7 shows that describing domain metamodel a-priori is not necessary for the meta-database – meta-database is ready for utilization even if it has no one entity and no one attribute. Using of a relational database (Figure 6) require pre-determine the structure of metadata and by this reason it is not suitable for resolving problem outlined in this paper.

We make some assumptions before giving the meta-metadata structure (structure of meta-database tables):

15. Metadata can be grouped and be represented by following corresponding tables:

- Hierarchical directory, which includes all metadata, barring "entity-entity" and "entity-attribute" relationships. Metadata stored in this directory is applicable for representing entities and attributes with the same efficiency, and therefore they may be represented by a single table.
- Attributes and relationships directory, which entries will refer to ascribable elements of hierarchical directory. Implemented as a separate table.
- Entity instances directory that refers to the elements of a hierarchical directory. Implemented as a separate table.
- Attribute values may be grouped by types into the similar tables for the purposes of efficiency. Ownership of these values will be defined by references to the metadata tables – name of field, which stores values, is not an attribute name. Attribute values make up the appropriate subschema of tables.

Whereas these assumptions, we represent the structure of meta-database as follows (fig. 8):

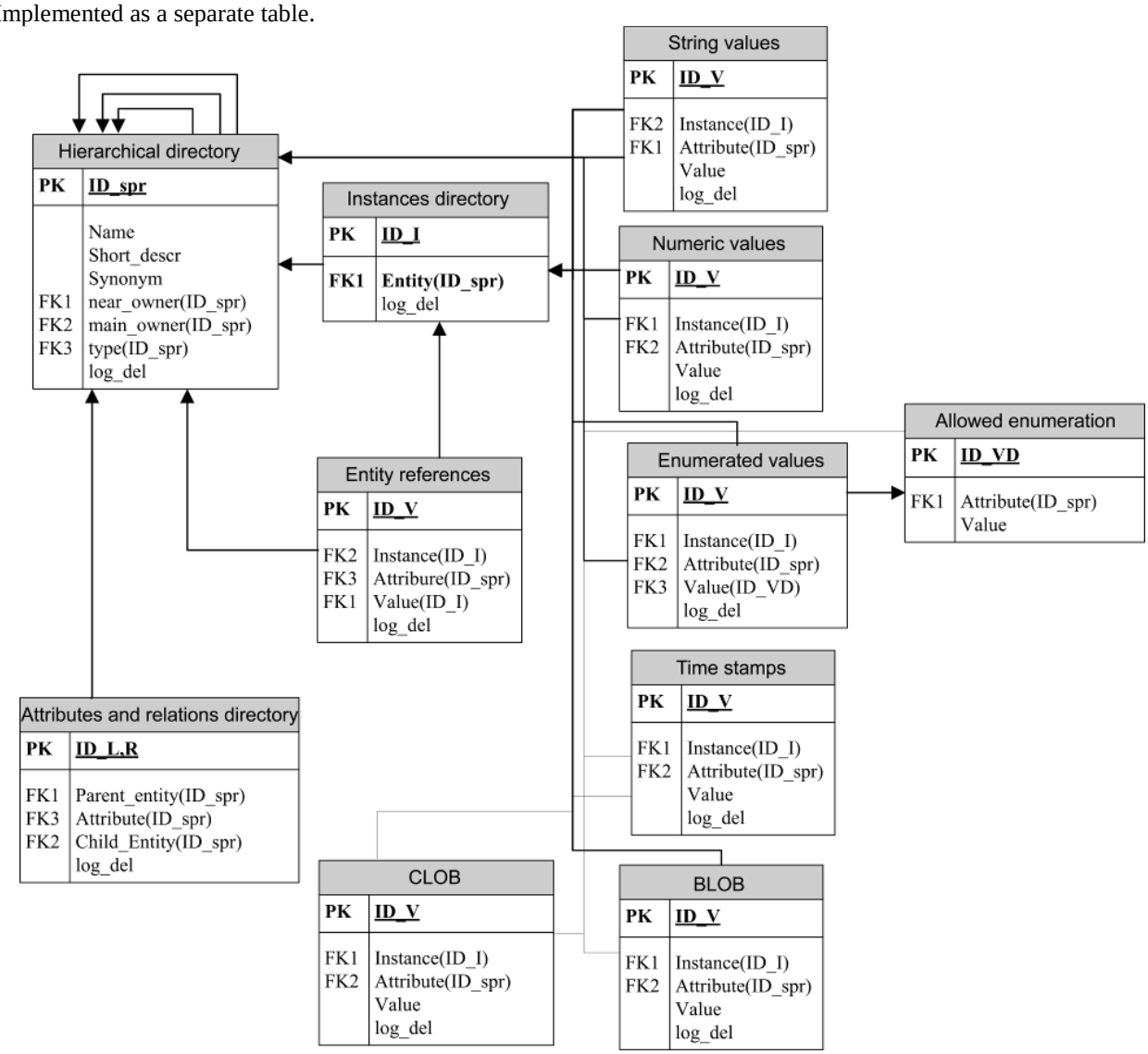


Figure 8. Logical model of meta-database

Such structure of data and metadata tables allows to implement all principles underlying the definition of meta-database, and it is satisfy the meta-database conception presented in fig. 4. In this case meta-metadata, which defining the rules of making metamodels, is the structure of meta-database logical model.

Summing up, we can formulate the following features meta-database:

- No need for a-priori description of stored data structure;
- Metadata is a descriptive. As a declarative tool we use meta-metadata;
- Independency of accessing data mechanisms from data structure.

Implementation of the meta-database is called SiDB (Structire-Independent Database) and described in detail in [17].

No less important question related to the meta-database is its utilization. For today the vast majority information systems are based on object-oriented paradigm. As a result, interact programs with relational databases is difficult - representation of application objects do not always correspond to representation of these objects in a relational database tables. To resolve these mismatch problems developers apply technology of object-relational mapping (Object-Relational Mapping). Actually, ORM is a layer between application and database that provides one mapping of data between two models (fig. 9).

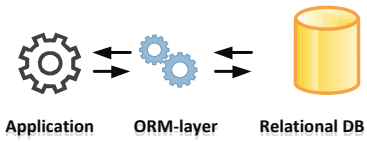


Figure 9. Using of ORM-layer

The main disadvantage of using ORM-layer is considered a loss of productivity. From this perspective, addition of a meta-level over the metadata would lead to even greater costs on the conversion and searching of data. However, using of metadata as a descriptive tool, which is not regulates strict access paths to data and permits access to attribute values directly in aggregate with a fixed structure of tables allows to replace ORM-layer by SQL-queries generator (fig. 10). By means of such generator any data from meta-database can be represented in the required form for particular application.

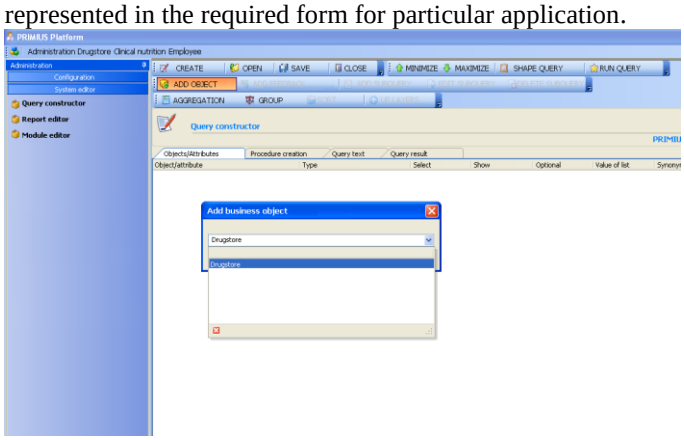


Figure 10. SQL-query generator interface

Thereby, building up another layer, that involved in the process of working with data does not occur (fig. 11).

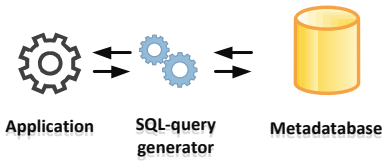


Figure 11. Scheme of interaction between the application and the meta-database

Lack of cost on data conversion, together with the possibility of using compiled stored procedures to access frequently requested data allow to achieve an enough level of performance, which is less than 10% inferior to relational databases created on the classical technology [18,19].

The effectiveness of the development platform is corroborated by its commercial use for automation of several social protection institutions in the Southern Federal District of Russia. This platform is called PRIMIUS (fig. 12) and in the working process is based on interpretation of the metamodel, which is created in terms of knowledge gained during many years of work in this area. The metamodel is implemented as entries in the metadata directories of SiDB.

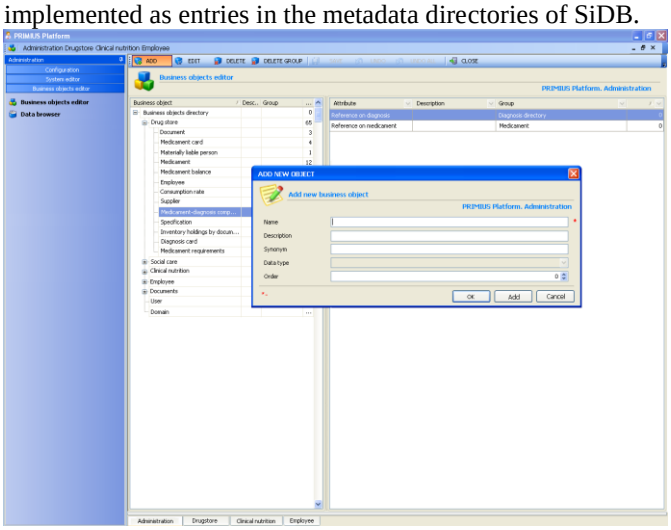

Figure 12. Interface of development platform PRIMIUS

Information systems of any complexity is developing by means of platform «PRIMIUS», both for individual departments, where the volume of stored data is relatively small, and for organizations with a wide range of workstations and complex logical structure of stored data.

## Conclusion

Proposed in this paper the meta-database is an effective solution, with close to a relational database performance measures. Meta-database has the following beneficial properties:

- storing of metamodel, information system model and user data simultaneously;
- absence of restrictions on the stored data structures;
- storing is not only the elements of models and relationships between them, but and the properties of these relationships;
- no need for a-priori description of the stored data structure;
- presence of means for formal description and data manipulation.

Due to these properties using of the meta-database allows to solve the next important problems:

16. Reducing the share of labor expenditures for writing the source code. Meta-database allows to build a development platform based on the paradigm of metamodeling, That allows the developers to generate source code by using of the models stored in meta-database.
17. Boundedness of development platforms which use metamodels as an aspect of the static structure. Built on the base of meta-database information systems development platform can contain a variety of metamodels, and permits their changing without loss of efficiency.

Current results of our work allow to form a basis for further research, among which the following main areas:

- Meta-database optimization and searching for its high-performance implementations;
- Creating tools for formalization description of subject domain and information system;
- Creating automate tools for the information systems development process;
- Creating tools for supporting variability of requirements and information systems.

## REFERENCES


[1] Colin Atkinson and Thomas Kühne. Model-driven development: a metamodeling foundation. IEEE Software, 20(5):36–41, IEEE Computer Society, 2003.

[2] Carlos Rossi, Antonio Guevara, Manuel Enciso, José Luis Caro, Angel Mora. A Tool for user-guided database application development - Automatic Design of XML Models using CBD. Proceedings of the Fifth International Conference on Software and Data Technologies, ICSOFT 2010, Volume 2, p.195-201.

[3] Xufeng (Danny) Liang, Christian Kop, Athula Ginige, Heinrich C. Mayr: Turning concepts into reality - Bridging Requirement Engineering and Model-Driven Generation of Web Applications. ICSOFT 2007, Proceedings of the Second International Conference on Software and Data Technologies, Volume ISDM/EHST/DC p.109-116

[4] Athula Ginige. Meta-design paradigm based approach for iterative rapid development of enterprise web applications. Proceedings of the Fifth International Conference on Software and Data Technologies, ICSOFT 2010, Volume 2, p.337-343.

[5] Meta-Object Facility (MOF) standard. http://www.omg.org/mof/

[6] Gilles Dodinet, Michel Zam and Geneviève Jomier. Coevolutive meta-execution support - Towards a Design and Execution Continuum. Proceedings of the Fifth International Conference on Software and Data Technologies, ICSOFT 2010, Volume 2, p.143-151.

[7] Stephen J. Mellor, Kendall Scott, Axel Uhl, and Dirk Weise. MDA Distilled: Principles of Model-Driven Architecture. Object Technology Series. Addision-Wesley Longman, Amsterdam, The Netherlands, 2004.

[8] Thomas Stahl and Markus Völter. Model-Driven Software Development: Technology, Engineering, Management. Wiley & Sons, 1st edition, 2006.

[9] Hsu, C., Bouziane, M., Rattner, L. and Yee, L. "Information Resources Management in Heterogeneous, Distributed Environments: A Metadatabase Approach", IEEE Transactions on Software Engineering, Vol. SE-17, No. 6, June 1991, pp. 604-624.

[10] Fischer, G. (2007): "Designing Socio-Technical Environments in Support of Meta-Design and Social Creativity", Proceedings of the Conference on Computer Supported Collaborative Learning (CSCL '2007), Rutgers University, July pp. 1-10.

[11] Fischer, G., & Giaccardi, E. (2006) "Meta-Design: A Framework for the Future of End User Development." In H. Lieberman, F. Paternò, & V. Wulf (Eds.), End User Development — Empowering people to flexibly employ advanced information and communication technology, Kluwer Academic Publishers, Dordrecht, The Netherlands, pp. 427-457.

[12] E. F. Codd: Relational Completeness of Data Base Sublanguages. In: R. Rustin (ed.): Database Systems: 65-98, Prentice Hall and IBM Research Report RJ 987, San Jose, California : (1972)

[13] Gonzalez-Perez, C. and B. Henderson-Sellers, 2008. Metamodelling for Software Engineering. Chichester (UK): Wiley. 210 p

[14] "Ecore" http://www.eclipse.org/modeling/emf/?project=emf

[15] L. Lyadova. Technology for creating a dynamically adaptable information systems // Proceedings of International Conference on Artificial Intelligence and Systems (AIS'07). – Moscow: Fizmatlit, volume. 2, 2007.

[16] Y. Rogozov, A. Sviridov, S. Kucherov. An approach for formal representation of metamodels. Proceedings of first international conference on Actual problems of information systems and processes constructing. Taganrog: SFEDU, 2010, p. 9-15.

[17] Youri I. Rogozov, Alexander S. Sviridov, Sergey A. Kutcherov, Wladimir Bodrov. Purpose-driven approach for flexible structure-independent database design. Proceedings of the Fifth International Conference on Software and Data Technologies, ICSOFT 2010, Volume 1, p.356-362

[18] S. Kucherov. Performance evaluation of the statistical DB structure. Proceedings of seventh All-Russian conference «Information technologies, system analysis and management». - Taganrog: SFEDU, 2009. p. 138-141.

[19] Y. Rogozov, A. Sviridov, S. Kucherov. Performance evaluation of the structure-independent database SiDB for the purposes of full-text search. Proceedings of first international conference on Actual problems of information systems and processes constructing. Taganrog: SFEDU, 2010, p. 196-199.