Method of a Structure-Independent Databases Design in Configurable Information Systems

Yuri I. Rogozov, Alexander S. Sviridov and Alexander N. Belikov
Department of System Analyses and Telecommunications,
Southern Federal University, 1 Engels, Taganrog, Russia

Keywords: Variability of Data, The Method of the Database Design, The Data Model, Structure-Independent Database.

Abstract: We propose a method of developing different structure-independent databases using relational technology. The method allows depending on the motivation of the developer to obtain various structure independent databases using relational technology. The requirements that structure-independent database must satisfy have been formulated. Alternatives of method realization were considered.

1 INTRODUCTION

Nowadays information systems work with complex data structures with a high degree of variability (Rogozov Y., Degtyarev A., 2014). Configurable information systems are one type of such systems (Rogozov Y., Sviridov A., Belikov A., 2014). Often we have to design a database (DB) in conditions of the absence of a priori description of the structure of stored information. Application of relational technology in such situation becomes difficult: variability of the structure of stored information affects the physical layer of implementation, which requires redesigning the entire database. As a result, there are additional costs, both time and financial. To reduce costs and continue to use relational technology, some developers and researchers create a database model with physical layer of implementation which is independent of the logical structure (Paley, 2002; Tenzer, 2001; Polishchuk, Chernykh, 2009).

In this paper, we propose a method that allows depending on the motivation of the developer to obtain various structure independent databases using relational technology. Sequence of steps that leads to compliance with the identified requirements and obtaining the physical structure of the database is defined.

2 REQUIREMENTS FOR THE METHOD

To formulate the method there has previously been carried out the analysis of the physical layer implementation of the relational model and the Entity-Attribute-Value model, which reveals the structural requirements to independent databases.

For structure-independent database there is no need of a priori description of the domain – they are ready for use, even if they don't contain any entities or attributes. A complete separation of the physical and logical levels of implementation is achieved:

•Logical structure that characterizes user data is described not by the physical structure of the database, but by the metadata directly stored in it;

•The physical structure of the database is unchanged and its creation relational technology can be used.

Conducted analysis of data models (Kucherov, S., Rogozov, Y., Sviridov, A., Zhibulis, Y., 2010) allows us to formulate the requirements that must be satisfied by structure-independent database, implemented using relational technologies:

1. The metadata that define the logical structure of the database is stored directly in the database in form of data sets in relational tables and make up the metadata subschema;

2. Data are grouped by type in the tables that store the attribute values indicating tenancy in a form of reference to the metadata tables, and make

Rogozov Y., Sviridov A. and Belikov A..
Method of a Structure-Independent Databases Design in Configurable Information Systems.
DOI: 10.5220/0005110201390145
In *Proceedings of 3rd International Conference on Data Management Technologies and Applications* (DATA-2014), pages 139-145
ISBN: 978-989-758-035-2

up the appropriate subschema. Name of the field, that stores the value, is not the name of the attribute;

3. Data structure in the database is represented by a sparse matrix. Logical data structure is formed actually, i.e. comprises only entities and attributes, that are really used in system, and can be supplemented or changed at any time;

4. Total number of tables in a database on the physical layer is fixed and depends only on the number of metadata tables and the types of data and is independent of the logical structure of data stored. Limitation of the number of tables in the physical layer is the key to increase the productivity and reduce the complexity of database queries and maintaining the characteristics obtained during its operation.

These requirements are fulfilled by most of known models of structure-independent databases that use relational technology (Paley, 2002; Tenzer, 2001; Polishchuk, Chernykh, 2009), but each model corresponds to a specific motivation of the developer.

3 THE METHOD OF CONSTRUCTING THE STRUCTURE-INDEPENDENT DATABASES

Regardless of the motivation of the developer there is a finite sequence of steps leading to obtain the necessary set of linked relational tables corresponding to above requirements. This set of tables is the physical layer structure-independent database, and the sequence of steps is a method of constructing a structure-independent database using relational technology (Rogozov Y., Sviridov A., Kucherov S., 2014).

For the first requirement it is necessary:

1. Determine the set of metadata needed to describe the logical structure of the database;

2. Identify the metadata structure and implement it in the form of a set of linked relational tables that make up the metadata subschema. This step includes:

•divide of generated metadata set into groups of tenancy;

•create for each such group a separate relational table;

•specify the relationships between tables;

To fulfill the second requirement it is necessary:

3. Determine the types of data that will be used to store the attribute values;

4. Develop data subschema in the form of a finite set of similar unrelated relational tables needed to store attribute values. It is advisable to store the values in the form of triples "entity- attribute-value", where the first two elements – are links to entries in the metadata tables. This enables to present the data structure in form of sparse matrix;

For the final formation of structure-independent database, you should:

5. Implement the physical structure of the database.

Performing the third and fourth requirement is a consequence of implementation of the first two requirements:

• Presentation of stored data structures in the form of sparse matrices is implemented by storing the values in the form of triples (sparse matrix of instances) and storage of metadata within the database (sparse matrices of entities and attributes);

• Independence between the physical and logical levels is implemented by storing metadata in the database.

Application of the above sequence of steps precedes the development of user (logical) data structures (Rogozov Y., Sviridov A., Kucherov S., 2014), which is performed by the developer-defined database technology.

4 THE PROOF OF METHOD

Proposed method allows to get different structure-independent databases using relational technology depending on the goals of the developer. To prove this statement, let's consider the example of obtaining the already known structure on the basis of this method.

Example 1. In (Paley,2002) authors describe the database model, which is the aim of developers to create a universal relational structure for processing and storage of quasi-structured data in terms of object-oriented technology – in the form of classes and objects. According to the proposed method of structure-independent database designing to obtain a model of such structure is as follows:

Stage One. Defining Metadata Sunschema.

Step 1. To store entities and attributes, it is needed to select the representation format "Directory". In this case, the implementation E^R of the entity set $E = \{e_i\}$ is represented by the following ternary relation (table):

$$E^R = (ID^E, E, F)$$

where ID^E – is a set of unique identifiers, E – is the original set of entities.

For relationship graph F the following is correct:

$$\pi_1(F) = ID^E, \pi_2(F) = E$$

$$\pi_3(F) = ID^{E_p}, ID^{E_p} \subseteq ID^E$$

Where ID^{E_p} - is the set of identifiers of parent classes of objects, because of using which hierarchy is achieved.

Implementation A^R of the attribute set is presented as ternary relation (table):

$$A^R = (ID^A, ID^E, \pi_1 A, \pi_2 A, VL, F)$$

where *IDA* – is the set of unique attribute identifiers, *VL* – is the lengths of the attribute values. At the same time because of the unique identifiers of classes A^R is also a realization of the relationship.

Step 2. Practical significance of relations exists only between objects. It is in this context the relationship $L = (E, A, F)$ should be used. To convert the relations $L = (E, A, F)$, $R = (E, I, F)$ into the corresponding tables, it is needed to define the following "References":

1. Directory of objects relations, the realization L^R:

$$L^R = (NL, ID^O, F)$$

where ID^O – is a set of unique identifiers of objects, *NL* – set of link names.

For relationship graph F the following is correct:

$$\pi_1(F) = NL$$

$$\pi_2(F) = ID_1^O, ID_1^O \subseteq ID^O$$

$$\pi_3(F) = ID_2^O, ID_2^O \subseteq ID^O$$

Where ID_1^O and ID_2^O - are the subsets of the first and second binding objects, respectively.

2. Directory of objects, the implementation of relation R^R:

$$R^R = (ID^O, ID^E)$$

Second Stage. Defining Data Subschema.

Step 3. It is nesessary to determine the range of acceptable values for the data elements. Since the model is conceptual, let's choose one base area – individual characters and character strings:

$$C = \{C_1\}$$

where *C1* – is a set of random character strings of limited length.

Let's specify a set *N* of allowed value area names:

$$N = \{n_1\}$$

where *n1* = "string".

Let's specify the correspondence, thereby determining the available data types in the database:

$$C = \{C_1\}, N = \{'строка'\}, T = C \times N = \{C_1, 'строка'\}$$

Step 4. Correspondence potency $|T| = 1$, so it's nesessary to create seven identical data tables, which are an implementation of relationship $V_{t_i} = (I, A, D, F)$:

$$V^R = (ID^V, ID^I, ID_A^{SPR}, D, O, F)$$

Third Stage. Building Relationships.

Step 5. It's nesessary to define the sets $ID^{SPR}, ID^{S^R,L^R}, ID^I, ID^V, ID^{V^D}$ as primary and foreign keys.

Step 6. Based on this set of tables with primary keys it's needed to specify the relations between tables. Thus, on the basis of the formulated method we obtained database structure-model that is similar to an extended relational schema for the quasistructured data processing, which is described in the literature (Paley, 2002). Targets of modelers (Grigoriev, 2001) coincide with those considered in this example.

Example 2. In the works (Polishchuk, 2009; Chernykh, 2011) authors describe the database model. The aim of model developers is to create a universal relational structure for XML-data storing in terms of object-oriented technology. According to the proposed method to obtain the structure-independent database model of such structure we should perform the following steps:

Stage One. Construction of the Metadata Subschema.

Step 1. To store entities (in the terminology of the target – the object classes), we should select the format for the "Directory." In this case, the

implementation E^R of the entity set $E = \{e_i\}$ is represented by the following ternary relation (table):

$$E^R = (ID^C, E, A^R, F)$$

where ID^C – is a set of unique identifiers, E – is the original set of classes.

In XML technology, the structure of any classes can be described by the corresponding XSD- scheme (Veselov, 2000). Therefore, the directory E^R includes the implementation A^R of attribute set, which is a set of XSD-class schemas. Such representation of the class also describes the implementation S^R of the set of classes attributes $S = (E, A, F)$, where the relationships of each class with attributes are stored in the XSD-scheme.

Step 2. There are two kinds of relationships in this approach: classes links and object links. Therefore, for relation $L = (E, A, F)$ let's define two implementations:

1. Directory of class relations L_C^R implementation of relationship $L = (E, A, F)$:

$$L_C{}^R = (ID^{L_C{}^R}, NL, ID^C, F)$$

where $ID^{L_C{}^R}$ - is the set of unique identifiers of classes links, NL – set of link names.

For relationship graph *F* the following is correct:

$$\pi_1(F) = ID^{L_C{}^R} \; \pi_2(F) = NL$$

$$\pi_3(F) = ID_1{}^C, ID_1{}^C \subseteq ID^C$$

$$\pi_4(F) = ID_2{}^C, ID_2{}^C \subseteq ID^C$$

where ID_1^C and ID_2^C - are the subsets of the first and second linked classes, respectively.

2. Directory of object links, the implementation L_O^R of relation $L = (E, A, F)$:

$$L_O{}^R = (ID^{L_C{}^R}, ID^O, F)$$

Where ID^O – is a set of unique object identifiers, NL - set of link names.

For relationship graph F the following is correct:

$$\pi_1(F) = ID_1{}^O, ID_1{}^O \subseteq ID^O$$

$$\pi_2(F) = ID_2{}^O, ID_2{}^O \subseteq ID^O \; \pi_3(F) = ID^{L_C{}^R}$$

where ID_1^O and ID_1^O - are the subsets of the first and second linked objects, respectively.

Implementation R^R of relation $S = (E, A, F)$ in the context of this goal is considered in the next stage.

Second Stage. Defining the Data Subschema.

Step 3. Let us determine the range of admissible values for data elements. Since the model focuses on data storage in XML, we will use the same range of permissible values:

$$C = \{C_1\}$$

where C1 – is a set of random character strings of limited length (because support for XML data type does not exist in all DBMS).

Specify a set N of range of admissible values names :

$$N = \{n_1\}$$

where n1 = «BLOB».

Let us specify the line $T = (C, N, F)$, thereby determining the available data types in the database:

$$C = \{C_1\}, N = \{'BLOB'\}, T = C \times N = \{C_1, 'BLOB'\}$$

Step 4. Correspondence potency $|T| = 1$, so it's nesessary to create one data table, which is an implementation V^R of relationship $V_{t_i} = (I, A, D, F)$ and implementation R^R of relationship $L = (E, A, F)$: $R^R, V^R = (ID^O, ID^C, D, F)$

Third Stage. Building the Relationships.

Step 5. It's necessary to define the sets ID^C, ID^O and $ID^{L_C{}^R}$ as primary and foreign keys.

Step 6. Based on this set of tables with primary keys it's needed to specify the relations between tables. Thus, on the basis of the formulated method we obtained model of database structure that is similar to the model data storage using XML-scheme, which is described in the literature (Polishchuk, Chernykh, 2009).

In order to prove the applicability of the proposed in this paper method for solving a wide range of problems of flexible database design, let's consider the example of creating the new SIBD on its base.

5 DEVELOPMENT OF SIDB MODEL

Let's construct a model of structure-independent database that store relational data structures with a high degree of variability (Kutcherov, 2010; Rogozov, 2008). This model is called SIDB. According to the proposed method of designing structure-independent database model the process of obtaining such structure consists of three stages.

Stage One. Defining Metadata Subschema.

Step 1. To store entities and attributes, it is needed to select the representation format "Directory". It is necessary to bring into standard set $E = \{e_i\}$ a number of additions that will determine the implementation of the directory:

1. Hierarchy. Will be used to simplify the navigation in the SIDB logical structure, to store data models of different domains within a SIDB instance and to separate entities and attributes. Hierarchy is defined on the basis of the set of unique directory entries identifiers $\boldsymbol{ID^{SPR}}$.

2. Synonyms will be used to store the artificial primary keys that have semantic charge. Synonyms are determined by the set ***Y***.

3. Brief descriptions will be used to simplify the understanding of stored data structure and each of its elements. Descriptions are determined by the set ***H***.

4. Logical deletion identifier. In the field of flexible data structures one of the most important features is the possibility to recover from errors. Identifiers are determined by the boolean set ***O***.

5. Data Types will be used to specify the type of value stored in one way or another attribute. Data types are determined by the projection of the second element of the attributes set $A = \{< a_j, t_k >\}$.

Thus, the implementation E^R of the entity set $E = \{e_i\}$ and attributes set A^R is represented with the help of hierarchical directory – relationship (table):

$$A^R, E^R = (ID^{SPR}, E, \pi_1(A), Y, H, O, \pi_2(A), F)$$

For relationship graph ***F*** the following is correct:

$$\pi_1(F) = ID^{SPR}$$

$\pi_2(F) = E \cup \pi_1(A)$ – names of all entities and attributes.

$$\pi_3(F) = H\,,\ \pi_4(F) = Y\,,\ \pi_5(F) = O$$

$\pi_6(F) = ID_1^{SPR}, ID_1^{SPR} \subseteq ID^{SPR}$ – hierarchy of attributes and entities.

Where ID_1^{SPR} – is a subset of the identifiers of all entities and attributes, that are parental for one or more records.

$\pi_7(F) = ID_2^{SPR}, ID_2^{SPR} \subseteq ID^{SPR}$ – determining record belonging (is that entity or attribute),

where ID_2^{SPR} – is a subset of the identifiers of all entities that are directories. This approach allows to organize a set of logical directories within a single physical.

Step 2. It is necessary to transform relationships $S = (E, A, F)$, $L = (E, A, F)$, $R = (E, I, F)$ into the corresponding tables. Relationshipsare defined on the same sets. Let's define them in a single directory:

Object relations directory, realization $L^R$of the relationship $L = (E, A, F)$:

$$S^R, L^R = (ID^{S^R, L^R}, ID^{SPR}, O, F)$$

where ID^{S^R, L^R} – is the set of unique identifiers of relations.

For relationship graph ***F*** the following is correct:

$$\pi_1(F) = ID^{S^R, L^R}$$

$\pi_2(F) = ID_{E_1}^{SPR}, ID_{E_1}^{SPR} \subseteq ID^{SPR}$ – A subset of the unique identifiers of the parent entities.

$\pi_3(F) = ID_A^{SPR}, ID_A^{SPR} \subseteq ID^{SPR}$ – A subset of unique identifiers of the attributes.

$\pi_4(F) = ID_{E_2}^{SPR}, ID_{E_2}^{SPR} \subseteq ID^{SPR}$ – A subset of the unique identifiers of the child entities.

$\pi_5(F) = O$ – Identifier of a logical links deletion.

In order to identify the entity instances that are stored in a hierarchical directory, it is needed to define the following implementation R^R of relationship $R = (E, I, F)$:

$$R^R = (ID_E^{SPR}, ID^I, O)$$

where ID^I – set of instances unique identifiers.

Second Stage. Defining Data Subschema.

Step 3. It is nesessary to determine the range of acceptable values for the data elements. Let's define

five basic range of acceptable values that are most commonly used today in the relational database:

1. Random character strings of limited length – C_1;
2. Numeric values – C_2;
3. Temporary markers – C_3;
4. Random text strings of great length – C_4;
5. Large binary files – C_5;
6. Enumerable discrete values.

Furthermore, in relational databases relations between entities are used (for description of the logical database model) along with relations between instances of the entities (for description of the data stored). Therefore, we introduce another range of acceptable values – pointers at entity instances (C_7).

$$C = \{C_1, ..., C_7\}$$

Specification of the set ***N*** of names of acceptable values range:

1. String – n_1;
2. Number – n_2;
3. Date / Time – n_3;
4. Text Document – n_4;
5. Binary file – n_5;
6. Enumerated value – n_6;
7. Entity reference – n_7;

$$N = \{n_1, ..., n_7\}$$

Step 4. Correspondence potency $T = (C, N, F)$: $|T| = 7$, so it's nesessary to create seven identical data tables, which are an implementation of relationship $V_{t_k} = (I, A, D, F)$:

$$V^R = (ID^V, ID^I, ID_A^{SPR}, D, O, F)$$

In addition, for storage of enumerated attributes it should be entered the appropriate directory of valid values, that is the following relationship:

$$V^D = (ID^{V^D}, ID^A, D^D, O, F)$$

where $\boldsymbol{D^D}$ – is an appropriate domain of acceptable values of the enumerations. In this case, domain set ***D***, where the implementation relationship V^R is defined, will comprise a variety of unique identifiers of enumerated values.

Third Stage. Building Relationships.

Step 5. It's nesessary to define the sets $ID^{SPR}, ID^{S^R,L^R}, ID^I, ID^V$ and ID^{V^D} as primary and foreign keys.

Step 6. Based on this set of tables with primary keys it's needed to specify the relations between tables. Thus we'll obtain SIDB logical model (Fig. 1).

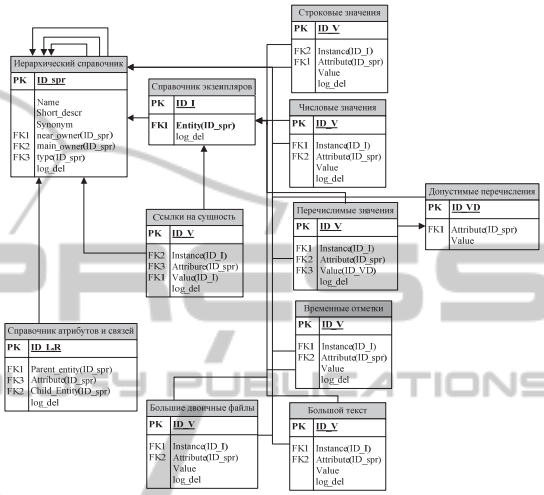

Figure 1: Logical model of SIDB.

6 CONCLUSIONS

The method proposed in this paper allows to build structure-independent databases using relational technology which has the following advantages:

• Limitation of the structural complexity of database queries by fixing the amount of relational tables at physical layer;

• Increasing of data processing operation speed. Relational technology is originally designed for static data structures and has rich set of productivity tools;

• Flexibility. There is no relationship between physical and logical database levels, implemented with the proposed method, allowing maximum flexibility;

• Low maintenance costs. There is no need to hold highly skilled professionals engaged in support of the physical layer performance, and that saves time and money.

As a result of applying the proposed in this article method there may be obtained a variety of different structure-independent databases. The implementation of each step of the method depends

on the motivation of the developer and is not limited by discussed in this article examples.

ACKNOWLEDGEMENTS

The research is performed within the government mandate № 0110021005901621. Theme № 213.01-11/2014-17 "Development of methods for data warehouse creation in configurable information systems and mechanisms for their implementation".

REFERENCES

Rogozov Y., Degtyarev A., 2014. The basic foundation of software framework for configuration underwater acoustic information systems with dynamic structure. *Information and Communication Technology for Education (ICTE-2013).* Publisher: WIT Press. Southampton, Boston, Vol. 58.

Rogozov Y., Sviridov A., Belikov A., 2014. Approach to CASE-tool building for configurable information system development. *Information and Communication Technology for Education (ICTE-2013).* Publisher: WIT Press. Southampton, Boston, Vol. 58.

Paley, D., 2002. *Modeling of quasistructured data.* Open Systems, 2002.

Tenzer, A. 2001. Database – is object storage // http://compress.ru/article.aspx?id=11515.

Kucherov, S., Rogozov, Y., Sviridov, A., Zhibulis, Y., 2010. *Approach to implementing database with the static structure of the underlying data model EAV*. News of Southern Federal University. Technical sciences.

Rogozov, Y., Dubrovsky, A., Sviridov, A., 2008. *Implementation of the new approach the idea of "programming without programmer"* News of Southern Federal University. Technical sciences.

Kutcherov, S., Rogozov, y., Sviridov, A., Bodrov, W., 2010. *Purpose-driven approach for flexible structure-independent database design*. Proceedings of the Fifth *International Conference on Software and Data Technologies.*

Rogozov Y., Sviridov A., Kucherov S., 2014. The method of configuring dynamic databases. *Information and Communication Technology for Education (ICTE-2013).* Publisher: WIT Press. Southampton, Boston, Vol. 58.

Polishchuk, Y., 2009. *Modeling of subsystems oriented on quasistructured objects storage.* New Technology, Information Technology, № (2009), - 2009. - 66-71

Chernih, T.A., 2011. *Improving the AMS of condensate field through the application of information subsystem of quasistructured data processing*. Orenburg.

Veselov, V., Dolzhenkov, A. 2000. *XML and database technology*. Publishing house Open Systems.

Rogozov Y., Sviridov A., Grishchenko A., 2014. *The method of data manipulation operations representation as a structure in structure-independent databases oriented on configurable information system development.* Information and Communication Technology for Education (ICTE-2013). Publisher: WIT Press. Southampton, Boston, Vol. 58.