The Formal Model of Structure-Independent Databases

Sergey Kucherov, Alexander Sviridov and Svetlana A. Belousova
Department of System Analyses and Telecommunications, Southern Federal University,
44 Nekrasovsky St., Taganrog, Russian Federation

Keywords: Structure-independent Databases, Formal Model, Metadata Manipulation.

Abstract: In this paper we'd like to propose a formal model of structure-independent databases. This formal model allows to describe and implement not only different SIDB that are implemented on relational technology, but also the tools of working with them. Model contains set of relations and operations which are basic and can be supplemented depending on the characteristics and requirements of implementation. To provide the flexibility structure-independent databases presupposes manipulation of data and metadata structures.

1 INTRODUCTION

In configurable information systems correspondence to the domain is recorded at the moment of delivery to the consumer, and this correspondence can be corrected at any time with minimal costs and without loss of efficiency of existing configuration (Rogozov, Degtyarev, 2014; Rogozov, Sviridov, Belikov, 2014).

Configurable information systems deal with data with variable structure – data sets with pre-defined and strictly fixed structure, which may be modified in accordance with changes of the domain. The main differences between data with variable structure and semistructured data with similar properties are shown during the work with them:

- For data with variable structure it is required to identify and record the database schema before using it.

- Schema of data with variable structure should be prescriptive, rather than describing, as in the case of semistructured data.

There are various private approaches to solving the problem of database flexibility for configurable information systems that can be generalized to two major categories: dynamic physical structure of the database (Ginige, 2010) and static physical database structure (Paley, 2002; Tenzer, 2001).

The conceptual model of structure- independent database will be considered in section 2.

Due to the fact that all known structure-independent databases (SIDB) are based on relational technology, the corresponding mathematical apparatus will be used for formalization: theory of relations (Anderson, 2004), theory of sets (Kuratowski, 1970) and Codd's relational algebra (Codd, 1972.). The formal model of SIDB will be presented in sections 3-4.

2 CONCEPT OF STRUCTURE-INDEPENDENT DATABASE

Static physical database structure provides a new level of flexibility – work with user data structures especially on the logical level, that is isolated from the majority of technical nuances, such as triggers, indexes, etc., and largely centered on the domain – on the structure and the links between the stored data, types of data stored and so on. Thus this variant of the problem solution of providing database flexibility for adaptive information systems is most preferred. Selection of this variant is caused by difficulties in creating the physical layer in the absence of exhaustive description of the subject area and by rising of maintaining and refining cost during the database usage.

Such decisions can be summarized by the term – structure-independent database. SIDB – is a database that is marked by the absence of any impact of changes at the conceptual or logical data model level of the domain on physical structure of the tables or records. SIDB is intended for use within a single implementation area of information systems

Kucherov S., Sviridov A. and A. Belousova S..
The Formal Model of Structure-Independent Databases.
DOI: 10.5220/0005110301460152
In *Proceedings of 3rd International Conference on Data Management Technologies and Applications* (DATA-2014), pages 146-152
ISBN: 978-989-758-035-2

(medical, economic, etc.) and is able to continue to operate and save the functionality after any change in the data model of the domain, which makes it universal within this area.

Representation of the data structure in the form of sparse matrix allows to provide the possibility of its dynamic changing, and direct storage of metadata in the database allows to make independent physical and logical levels of implementation. Using these statements to solve the problem of flexibility of databases leads to three-dimensional representation of this database in the basis: "entity-instance-attribute" (Kucherov, 2010; Kucherov, 2011) shows that there is no need in priori domain description for flexible database – it is ready for use, even if it does not contain any entity or attribute. A complete separation of the physical and logical levels of implementation is achieved.

SIDB should store not only user data but also a description of their structure. Therefore, SIDB should contain the relevant subschemas for explicit storage of both data and metadata. This statement is also supported by general features of considered databases for storing user data structures.

An important point is the access to data and metadata. The principles embodied in the data model EAV lay in the basis of SIDB. One of the fundamental principles is to store data in columns (Abadi, 2008; Boncz, 2005; Boncz, 1999; Stonebraker, 2005). Unlike traditional storage by rows, which is the characteristic of table data presentation, data storage by columns suggests such correspondence: one line = one value of one attribute of one entity. In this case the defining of entity and attribute, which value is stored in the row, is performed not by naming tables and fields, but making links to the appropriate tables of entities and attributes. This makes it possible not only to change the value, but also other elements: entities and attributes.

Since column-storage is not common form of data presentation, and the user perceives data in table form (represented by rows) better, there is a problem of direct and inverse transformation between views. It is obvious that operation of transformation will always be performed by one algorithm (only the values of the rows will vary). Thus, it makes sense to identify and include in SIDB number of mechanisms that provide for the user the possibility to work with the data presented in columns in a familiar table form.

SIDB include four main components:

1. Metadata. Set of directories, implemented as static subschema of tables;

2. Data. Set of directories, implemented as a static unbound set of similar tables on the number of data types used.

3. Mechanisms for data processing. They are a set of parts comprising procedures, functions, and interfaces for providing to the user the normal operation mode with table data view.

4. Mechanisms for metadata processing that constitute set of components, including procedures, functions, and interfaces for multiple changes of user data structures described within SIDB.

Generalized conceptual representation of structurally independent database (Kucherov, 2009) is shown in Figure 1.

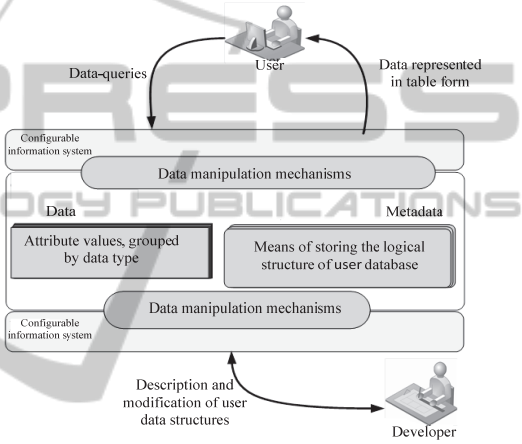

Figure 1: Conceptual model of structure-independent database.

SIDB range of users is expanded as compared with a conventional relational databases, for which it is allocated only data consumers. Innovation is the inclusion of developer (database administrator) to the SIDB users. Metadata manipulation mechanisms should be available for this category (Rogozov, Sviridov, Grishchenko, 2014). Through these mechanisms the description and modification of user data structures is performed. Also it is possible to organize the interface part of the mechanisms in the way when changes will be carried out without the involvement of technical experts.

Based on this concept representation the usage of SIDB can be introduced as follows:

1. Logical Data Model of the domain that is defined by one of the classical types, is converted by means of deterministic procedures of conversion into the form of its storage in SIDB;

2. Logic level of SIDB is filled with relevant directories, hierarchies , key values , etc.;

3. SIDB logical model with the help of deterministic procedures is converted into a

relational form. Metadata and data are divided into relevant tables.

According to this we can mark the main task of SIDB formalizing – formalization of the structure and methods of data manipulation and data structures manipulation (Rogozov, Sviridov, Kucherov, 2014).

3 FORMAL MODEL OF STRUCTURE-INDEPENDENT DATABASE

The structure of considered existing databases can be generalized to the following basic components: domain objects; characteristics of domain objects; relations of characteristics with objects; relations of objects with objects; object instances; relations of object instances; data stored in the form of values of objects characteristics.

We use the following notation:

- object (in object-oriented and graph data models) , the relation (in relational data model) = entity;
- characteristic, feature, quality of the object = attribute.

Based on this, we formulate a mathematical model of the SIDB structure, which can be represented as follows (Kucherov, 2011; Kucherov, 2010):

$$M \equiv \langle E, A, S, L, R, V \rangle \tag{1}$$

where: E – component "entities"; A - component "attributes"; S - component "entities structure"; L - component "structure of relations"; R - component "entity instances"; V - component "data" (or "attribute values").

Entity. Suppose there are n entities that can be identified. Then all entities will be described in the following finite set:

$$E = \{e_i\} \tag{2}$$

where $i = \overline{1,n}$

Attribute. Suppose that there is a finite set *C* of allowed value area:

$$C = \{C_1, C_2, ..., C_n\} \tag{3}$$

where *Ci* – i-th set of allowed values.

Suppose that there is a finite set *N* of allowed value area names:

$$N = \{n_1, n_2, ..., n_k\} \tag{4}$$

where ni – i-th name of certain allowed value area. For sets *C* and *N* there can be set bijective correspondence of names *ti* and areas *Cj*, called the set of data types.

$$T = (C, N, F) \tag{5}$$

where $F \subseteq C \times N$, $t_i \in T$, – i-th data type.

Then domain *Di* will be a finite subset of the named elements of bijective correspondences *tj*, and domain name *Di* will be called as attribute *Aj*:

$$D = \{D_1, D_2, ..., D_n \mid D_i \subseteq t_j\} \tag{6}$$

Thus, the finite set of all attributes stored in SIDB is represented in the form of pairs:

$$A = \{< a_j, t_k >\} \tag{7}$$

where *aj* – certain j-th attribute, *tk* – the data type of the attribute.

Structure of Entities. Each entity of the domain is defined by its name and has the structure in a form of attributes set. Entities structure can be given in the following form:

$$S = (E, A, F) \tag{8}$$

S is a binary relation defined on sets *E* and *A* with the relationship graph *F*, that the projection on the first component $\pi 1$ of the graph *F* is a subset of entities having the structure of *E′* of the set *E*, and the projection of the second component $\pi 2$ of relations graph *F* – is a subset of attributes *A′* of set *A*:

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

If $\exists e_i \in E \,\&\, \exists a_j \in A$ are such that the relation *S* contains a tuple *U*, for which $\pi_1(U) = e_i$, $\pi_2(U) = a_j$, than the attribute *aj* belongs to entity *ei*. Unlike traditional relational databases in SIDB the same attribute can belong to multiple entities.

The Structure of Connections. The data model of any subject area, made in any of known forms, should reflect not only a set of entities of the domain and their structure as a set of attributes, but also the relationships between individual entities. In SIDB such links can be formally presented as follows:

$$L = (E, A, F) \tag{9}$$

L is a ternary relation, defined on the sets *E* and *A*, with the relationship graph *F*, that the projection on the first component $\pi 1$ of the graph *F* is a subset

of parent entities having the structure of E' of the set E, and the projection of the second component $\pi 2$ of graph relations F – is a subset of child entities E' of the set E, and the projection of the third component $\pi 3$ of relations graph F – is a subset of key attributes A' of set A:

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

If $\exists e_i, e_j \in E \,\&\, \exists a_k \in A$ such that the relation L contains a tuple U, for which $\pi_1(U) = e_i$, $\pi_2(U) = e_j$, $\pi_3(U) = a_k$, entities *ei* and *ej* are linked by the attribute *ak*. And *ei* – is the parent entity, *ej* – child entity, *ak* is primary key and *ej* is foreign key of entity.

Entity Instances. In the relational model, both in the physical and logical levels, each instance of a single entity is uniquely determined by the values of one tuple (row in the table), with no need to introduce extra copies of identifying features.

At the same time SIDB storage of entity instances in the physical layer is carried out not in the form of sequences of length n, where n – is the number of attributes of an entity, but in $n' \le n$ triples – ternary tuples, each of which stores value of one attribute of the entity. Here, the *n'* refers to the fact that the triple *ni* exists if the attribute has the value *ai*. With this approach, the value «null» is not stored and the place for such value them is not reserved.

Due to these features it is necessary to identify groups of tuples belonging to one entity instance. Formally, the identification can be represented as follows:

$$R = (E, I, F) \tag{10}$$

R is a binary relation defined on the sets *E* and *I*, with the relationship graph *F*, that the projection on the first component $\pi 1$ of the graph *F* is a subset of parent entities having the instances E' of the set *E*, and the projection of the second component $\pi 2$ of relations graph *F* – is a set *I* of all instances of all entities that are stored in database:

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

If $\exists e_j \in E$ such that the projection of the second component $\pi 2$ of relation graphic *F* with selection operator $\sigma_{e_j}(F)$ is a subset I' of set *I*:

$$\pi_2(\sigma_{e_j}(F)) = I', I' \subseteq I$$

then for the entity *ej*, there is a set $I' = \{i_k\}$ of instances of amount $n = |I'|$. For entities which do not have any entities in SIDB $\pi_1(R)$ and $\pi_2(R)$ are equal to zero.

Data (Attribute Values). As has been noted, in SIDB data is stored as tuples of fixed length and structure that regardless of the data type can be represented as follows:

$$V_{t_i} = (I, A, D, F) \tag{11}$$

V_{t_i} is a ternary relation, defined on the sets *I, A* and *D*, with the relationship graph *F*, that the projection on the first component $\pi 1$ of the graph *F* is a subset of all entity instances that consist of attributes type *ti, I''* of the set *I*, and the projection of the second component $\pi 2$ of relations graph *F* – is a subset *A*' of attributes having type *ti* of the set *A*, and the projection of the third component $\pi 3$ of relations graph *F* – is a subset *D''* of domain set *D'* of type *ti*:

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

Relations between Entity Instances. Along with storing the structure of relationships between entities there exists the problem of storage of relations between instances of these entities. In other words, the first defines the logical data model of the domain, and the second allows you to create in the output the separate information entity of the domain, which in a logical model can be represented by a set of tables. For example, such a task is to build a directory that stores information entity "Employee", which in the logical data model is represented by two entities "Passport data" and "History of the salary".

To solve this problem, you can use the relations (9) as a container for the storage of such relations. But since the relationship between specific values are related to the field of data and appear during the immediate operation of the database, it is advisable to:

1. Identify specific data type *Cn* named *tk*, such that *Ci = I*. That is, the allowed value area of this data type will include all elements of the set *I* of entity instances. Than to add this type to the set (3), and to add name of this type to the set (4). To add

the correspond domain *Dn* to the set (6). In this case, *Dn= Cn*.

2. Use for storage of links between entity instances the relation (11), where use the desired type *Cn* named *tk* as the attribute type:

$$V_{t_k} = (I, A, D, F)$$

Where the projection on the first component $\pi 1$ of the graph *F* is a subset of all instances of parent entities *Ip* of set *I*, the projection of the second component $\pi 2$ of relations graph *F* – is a subset *A'* of attributes having type *ti* of the set *A*, and the projection of the third component $\pi 3$ of relations graph *F* – is a subset of instances of child entities *Ic* of set *I*:

$$\pi_1(F) = I^p, I^p \subseteq I$$
$$\pi_2(F) = A', A' \subseteq A$$
$$\pi_3(F) = I^c, I^c \subseteq I$$

The developed formalized model is basic and can be supplemented depending on the extension of the SIDB domain. On the basis of the model (1) there can be obtained a variety of different SIDB implementations depending on the particular purpose of the developer.

4 FORMAL MODEL OF METADATA MANIPULATION MECHANISMS IN SIDB

In classical relational databases to describe the manipulation methods of data structures the data definition languages are used (in particular, DDL-subset of SQL operators – Create, Alter, Drop). In contrast to this, the way in which SIDB provide the flexibility along with the physical layer independence (Abadi, 2008; Boncz, 2005; Boncz, 1999; Stonebraker, 2005; Kucherov, 2009) allows to solve these problems using data manipulation languages (in particular, DML-subset SQL operators – Select, Insert, Update, Delete) (Kuratowski, 1970).

Manipulation of data structures presupposes the presence of operations of adding, deleting and modifying the following components of data structure: 1. Entities; 2. Attributes; 3. Relations "entity-attribute"; 4. Relations "entity-entity".

In SIDB by storing data in columns operations are performed with the help of the data manipulation language SQL. Let us examine them in detail.

Entity Adding. To add the entities you should perform operation of sets union:

$$E'' = E \cup E', E' = \{e'_{i+1}, ..., e'_k\}$$

where:

E = {e1,...,ei} – is a finite set of entities available in SIDB;

E' = {ei+1,...,ek} – is a finite non-empty set of entities to be added;

E'' = {e1,...,ei,ei+1,...,ek} – is a result finite set of entities available in SIDB.

Entity Deletion. To remove the entity you should perform operation of sets difference:

$$E'' = E - E', E' = \{e'_{i+1}, ..., e'_k\}$$

where:

E = {e1,...,ei,ei+1,...,ek} – is a finite non-empty set of entities available in SIDB;

E' = {e'i+1,...,e'k} – is a finite nonempty set of deleted entities;

E'' = {e1,...,ei} – is a result finite set of entities available in SIDB.

Entity Changing. To change the entity you should perform two operations - the difference and the union:

$$E'' = E - E', E' = \{e_{i-n}, ..., e_n\}$$
$$E = E'' \cup E''', E''' = \{e'''_{i-n}, ..., e'''_i\}$$

where:

E = {e1,...,ei } – is the initial and the resulting finite non-empty set of entities available in SIDB;

E' = {e'i-n,...,e'i} – is a finite nonempty set of entities to be changed;

E'' = {e1,...,ei-n-1} – is an intermediate finite set of entities available in SIDB;

E''' = {e'''i-n,...,e'''i} – is a finite nonempty set of modified entities.

Creation of "entity-entity" Links. To add relationships between entities it is needed to perform operation of sets union:

$$L'' = L \cup L', L' = \{<e'_{i+1}, e'_{k+1}, a'_{j+1}>, ..., <e'_m, e'_n, a'_p>\}$$

where:

L = {<e1,e2,a1 >,...,<ei,ek,aj>} – is a finite set of entity-entity links available in SIDB;

L' = {<e'i+1,e'k+1,a'j+1>,...,<e'm,en,a'p>} – is a finite nonempty set of entity-entity links to be added;

L'' = {<e1,e2,a1>,...,<ei,ek,aj> , <e'i+1,e'k+1,a'j+1>,..., <e'm,en,a'p>} – is the result set of entity-entity links available in SIDB.

Changing the "entity-entity" Links. To change the connections it is necessary to perform two operations – the difference and union:

$$L''=L-L',L'=\{<e'_{i-n},e'_{k-n},a'_{j-n}>,...,<e'_{i},e'_{k},a'_{j}>\}$$
$$L=L''\cup L''',L'''=\{<e'''_{i-1},e'''_{k-1},a'''_{j-1}>,...,<e'''_{i},e'''_{k},a'''_{j}>\}$$

where:

L = {<e1,e2,a1>,...,<ei,ek,aj>} – the original and the result finite nonempty set of entity- attribute links available in SIDB;

L' = {<e'i-n,e'k-n,a'j-n>,...,<e'i,e'j,a'k>} – finite nonempty set of entity-attribute links to be changed;

L' = {<e'i-n,e'k-n,a'j-n>,...,<e'i,e'j,a'k>} – intermediate finite set of entity-attribute links available in SIDB;

L''' = {<e'''i-n,e'''k-n,a'''j-n>,..., <e'''i,e'''j,a'''k>} – final nonempty modified entity-attribute links.

5 CONCLUSIONS

So the formal model of structure-independent database was proposed and developed (Kucherov, 2011; Kucherov, 2010), based on the theory of relations, set theory and Codd's relational algebra. As part of a formal model of structure-independent databases are available: SIDB structure model, model of operations on metadata (user data structures), model of operations on user data.

Formal model allows to describe and implement not only different SIDB that are implemented on relational technology, but also the tools of working with them. Considered in model set of relations, sets and operations are basic and can be supplemented depending on the characteristics and requirements of implementation.

The result set of SIDB models must be implemented according to known database technologies. To ensure the feasibility in the form of various particular structure independent databases, which satisfy the requirements of the particular configurable information system, it is required to develop an appropriate design method.

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

Abadi, D.J., 2008. *ColumnStores vs. RowStores: How Different Are They Really?* Proceedings of the ACM SIGMOD International Conference on Management of Data, Vancouver, BC, Canada.

Anderson, J., 2004. *Discrete mathematics and combinatorics.* Moscow: Publishing House of Williams

Boncz, P., 1999. *MIL primitives for querying a fragmented world.* VLDB Journal, 8 (2)

Boncz, P., 2005. *MonetDB/X100: Hyper-pipelining query execution.* In CIDR.

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

Ginige, A., 2010. *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, Volume 2.

Kucherov, S.A., 2009. *Structure independent database SIDB for web-oriented systems development in automated directory systems.* Collected materials «Science Week 2010" V.2. - Taganrog Univ Tsure ,

Kucherov, S.A., 2010. *Way a formal representation of metamodels.* Actual problems of information systems and processes – SFU-publishing. Taganrog.

Kucherov, S.A., 2011. *Formalized model of structurally independent databases.* Structure and manipulation Informatization and Communication , № 3 (2011)

Kucherov, S.A., 2011. *The method of constructing structure-independent databases using relational technology.* New Technology, Information Technology, № 2.

Kuratowski, K., 1970. *Set Theory: Translation from English*. MI briefly edited Taimanov. Mostowski – Mir.

Kutcherov, S.A., 2010. *Purpose-driven approach for flexible structure-independent database design.* Proceedings of the Fifth International Conference on Software and Data Technologies, ICSOFT, Volume 1.

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

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. 1.

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. 1.

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. 1.

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. 1.

Stonebraker, M., 2005. *C-Store: A Column-Oriented DBMS.* In VLDB.

Tenzer, A. 2001. Database – is object storage. Computerpress.