Concept of Factographic SQL-Based Database

Alexander Grigoriev
Moscow Engineering Physical Institute
Moscow, Russia
e-mail: mazag@glas.apc.org

Alexey Grigoriev
VNIIEF
Sarov, Russia
e-mail: mazag@glas.apc.org

Sergey Nikonov
VNIIEF
Sarov, Russia

Abstract

We describe new concept of object-oriented factographic database. The goal is to collect data of different types, to store and to analyze them by means of SQL-compatible database management system (DBMS). Such DBMS should not require any changes in the database structure in order to add a new object class and, consequently, make structure hidden (encapsulated) for end user.

1. Introduction

Development of any data retrieval system requires on initial stage to build and maintain the factographic database, that is used to collect and analyze information. Usually, not all necessary information concerning data structures is known at the beginning, thus making it difficult to predict all the possible changes in data structures. Collection of information and data identification may be split between independent research groups with subsequent combining of developed fragments and establishing required relations between data structures of different types.

The interface of data access has to hide internal structure, maintain a referential integrity, dynamic change of data structures and their relations, mechanism of data retrieval and separation of access rights for user and administrator.

We propose the model of database which consider description of user structures as an information stored in DBMS. In this case any changes of data structures will not require change of the structure of tables, rewriting of the stored procedures, revision of referential integrity and so on.

Such an approach also allows to keep different descriptions of the whole structure, making all necessary transformations during analysis. It makes more simple to control the set of dynamic relations which allow to construct any graph of relations and do not violate the referential integrity.

The idea to develop the factographic database with described above properties emerged after many attempts to create complicated data retrieval system based on relational DBMS for one stage. The expansion of data types and analysis of already collected data required constant revision of data structures and modification of completed fragments. Number of relations between tables was increasing, so that nearly each table could be referred to all others.

It became obvious that the number of stored procedures, providing the retrieval and data filtering with account of all possible relations, grows as the number of tables squared. We decided to change the approach and construct the data retrieval system on the basis of proposed concept.

2. Choice of realisation terms

The use of the different networks and specialized factographic DBMS were not convenient because of the absence of portability. Such problem usually forces developers to make the choice of platform at the very beginning. The use of SQL, on the other hand, permits to make a database model independent of platform, which greatly facilitates the transition from one SQL DBMS to another. We selected platform in order to minimize expenses of development. The following hardware and software were used to test proposed concept:

- computers - Intel Pentium Pro;
- operation system - Windows NT 4.0;
- DBMS - MS SQL Server 6.5.

3. Conceptual database model

To give the user an opportunity to define structures of his data, all information in a database presented as a set of objects. Each object belongs to its class (see Figure 1). The set of properties is defined for each class. Types of properties
are the same as the types defined in common SQL standard, including the types for textual and graphic data.

Figure 1 Classes and objects

This means that each object consists of its kernel and the list of properties with assigned values. The kernel of the object is the same for any object class. It keeps the unique name of the object and reference to its class. Description of each class contains the unique class name and comments of its owner, as well as reference to the list of properties which belong to given class. The names of properties are unique inside the class. Each property has one of the types predefined in common SQL standard. Values of properties of different types are kept in a different tables.

In such a way it is easy to link the different objects due to identical kernel structure of different object classes. The kernel of each object is connected with the list of references to the kernels of other objects. Such a link will be called as "a short link" (Figure 2). The objects \(A_1, A_2, \ldots, A_n\) will be called as "connected by a distant link" if there is no short links between them but we can find such a sequence of objects \(A_1, A_2, \ldots, A_{n-1}, A_n\) which are connected in pairs by the short links. All distant links as well as short links for each object are kept in a database. Distant links are updated automatically, if there is any change in short links, by means of triggers of the table, which keeps track of short links.

All distant links are arbitrary and used only to retrieve the objects and to eliminate additional costs from walking around the graph of references. This increases an efficiency by making it possible to use only one query instead of a cycle of queries if we use the short links only.

The base types of short links are included in model to regulate use of short links. As these links are directed, identifiers of two classes accordingly representing beginning and end of the link are defined for each such a type. Besides each type of a short link has a unique name. We pursued two purposes introducing the given classification:

1. Convenience of the user to realize navigation among links because it allows to sort objects by classes to which the transition can be carried out and provides rendering of friendly names for hyperlinks formed by means of browser.
2. Prompting about at filling a database. The user engaging in input of the information in a database loses an opportunity to set short links which were not stipulated by the developer.

Also the keeping of the similar information is useful for understanding the structure of the particular project.

In order to divide the data on different thematic fragments we introduce the "themes". The list of such themes is kept in the database. Each object belongs to the certain theme (see Figure 3).

The set of themes allows to extract the subset of objects which can be processed by one user. This means that we can define the access area for user, which is very important for multi-user environment. We introduce also the term "section". Sections are used in 2 cases:

- if we restructure the part of or all stored information, then the new and old versions of structures are kept in the different sections;
- if we make a regular duplication of database fragments for remote users, then all duplicated objects unified into one section.

To provide more abilities for users and to organize internal control of their activities, we included a special table, containing the user context (Figure 4) into the database. Internal mechanisms of identification and authentication are used to detect which user entered into the system. Every user has its own set of themes, which he may use, written in his context.
That grants him ability to see only a subset of the objects, which belong to the given themes. Such mechanism is intended for creation of hierarchical organization of experts processing the stored information. It begins with the groups working on the particular problems at the bottom level and ends up with the project coordinators at the top level.

The user context keeps also the list of stored samples. The term "stored sample" is discussed below.

The important component of the database is a powerful data retrieval system. We suggest to use the service functions of standard DBMS, where each data structure has its own table (or set of tables). The possibility to create and execute SQL queries may be granted in this case to the client applications. Another possibility to hide internal structure of database is to give a set of stored procedures as a program interface. The given procedures can provide a search of records in the tables by the conditions, when values of fields belong to the given intervals. Besides, if the table $A$ contains the references (Foreign Key) on the table $B$, a choice of records from the table $A$ by the conditions for referred records from the table $B$ should be possible.

Unfortunately, both structure of classes and sets of properties are unknown at the moment of database development and programming the client applications. That's why the use of predetermined queries both directly and inside the stored procedures is impossible for this reason.

To solve such problem, let us introduce the concept of "stored sample" (see Figure 5). As it was specified above, the samples are associated with users, that created them. All stored samples of the same owner should have unique names. The stored sample contains the sets of conditions applied to the properties of objects. Each condition concerns a certain property of a certain class and determines the range of property values for this condition. We shall note that, if there are several conditions for one class, all of them should be taken into account.

Besides a set of conditions, the stored sample contains the list of classes (classes of sample), which objects are given to the user as result of selection. Set of objects which satisfy to conditions of stored sample (i.e. the sample contains a condition for any its property and this condition is true) we shall name as a "direct result of sample".

![Figure 3 Sections and themes](image)

*Figure 3 Sections and themes*

Set of objects which are association of direct result of stored sample and all objects connected with them by a distant or short link we shall call as "complete result of sample". Only those objects are retrieved from a complete result of stored sample after execution of sample which classes belong to the classes of sample and the themes are accessible to the user.

To create or remove the stored sample and to add some conditions to it, the mechanism of stored procedures is used. The validity control is carried out at any attempt to add a condition to a sample. Validity control consists of checking the property presence and validity of a type and a range.

Stored procedures are used not only to work with samples. The same approach is used to organize all interactions between the client applications and a database. The given way allows to make the client applications independent on a database internal structure and keeps them working even if we change structures of tables. Besides, in case of DBMS provides the division of access rights (essentially, it is already provided in all modern enterprise-level DBMS), it is possible to organize a restricted access to the tables by giving users the right only to execute stored procedures. It guarantees that the data will change accordingly and exactly as it was determined by the database designer. Many DBMS provide tracking for the references between the tables (Foreign Key), i.e. to check that records, to which someone refers, really exist and also to refuse removal of records to which there is a reference. In our case it is used to maintain the referential integrity between object and its class, class and its properties, stored sample and its conditions, links between objects and etc.

4. Discussion of a database concept

Main distinctive feature of proposed concept is that the process of database design and installation is separated from the analysis of the information and structuring of the data. As it was already specified, this feature can appear especially useful at the stage of data collecting and analysis during creation of information system, as it allows to keep any data, including significant volumes of the textual and graphic information.
This feature provides an opportunity to create and to change structure of classes easily, without participation of the designers and programmers, that transforms the given model of a data structure into the basis for creation of RAD tool of databank development. We will discuss creation of such a system and other plans of development later.

The obvious advantage of the proposed concept is an opportunity to create system with any connections between the objects of any classes with simultaneous preservation of referential integrity. The user can actually construct the graph of connections of any complexity without restrictions. The stored procedures for retrieval of all short links of the object and the information concerning given object provide convenient navigation in the information kept in a database.

To maintain the referential integrity in case of realization of a standard approach on the basis of relational DBMS, it is necessary to include some fields in the tables which contain the references. If relations between any tables are possible and the number of tables is large, it leads to the explosive growth of the number of fields, containing the references. As a consequence, there is unjustified growth of the database size, due to the significant volume spent on keeping the references, and fall of efficiency (substantial growth of the table width induces increase in time of response on query for some DBMS). Besides, the navigation in such a system is very difficult and demands the complete knowledge of a database internal structure. Thus, the suggested concept appears to be superior for construction of the system with complex internal relations.

Obvious advantages of developed system are its fullness and closeness. The fullness means that the whole functional part, i.e. tables and stored procedures, is already completed to the moment when data model designers start to work and does not require expenses on programming. The closeness means that any interaction is put into effect only by means of interface of stored procedures, which contains algorithms ensuring coordinated change of data. All above-mentioned allows to assume that the whole system can be provided to the users "on a turn-key basis".

The offered technique of object search and selection via stored samples is less effective than simple SQL-queries since the procedure of fulfillment of stored sample contains one additional level of enclosed query. However, the simple models of a data structure do not require the special approach and are realized easily with use of standard technologies. On the other hand, the proposed methods should provide increase of efficiency for data with complex interrelations in comparison with the complex queries of a high degree of an enclosure. The authors had to use queries containing up to 7 levels of enclosed queries.

Thus, the described concept provides fast installation of system for the end users, flexible data structure, simple in use navigation by links and also effective search and selection for complex data structure models containing a large number of types of data structures and advanced system of relations between them.

5. General structure of software

The database described in previous section is a platform which all software is based on (see Figure 6). As it is shown in a picture, this software is divided into three subsystems or layers:

- a database (bottom) layer giving the programming interface which consists of stored procedures;
- servers of COM-objects as an intermediate layer;
- the end user applications providing a visual interface.

Only the bottom layer is an independent one from environment as it can be realized with use of any SQL-compatible DBMS. All other layers are realized for "Windows NT" with the use of the COM (Component Object Model) specification. The COM defines the standard mechanism which helps one part of the software gives the services to the others. Server supports internal COM-objects which will be created in reply to the user's query to a database. After processing of the query the user will receive the reference to one of such COM-objects via the standard interface IUnknown. It will be necessary to receive one or several interfaces depending on a type of a query and required data to get an access to the results of a query:

- IEnumUnknown - to organize access to all database objects received as a result of stored sample;
- IEnumString - to organize access to the property names of the database objects;
- IPropertyBag - to organize access to the separate properties of a database object by name; value and the type of property returns as a VARIANT structure;
- ILockBytes - to organize access to the binary properties. The reference to the interface IUnknown is returned in a VARIANT structure for properties of such a type which is used in its turn to receive the interface ILockBytes.
Given component can be used for faster development of the applications together with a database as it hides the mechanisms of connection and communication with a database from a programmer. Together with the server of COM-objects we use the special application to setup it. In particular this application is also responsible for a choice of a database server. The following applications belong to the group of end user application which are responsible for work with a database structure and to browse the database contents: the database management application which is intended for restructure of object classes and change of its properties; an application to organize input of data in a database intended for editing the database objects, its properties and interrelations; an application for a data revision.

The data revision application is realized with use of a metaphor of hyper references as the given approach is natural representation of relations between objects. The opportunity to search the objects by a given set of ranges of property value (i.e. creation and fulfillment of stored samples) will be also included in this application.

6. Comparison with known models

6.1. Semantic (extended relational) models

There are two aspects of semantic modeling:
1. use extended relational models, e.g. RM/T [6];
2. use of semantic modeling at development of database.

Methods of designing based on semantic modeling use a high level design abstractions ("objects" and etc.). In such approach it is supposed that the world consists of objects, the objects have types and all objects of the same type have some common properties. Each object should have identity and can be connected to other objects by means of relationships. Also semantic models include some special relations. For example, in RM/T model these relations are as follows: Entity-Relations which represent objects and Property-Relations which represent properties.

For the models such as RM/T there is very important rule - "integrity of properties" i.e. each property should be a property of object. As follows from above-stated, semantic (or extended relational) models coincide with proposed model at this point since the classes of objects are direct analogy of types, each object has identity and rule of integrity of properties is not violated also. It is necessary to note that the term "relationship" used in similar models (for example, the object / relationship model, offered by Chen) [7]) does not coincide with the appropriate term of relational model. In this meaning the types of short links are the types of the binary relationships of the object / relationship model and short links are copies of such binary relationships.

However, the model offered the authors is much more simple as it does not support inheritance. Also it does not maintain the concept of "weak objects ", i.e. such objects which are depending on other objects (some analogue of aggregation in COM-model). Compound and multiple-valued properties are not supported also. The realization of absent opportunities would exceed needs of a particular problem making realization excessively complicated. As Date in [8] notes, the ideas of semantic model are rather useful at creation of data dictionary, i.e. "databases of the developer for a database creating by him ", in which he keeps the decisions accepted during design-time. In this meaning the proposed concept contains the data dictionary (classes, their properties and types of short links).

![Diagram of software structure](image_url)

Figure 6 General structure of software

Actually when the developer will have the complete information concerning the project (i.e. the filling of the data dictionary will be completed), he can start (if it will be necessary) to construct a relational database. What he should do is to put the relationship of relational model (i.e. table) in conformity to each class and external keys (when all objects of the given class have short links of given type) or separate relational relationships in conformity to types of short link between object classes. The preliminary creation of data dictionary does not relieve from necessity of the subsequent reduction of the relations created on its basis to higher nor-
6.2. Comparison with object-oriented models

Object-oriented DBMS approaches for the purposes of designing of factographic database most of all at the first look. As usual they represent a set of tools to work with DBMS (tools to support libraries of objects, method compilers and etc.). The shortage is that it is hard to insert a new object class in object-oriented DBMS without participation of the programmer, who should create internal structure of class and code for its methods. The basic idea of proposed model is to simplify a procedure of changing internal structure of database. Object-oriented DBMS supports hierarchy of containers, i.e. the class objects can comprise variable, representing the objects belonging to other classes. Therefore n-to-n relationships between classes of objects are built on the basis of common access (for example, two different classes “contain” common object of the third class) and the method of “reverse variables” (i.e. variable, containing the upward reference in hierarchy of containers). Thus, the addition of new types of relationships requires addition of new internal variable of class, change of the interface of class (addition of methods to work with new relationships) and, probably, change of code of existing methods.

Object-oriented DBMS poorly approaches to create a database which structure is continuously increased. The programmer should precisely imagine all details of the project writing a code for objects and methods. In our case we want to give an opportunity to add classes of objects and types of relationships without special efforts. Among other lacks of object-oriented systems we shall also note an absence of symmetric operations resulting in difficulties with fulfillment of queries unplanned by the developer. No doubts the object-oriented systems have a number of advantages. First of all there is essential possibility to support the several versions of one object. Thus there is an opportunity to export the version (transfer it to the workstation) and subsequently update it when modified version returns from workstation. Also there is a possibility to maintain archive of out-of-date versions and view a history of changes in versions. The concept of configurations (sets of the mutually coordinated versions of mutually related objects) can be simultaneously realized together with the concept of the versions.

The version control when implemented would be rather useful though the ways of realization of the given opportunities are only studied at the moment. OO DBMS give an opportunity to create the class - collection (i.e. class of objects, which copies contain sets of objects of other certain class). Each copy of class - collection needs the separate table in relational model. In our model that is possible by means of creation of one more class - collection for each class and definition of short link between class - collection and class - collection item which has a type “collection item of the given class”. Finally, when data dictionary and its analysis are completed it can appear that the best system for final realization of the project is object-oriented DBMS. However, the use of such DBMS will concern with large difficulties described above.

7. Conclusion

The concept of developing software that provides flexible organization of data structure, convenient system of navigation, effective search and selection of data was described. The first release of this software will be a toolkit ready to use by the end user. Besides the visual interface, program interfaces of two levels (stored procedures of a database and the server of COM-objects) will be put at user's disposal. One can use them to extend the software and to create additional applications for particular problems.

Development of the database table structures is completed at present, completion of stored procedures is nearly finished. The prototype of COM-object server was developed. It provides connection to a database and gives basic COM interfaces. Latter are used by the first version of the data revision application which uses a metaphor of hyper relations. The newest plans are to complete development of a database and a server of COM-objects, to start developing the applications providing restructure of classes, modification of objects, its relations and sections (in a sense of duplicating the sections and adding the sections from remote groups of database users). Further plans are to develop structure of classes to store the documents (texts, photos and etc.) with advanced system of the references and to analyze efficiency of created system on its basis.

References

3. "Integrity and Concurrency in Microsoft SQL Server Business Services Division, Microsoft Corporation, Microsoft Developer Network