Situation-oriented databases: using heterogeneous data sources

V. V. Mironov
Department of Computer Science and Robotics
Ufa State Aviation Technical University
Ufa, Russia
e-mail: mironov@list.ru

G. R. Shakirova
Department of Computer Science and Robotics
Ufa State Aviation Technical University
Ufa, Russia
e-mail: gulya_shakirova@mail.ru

V.A. Dolzhenko
Russian Foundation for Basic Research
Moscow, Russia

Abstract[1]

The paper is concerned with discussing one of the aspects of situation-oriented databases. The subject is about using heterogeneous data sources to fill the database. The main question discussed: main terms of situation-oriented database, main features of heterogeneous data sources and their interaction with the database.

1. Introduction

Since their foundation to the present time the databases permanently occupy a niche in the field of information technology. With the Web 2.0 inception and the upcoming Web 3.0 expansion theory and practice of databases have significantly strengthened their positions. Evidence to this is the new concepts and terms, which are firmly entered into use of experts in the field of information technology. So, often the term "infoware" is used instead of usual "software". It concerns any kind of application (both desktops and webtops). In this regard any research on the organization and use of database, no doubt, are relevant.

2. Situation-oriented databases

Nowadays the optimization of data storage mechanism is some kind of a vital necessity, 'cause too much things are related to the provision of information. Choosing a model of data representation and storage mainly determine the efficiency of the application. The range of known data models is broad, from classic relational to contemporary post-relational (dimensional, object-oriented, etc.). All the models are oriented on the representation of logically connected data and provide access and effective storage.

In the applications using databases, based on such models, supposes a number of typical operations:

- Connecting to data;
- Preparing application to obtain data;
- Data query for the applications;
- Data representation in the application;
- Data editing in the application;
- Verifying data;
- Saving data.

Each of these steps defines the databases as some kind of auxiliary component of the application, which supplies data by a query. All the situations of using data are moved solely on the level of application logic. The organization of the database does not take into account the situations of its further use.

It's suggested, that more effective database architecture implements the logical partitioning of data into groups according to some situation or the stage of its development. The suggested approach is supposed to be based on the situational approach, in which the data are associated with the internal states of the dynamic model and can be handled in the context of its current state.

The main point of the suggested approach is a situation. The term "situation" supposes a number of conditions and circumstances, which define an appropriate context. A situation is defined by the number of joined together stages, which are described as a dynamic model, represented by the hierarchy of graphs with transitions

[1] **Proceedings of the 14th international workshop on computer science and information technologies CSIT'2012, Ufa – Hamburg – Norwegian Fjords, 2012**

and finite number of states (so called "Finite State Model") [2–4].

The architecture of situation-oriented database includes [1]:

- Dynamic models library, which contains of a number of hierarchical situational models (HSML, HSM Library);
- Current state memory (CSM), which is intended to store data about the current states of the dynamic models;
- Associated data memory (ADM), which is used for storing XML data, associated with dynamic models states;
- Associated functions library (AFL), which is intended for storing data processing functions and for describing functions and methods, associated with dynamic models states;
- Dynamic model interpreter (HSMI, HSM Interpreter), which is intended to manage current states, to process associated with them data and to perform the appropriate functions.

The realization of situation-oriented database is suggested to be performed on the basis of XML platform: mark-up for describing model and data, methods of XML data processing – for interpreting dynamic model. There suggested a broad spectrum of methods for managing XML data in the database – from standard classes to dynamic DOM-objects [1].

3. Heterogeneous data sources in the situation-oriented databases

In the situation-oriented databases XML data, which are associated with various states of dynamic model, are represented in the special element – associated data memory (ADM). The complexity of ADM architecture is explained by its variegated character: data sources can be defined by various objects – from the files physically stored on the discs to virtual streams, supplied by other programs, systems or platforms (Fig. 1).

To fill associated data memory there can be used heterogeneous data sources. These data sources are divided into two groups:

- Internal objects (data, which are already stored in ADM);
- External objects (data, which are obtained from the other sources).

External objects include:

- External XML documents, which are the files, physically stored on the discs and identified by their full addresses;
- Databases, which can be based both on relational and post-relational models. Data can be extracted from the database with using special queries. So, dynamic models states can be connected with database requisites (database name, data model, data provider in necessity), and the query text, which is necessary to get data;
- Result of some function (or functions) call. These are the functions of associated functions library (AFL), which are identified by their unique name and can use input parameters;
- Web services. A term "Web service" means identifiable web address of a software system with standardized interfaces. Actually this is the code, which is available via HTTP or SOAP protocol and return information in XML format. On access a Web service there must be specified the name of one of its Web methods and if necessary set the values of input parameters. Therefore, in the situation-oriented database, an appeal to the Web service is characterized as the most essential elements of a Web service (Address, WSDL-description, etc.) and the called Web method (name, parameters, etc.).
- XML-oriented databases (for example, on the basis of database management system Sedna). Their structure is represented by the number of XML files. To obtain information from such a database there are used the queries of special type, based on languages XQuery and XPath, etc.

In the known architecture of situation-oriented database [1, 5, 6] the associated data are the part of the original hierarchy. These elements are defined by the special prefixes.

Generally speaking, the prefixes are a powerful tool in the situation-oriented databases: to identify the components of the dynamic model does not introduce special items. In the ideology of XML-hierarchies prefixes declared in the root element, along with the XML-namespace to which they relate. With regard to the implementation of XML-situation-oriented databases, this means that all the prefixes and namespaces are declared in a head (or main) state of dynamic models.

Data processing is performed using the standard mechanisms for handling XML (DOM-objects, classes XMLReader / XMLWriter, etc.). Actually the content of situation-oriented database is not a single document or a function. These are virtual objects generated in the process of working with a dynamic model and connected to the appropriate states of the model.

The difference depends on the purpose for which the virtual object will be used: whether it will be performed a lot of time to store the information and upload it, for example, to a file, or is it just an intermediate stage of processing for other data.

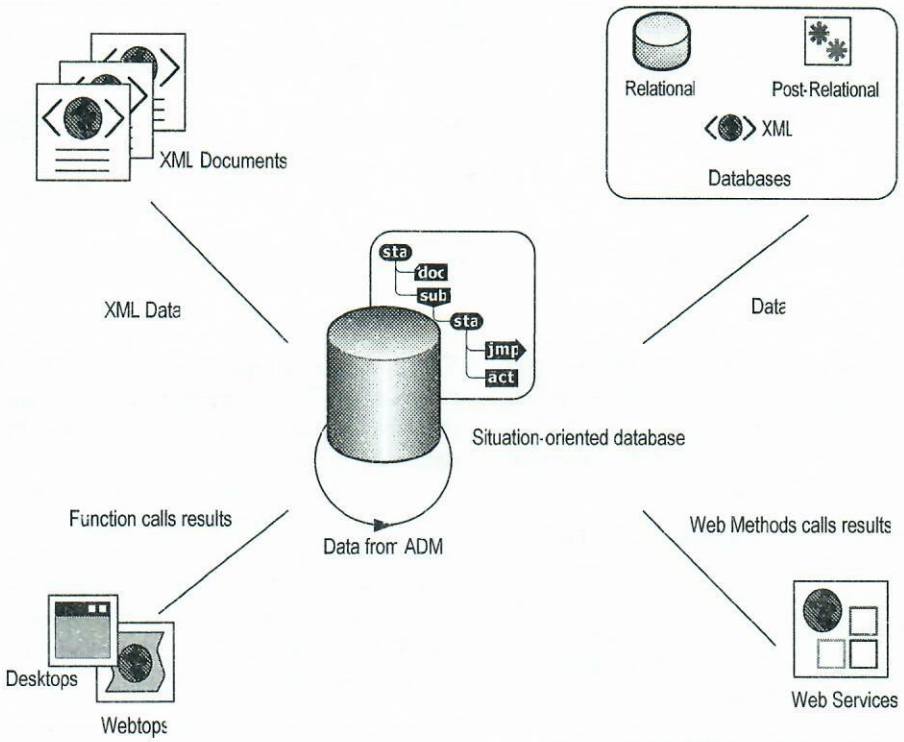

Fig. 1. Heterogeneous data sources for situation-oriented databases

Since the processing is based on the methods of handling XML-data, then these problems can be solved by non-cacheable or cacheable ways. In that case, if the loading of data into an object involves their use in the current session, they must "remembered", there should be chosen a cached method to implement – objects or classes of DOM.

DOM (Document Object Model) – a platform-independent object-oriented interface to access documents XML, XHTML, HTML, is currently receiving widespread. It involves creating an exact copy of the document in memory and XML-duplex operations with them (i.e., both read and write / modify).

If the virtual object is just an intermediate support element for further processing, it is recommended to use a non-cacheable data processing. For this XML provides classes XMLReader and XMLWriter. They do not load XML-document into memory, but simply move the cursor over the document tree in a given direction. These classes are working only in one direction – either read-only (class XMLReader), or write-only (class XMLWriter). Demonstration of the prefixes is shown on Fig.2.

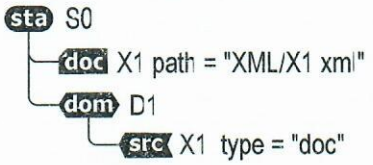

Fig. 2. A fragment of dynamic model of situation-oriented database

Here is a fragment of a dynamic model of situation-oriented database, represented by the head state with name S0. With the given state associated the document, identified as X1, which refers to the XML-document. The document, delivered in an appropriate state, denoted as an element "doc" (in the conceptual schema) named X1 and as XML-element names "X1" with the prefix "doc" (at the level of XML-markup). The path to the given XML-document is specified using the attribute "Path" (in this case it refers to the XML-document "X1.xml" in the directory "XML", it is relative to the directory with ADM situation-oriented database).

Virtual object that loads data from a data source (in our case – from the XML-document) is represented by an element of type "dom" (in XML-markup this is an XML-element with the prefix "dom"). In this case, this element means the DOM-object that is dynamically generated by changing the current state (when the state S0 is current).

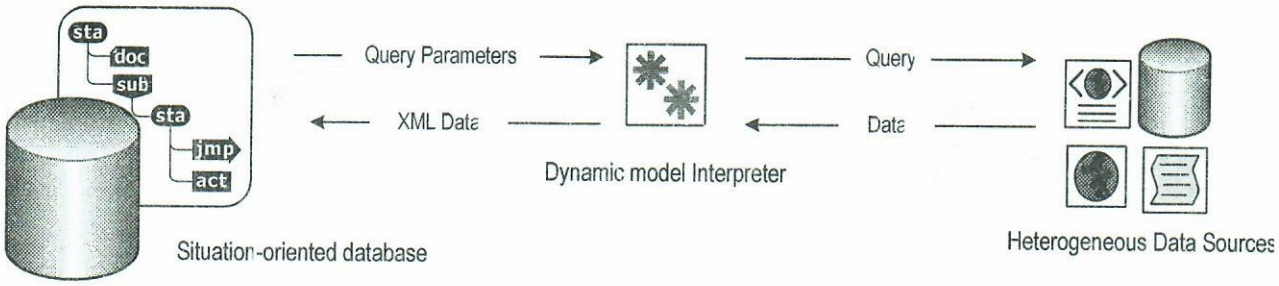

Fig. 3. Situation-oriented database's interaction with heterogeneous data sources

To indicate the data source for the given DOM-object uses an element name "src" (i.e. "source"), which defines a reference to the type of doc with identifier X1. The fact is that the data source for the DOM is an XML-document, defined in the "type" attribute with a value "doc".

In the mentioned example, there is a picture of a dual use of data sources. On the one hand, the element type "doc" sets a reference to an XML-document, i.e. an external source. On the other hand, an element "dom" refers to an element "X1", which is already part of a situation-oriented database, i.e. an internal source.

4. "Mash-up" technology in situation-oriented databases

Heterogeneous character of data sources for the situation-oriented databases is another challenge: to ensure their uniform representation in the structure of the database. Accessing a situation-oriented database a user or an application must see the same any data associated with the current states of the model, regardless of where the data were obtained. This task should be entrusted entirely to the level of the dynamic model interpreter.

The essence of the process of a uniform presentation of information from heterogeneous sources is as follows. The algorithm provides a set of shell functions that play a role of data providers (or suppliers). The purpose of the data is connected to the specified data source, query for data, sending the query and processing the result.

The structure of the query to the data source is defined by its type. For example, to retrieve information from a relational database, the request must be SQL-typed.

If the required data is contained in one of the fragments of XML-document, it uses XPath-or XQuery-query.

For some relational databases queries may be represented by the so-called mapping schema, which will automatically return results in XML format and do not require any additional changes.

In simplified form, the user or application receives only the parameter values for the request. All other works data providers in the interpreter make their own.

The data source type is also a determinant in the result processing. The structure of the situation-oriented database is based on XML, so it implies that the received data request must be submitted in XML-like format. The translation function for information came from the source data into XML format also takes an interpreter. To provide this support service function that builds an output XML-tree, suitable for further processing of the resulting query data. Additional features need to be generated to test XML-tree on the correctness and validity (matching the specified XML-schema).

This data structure ensures the independence of situation-oriented database-specific data models – all converted into XML format using the interpreter (Fig. 3).

Using of situation-oriented databases, heterogeneous data sources important especially now, during the domination of the principles of Web 2.0 in the World Wide Web. Technology "mix" (mash-up) is actively used in modern web applications: web pages are formed together from many sources.

In fact it is a service that fully or partially used as sources of information, other services, providing customers with new functionality for work. As a result of such a service could also become a new source of information for other web mash-up services. Thus is formed a network of mutually dependent services that are integrated with each other.

By analogy to the situation-oriented databases, XML-contents of the memory associated data is formed from a variety of sources. All of this goes unnoticed for applications and users, who see a single picture of the data, hiding their heterogeneous nature.

5. Conclusion

1. Situation-oriented databases suppose a logical partition of data into groups in accordance with a certain situation or stage of its development. Practical value of this approach is the ability to manage data with models of high-level abstractions, programming logic with database interaction in the structure of the database itself.

2. Architecture of situation-oriented database includes a dynamic models library, which contains of a number of hierarchical situational models, a current state memory, an associated data memory, an associated functions library, a dynamic model interpreter.
3. The complexity of situation-oriented database architecture is explained by its variegated character: data sources can be defined by various objects – from the files physically stored on the discs to virtual streams, supplied by other programs, systems or platforms
4. The interpreter of the dynamic model of situation-oriented database uniformly submit any data associated with the current states of the model, regardless of where the data were obtained. The basic format for the presentation of data obtained from a query is XML.

Acknowledgements

The research is supported by the Russian Fond of Foundation Research grant № 10-07-00167-a.

References

1. Mironov V. V., Gusarenko A. S. "Situation-oriented databases: the concept of XML-data control based on dynamic DOM-objects". In: *Vestnik USATU*. 2012. Vol. 16, № 3 (48). P. 159–172.
2. Mironov V. V., Malikova K. E. "Internet applications on the basis of dynamic models: architecture, data structure, interpretation". In: *Vestnik USATU*. 2010. Vol. 14, № 2. P. 154–163.
3. Mironov V. V., Malikova K. E. "Internet applications on the basis of dynamic models: the user interface controls". In: *Vestnik USATU*. 2010. Vol. 14, № 2. P. 170–175.
4. Hierarchical data model: the concept and implementation based on XML: monograph / V. V. Mironov, N. I. Yusupova, G. R. Shakirova. M: Mashinostroenie, 2011. 411 p.
5. Mironov V. V., Yusupova N. I., Shakirova G. R. "Situation-oriented databases: concept, architecture, XML-implementation". In: *Vestnik USATU*. 2010. Vol. 14, № 2 (37). P. 233–244.
6. Mironov V. V., Yusupova N. I., Shakirova G. R. Situation-oriented databases: external view based on XSL. In: Vestnik USATU. 2010. Vol. 14, № 1 (39). P. 200–209.