Conception of Situation-Oriented Databases

V. V. Mironov
Department of computer science and robotics
Ufa state aviation technical university
Ufa, Russia
e-mail: mironov@list.ru

N. I. Yusupova
Department of computer science and robotics
Ufa state aviation technical university
Ufa, Russia
e-mail: yussupova@ugatu.ac.ru

G. R. Shakirova
Department of computer science and robotics
Ufa state aviation technical university
Ufa, Russia
e-mail: gulya_shakirova@mail.ru

Abstract[1]

The paper is concerned with discussing situation-oriented approach within new class – situation-oriented databases. The main principles of situation-oriented databases organization are formulated. Also architecture and structure characteristics of situation-oriented databases in its conceptual context are defined.

1. Introduction

Today any activity area supported by information technologies is impossible without databases. Their role is much more than just information accumulation and storage – often databases are used for decision making support, are treat like sophisticated intellectual systems. Databases are the reflection of any innovations in hardware and software. New standards, new technologies, and new formats – all are inevitably mention in database sphere. That reason explains wide variety of database classes, and tags of their classification.

One of the key signs, which are used for database classification, is type of data model, which provides the data management. In that context there can be defined hierarchical, net, relational, object-oriented databases, based, accordingly, on hierarchical, net, relational and object models.

In many cases business processes modeled in the databases can be described as situation-oriented. In other words they can be interpreted as some situation states changing. For example, process of technical object projecting can be defined as collection of consistently / parallel way executable states: a requirement specification, the preliminary design, and the equipment design. A part of information must be inserted on the one state, the other – on the next one, etc. In this context appears the necessity of creating and manipulating data, characterizing the appropriate states of projecting situation development.

Today on the level of database management systems (DBMS) the mechanisms of data management according to the current situation are unknown. The traditional approach to database organization is not oriented on situational «filtration» of user data. This problem is to be solved on the application level. In this context it's suggested to apply to databases the technology of embedding dynamic models. This approach was applied in systems of different kind – from systems of technical objects management to electronic documents [1, 4]. During developing and research of dynamic documents (that means documents with in-built dynamic models describing main states of their life circle or using) was investigated that there must be a special class of databases, based on dynamic (situational) model.

This paper is concerned of describing conception of situation-oriented databases driven by inbuilt dynamic models, and the possibilities of its XML-realization.

2. Situation-oriented databases conception

Definition. Situation-oriented database (SODB) is intended for manipulating data and describing some subject area situation developing on two levels:

- macrolevel – as the integrated (macro-) states;
- microlevel – as detailed (micro-) states, associated with the macrostate

Proceedings of the 12th international workshop on computer science and information technologies CSIT'2010, Moscow – Saint-Petersburg, Russia, 2010

and its differing feature is concerned with supplying more convenient data access to the applications by such components as:

- dynamic model of macrostates, defining the situation development on the macrolevel;
- possibility of active behavior for the external events for describing current macrostate;
- supplying access to the microstates in the context of current macrostates.

This definition contains a number of key terms: two-level situation model; dynamic macrostates model; active behavior of situation-oriented database; situational data access. These questions are described below.

Two-level situation model. Discussion of situation-oriented database is important to be anticipated by defining its basic term – «situation». The situation is a number of conditions and circumstances, defining the particular situation, state.

It's supposed that the subject area, modeled in the database, can be described as a development of some situation that means the changing of its current states. In this context situation model can be defined on two levels. On the upper one – the macromodel of integrated states, describing quality situation development as a finite set of states with jumps defined by the set of rules. On the lower one – micromodel of detailed states defining situation with more details: each situation macrostate is described by the set of its microstates.

For example, considering business process of some product projecting as a situation development on the macrolevel we can use standard project steps as macrostates: requirement specification, equipment design, contract design, etc., and separate substates of that states. So the process of situation development is the pass from one current macrostate (state) to another; from one substate to another. By this situation macrostates are the sets of the appropriate properties, defining the appropriate macrostate: date of decision making; persons, making decisions, technical documentation, prepared on some states / substates, etc. It's clear, that storage of that data can be realized in the traditional databases, relational, for example. But in this case the problem of describing current macrostates and access organization for the appropriate microstates is to be solved on the level of applications, working with databases. But we are going to put these tasks mainly to the DBMS, managing data.

Dynamic macrostates model. It's necessary for SODB to consider current states by the situation states model (situation development model). We assume that the situation development on the upper level can be defined by the finite state model (FSM) in any form (states graph, Petri net, etc), that is as the finite set of states and set of rules describing conditions for changing current states from one to another according to the external data. That's why the FSM-driven database (FSM-DB) can be considered as situation-oriented. In this paper FSM is considered as so called hierarchical situational model (HSM).

In this case stored states are the information about which FSM states are current (and which were in the past), and the stored data are associated with FSM states. Stored states and stored data are defined in the current states model (CSM). It is a part of the SODB intended for describing situation current states change.

Concerning microlevel we suggest that each situation macrostate can be brought in correspondence with an appropriate data set, characterizing data microstates of macrostates. It's also suggested that the data development can be premeditated by the definition of schema (structure, limitations, that is data model). So microstate's change within its macrostate is evinced by data values change limited by schema.

For example, defining macrolevel of product development situation, in the simplest case we make states graph with macrostate (development stage) as a node and jumps between that states – as arcs. Each node is followed by properties set, characterizing this stage.

Thereby if on SODB macrolevel its behavior (microstates change) is quite strongly limited by inbuilt dynamic model, then on microstates dynamic restrictions are not supposed: microstates values are limited by static restrictions according to the appropriate schema.

On passing to new macrostate associated microstates are initializing according to schema. After that application access to microstates opens to create, update and delete the appropriate values. It's possible that start values of some macrostates inherit final values of definite microstates of past macrostates. This behavior is to be defined by schema. Thereupon it's important to outline temporal SODB, supposing storage of both current and past states that is tracing prehistory of situation states.

Situation-oriented database active behavior. By a definition, database is active, when it has some premeditated processes, changing the database state. These processes are set up not by user (or application) initiative, but by the system mechanisms of the database as an answer for the some premeditated conditions (or events). The SODB active behavior is associated with the rule called «event-condition-action» (ECA). This rule is concerned with the following. At the defined event the condition defined in the rule is checked; in the case if this condition is entry, the defined action is also performs.

Special organization of SODB gives its own meaning to the events and conditions terms. In this context the event is concerned with FSM current states change, and also operations of associated (with states) data creating and manipulating, etc. The action in SODB is a set of the processes defining FSM current states change, forced current state change (FSM internal predicates), «moving» associated data values from one state to another, etc.

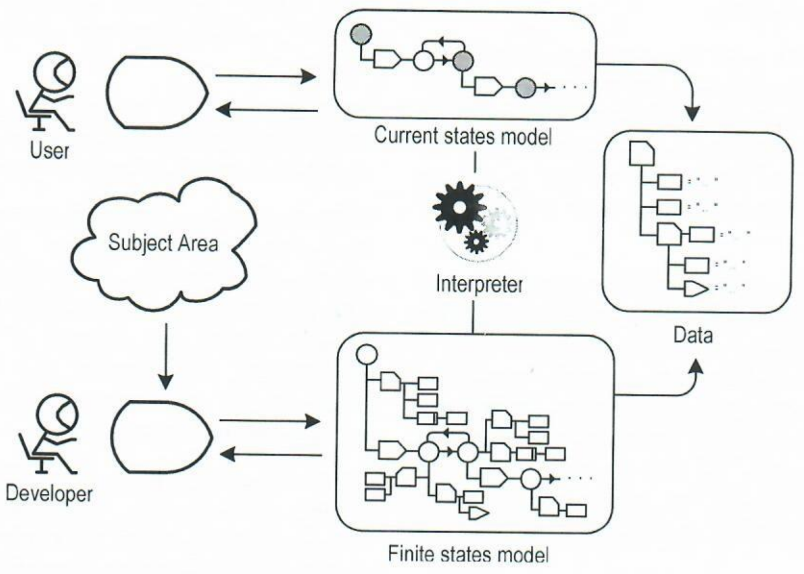


Fig. 1

Thematic example of ECA-rule is concerned with CSM management in FSM. The base event for this rule is the operation of state pass set up, the condition is a possibility of performing that pass (predicates), and the action is creating a new state in the CSM.

The active behavior is to be shown on both SODB levels:

- on the macrolevel – by tracing FSM current states change and it's clamping in the CSM;
- on the microlevel – by accumulating the associated data values on the current states change.

Data access in the situational context. In the extreme cases SODB must provide some traditional functionality that is an access to the stored data irrespective of the situation current state. In this context there can be defined:

- situation-dependent data, appropriate for any situation current states and demanding current state control for providing access to them;
- situation-independent data, appropriate for any situation current states and not demanding current state control for providing access to them.

In the first case SODB provides to user some advantages that are concerned with fact that the SODB performs situation current state management, and also provides data access in the current state context.

In the case of situation-independent type of data organization the information can be inserted both on the start state of some situation and updated during situation development.

It's possible there to provide access dualism: all the same data can be selected both by situation-dependent queries considering current states and by traditional situation-independent queries not considering model current states.

Situation-oriented database development and manipulation. Operating with SODB can be represented as two-step process (Fig. 1). On the first (start) step database developer defines the FSM structure based on the premeditated subject area analyses, describes data structure and assigns a correspondence between model data and states. The next step is concerned with generating database copy – CSM, which is intended for track user moving between model states and define associated data values.

Situation-oriented database realization platform. SODB organization principles define selection of its realization platform (and certainly developing management system for such kind of database). That's why it is reasonable to perform SODB on XML platform according its two main advantages: variety of XML technologies and wide spectrum of their possibilities for describing and manipulating data; hardware and software independence of XML data.

2. Architecture and structure characteristics of situation-oriented database organization

Typical database architecture is defined by the set of two invariable components – metadata and data itself.

Data is the information accumulated and stored in the database. By one of the definitions, data are defined facts, which logically provide getting new information [3].

Metadata is higher level data which describes structure of data stored in the database, «data about data». These are additional, reference information about the data. Metadata is the object providing DBMS information for data management and internal processes in the database, their relations and way of use.

Representation of both data and metadata depends on data model, used by the database. For example, relational database's metadata provides relational structure and contains descriptions of tables, fields, indexes, rules and other components of the database. The data themselves are represented also in relational form as a set of connected tables.

Data and metadata organization defines mechanisms of data describing and processing – data definition language (DDL) and data manipulation language (DML). DDL is intended to describe information in basic conceptual schema and it's following transforming into appropriate object form. Data processing supposes selection and manipulating information in the database, and also inserting there new data. In other words, it is manipulation of information, stored in the database, via DML.

As it was mentioned in SODB metadata is FSM, which defines a sequence of available states and describes structure of data, linked to states.

DDL and DML perform two kinds of operations – develop FSM and tracking CSM. According to its possible XML realization data defining and manipulating supposes operations of creating and manipulating XML-structures on the basis of appropriate XML-technologies (DOM-objects, XSL-transformation, XML Schema, etc).

SODB class model can be represented as a set of three models (Fig. 2):

- dynamic model (FSM mentioned before) defining situational component of the database;
- associated data schema – its informational component;
- current states model – user copy of situational model with data values.

According to XML-organization of SODB each one of that components must have object XML-structure. Technologies of XML-platform supply a wide variety of possibilities for describing and manipulating data content and structure.

Finite state model (dynamic model). Finite state model (FSM) as a situational component of the database defines a set of the states, integrated to submodels, connected together with jumps and containing dives into other submodels.

FSM can be considered as copies set of three object types – submodels, states and jumps (Fig. 2). One of the states is called head. It contains dives into submodels defined by the set of states (submodel states). At the same time the submodel structure doesn't depend on the state type – it's identical both for head and different states. Additional components of FSM are associated data schema and current states model.

Object structure of XML language allows for each type of FSM objects to put in correspondence its own XML-element and their properties can be represented as XML-attributes.

Diagram of FSM XML-elements is shown on fig. 2, *a*. The scheme uses notation used for conceptual description of database models [2] and XML-structures [5].

FSM elements are concerned to be close type. That means that their copies can't contain attributes and embedded elements besides those, which were defined in the model. The diagram represents the close type property by dark shadow behind the contour of object type graphic presentation. The diagram also describes three main properties of each model object type – multiple, compulsion and sequence order.

FSM elements multiple values ability defined allowed number of their copies. Single elements, supposing the only copy in the model, are outlined by the circle in the end of the extension line, connecting these elements with parent element, multicopies elements – with arrow-triangle. So, for example, in the submodel all elements (excluding data schema) are multicopies type.

Among the FSM states only head state submodels are required, that is shown by dark symbol in the end of the extension line, connecting graphic representation of model with head state. Optionality of other elements is shown by white symbol in the end of the extension line, connecting these elements with parent element.

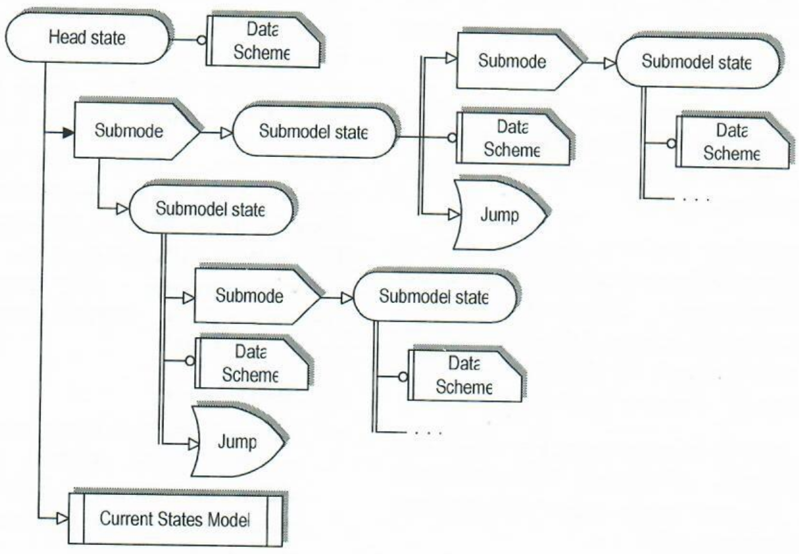
Fixed sequence order of the elements is typical also for the head state. Optional sequence order of other elements is shown by parallel lines, grouping them within parent element.

Associated data schema. Data schema within dynamic model defines definition of structure of data, associated with FSM states.

Data scheme supposes four types of objects: simple and complex elements, referential elements and description elements. Simple elements (by analogy with simple indexes in databases) are elements, which contain only single values. Complex elements don't contain values, but they contain sets of simple data elements.

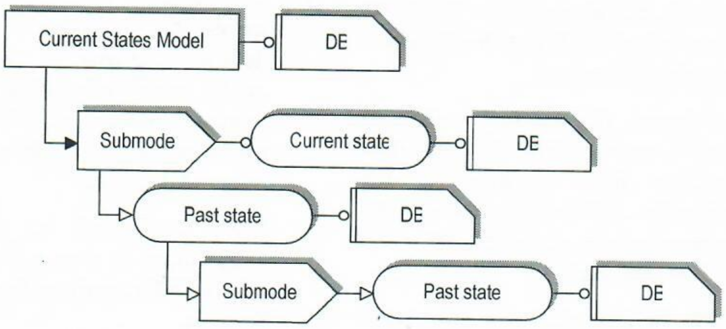
Description elements can be considered as some kind of vocabularies – they contain simple and complex elements, describing some class of subject area objects. One of the simple elements there is supposed as identification value – that means a value, given unique characteristics to its copy. Referential elements are values, referencing for that identification values.

Diagram of data scheme elements is shown on fig. 2, *b*. It uses the same notation as in the description of FSM structure. XML-realization of data scheme is possible with XML Schema technology. This allows describing both structure of separate elements and defining restrictions for their values.



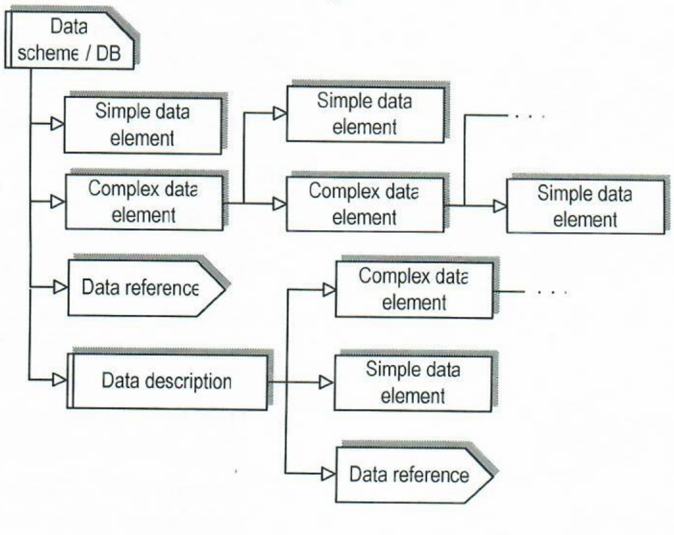
Head state
Data Scheme
Submode
Submodel state
Submodel state
Submode
Submodel state
Data Scheme
Jump
Submode
Submodel state
Data Scheme
Data Scheme
Jump
Data Scheme
Current States Model

a



Current States Model
DE
Submode
Current state
DE
Past state
DE
Submode
Past state
DE

b



Data scheme / DB
Simple data element
Complex data element
Data reference
Data description
Simple data element
Complex data element
Simple data element
Complex data element
Simple data element
Data reference

c

Fig. 2

Current states model. Current states model (CSM) is the only SODB component, providing non-system information. User manipulations with SODB, in final, come to the tracking CSM copy, where all passed states and data, associated with them, are stored.

CSM structure is represented by three types of objects: root element, associated with FSM head state; submodels and states. Fig. 2, *c* shows a diagram of CSM objects, based on upper mentioned notation.

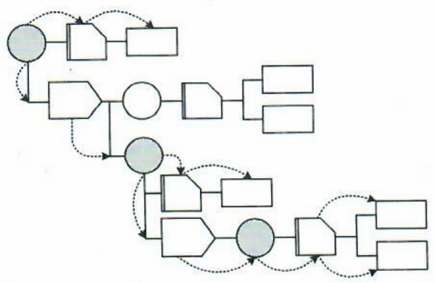
For every state (including the root element) can be defined object of type «database» (DB). By its structure this element is like data scheme – it also contains simple and complex elements, references and descriptions. The only difference is that this type of object is not just a structure, but also a set of values, defined by user.

CSM architecture supports «history» property – each new current state doesn't replace all the past, that allows both tracking sequence of user made steps and perform rollback to any other passed state.

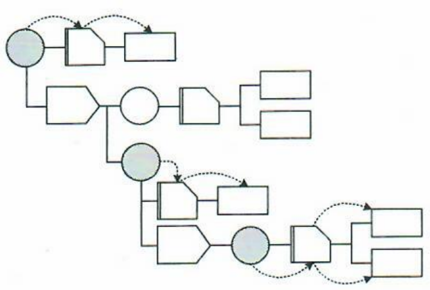
XML-realization of CSM supposes using hierarchy of XML-elements and attributes. So, for each user accessing XML SODB assumes creation of XML-document, where passed states and inserted data are stored.

Addressing. Important aspect of structure model is addressing data elements in user's CSM. For any data element in CSM hierarchy must be available it's «address», that is expression allowing access to data element. Key part of addressing is «entry point» – the node, from which the address emerged.

Addressing within XML SODB supposes two ways (Fig. 3).



a



b

Fig. 3

First one is concerned with the following. XML database as XML document supposes traditional absolute XPath-addressing. This address includes full path to the state, with which this data element is associated. At the same time full path to the state contains consecutive counting of all parent elements – from head CSM element to submodel with state in it (Fig. 3, *a*).

Second way supposes that addressing is situation-oriented. It means that address tracking is performed not from the hierarchy root, but directly from current states. That's why XPath-expression is added by logic of predicates tracking – «searching» submodels with actual value of attribute «current» and defining their current states (Fig. 3, *b*).

4. Conclusion

- Situation-oriented databases are the databases with inbuilt dynamic models. Their key advantage is ability of interacting with data in the context of situation, development of which is described by finite states model.
- Architecture of situation-oriented database is represented by three components – dynamic model, defining sequence of available states in finite state model terms; associated data scheme, limiting data structure, associated with model states, and currents states model, describing user copy of situation-oriented database.
- XML realization of situation-oriented database architecture is intended for representing describing data in XML markup form, and limiting – via XML Schema.

Acknowledgements

This investigation is supported by the grants 10-07-00167-a and 09-07-00408-a of Russian Foundation for Basic Research, President of Russian Federation grant НШ 65497.2010.9 for leading scientific schools.

References

1. Yusupova N. I. Critical situations and decision making in interferences: monograph. Ufa: Gilem, 1997. 112 p (In Russian).
2. Mironov V. V., Yusupova N. I. XML-technologies in databases. Ufa: Ufa state aviation technical university, 2004. 182 p (In Russian).
3. Deit K. J. Introduction to databases. M.: Williams, 2001. 1072 p (In Russian).
4. Mironov V. V., Shakirova G. R. Electronic documents with inbuilt dynamic model on the basis of XML: monograph. Ufa: Ufa state aviation technical university, 2009. 179 p (In Russian).
5. Mironov V. V., Yusupova N. I., Shakirova G. R. XML-technologies in electronic documents. Word documents. Ufa: Ufa state aviation technical university, 2009. 208 p (In Russian).