Distribution of the Object Oriented Databases. A Viewpoint of the MVDB Model's Methodology and Architecture

Mircea A. MUŞAN, Marian P. CRISTESCU, Daniel I. HUNYADI,
Lucian Blaga University of Sibiu
Marius POPA, C.S.I.E. Faculty, A.S.E. Bucharest

In databases, much work has been done towards extending models with advanced tools such as view technology, schema evolution support, multiple classification, role modeling and viewpoints. Over the past years, most of the research dealing with the object multiple representation and evolution has proposed to enrich the monolithic vision of the classical object approach in which an object belongs to one hierarchy class. In particular, the integration of the viewpoint mechanism to the conventional object-oriented data model gives it flexibility and allows one to improve the modeling power of objects. The viewpoint paradigm refers to the multiple descriptions, the distribution, and the evolution of object. Also, it can be an undeniable contribution for a distributed design of complex databases. The motivation of this paper is to define an object data model integrating viewpoints in databases and to present a federated database architecture integrating multiple viewpoint sources following a local-as-extended-view data integration approach.

Keywords: *object-oriented data model, OQL language, LAEV data integration approach, MVDB model, federated databases, Local-As-View Strategy.*

1 Introduction

Object-oriented databases are becoming more and more popular for applications to support the complexity and the irregularity of the real-world entities. Moreover, with the expansion of the distributed technology and the Internet, new needs related to data sharing and data exchange appears. Thus, the development of advanced database models is required. Object-oriented technology seems to be the keystone of this evolution. Hence, much work has been done recently towards extending object-oriented database models with advanced tools such as view technology, schema evolution support, multiple classifications, role modeling and the viewpoint paradigm. All these extensions require more flexible and powerful constructs than are currently supported by existing object –oriented models [10].

In the conventional object-oriented database model, the conceptual structure, that is a schema, is embodied by a collection of abstract data types called classes. The unique and permanent bond between an object and its class forbids a dynamic evolution of its structure and behavior, or the representation of several points of view, independent or otherwise. However, in the real world applications, it's often useful to cope with a multiple and evolving modeling of objects. This perception mode of data is called the viewpoint approach.

The viewpoint paradigm is an active subject of research in many areas such as software engineering [1], knowledge representation [2], database systems [3, 4], web applications [5], etc. In DataBases (DBs), we notice few works on the integration of the viewpoint concept into the data models. Most of these works consider the view and the role mechanisms. Views [6, 7] are external schemas that provide the user with a part of the global schema, a kind of viewpoint on the description of its entities. Roles [8, 3, 9] deal with the multiple aspects that an object acquires and loses during its life-time within a unique representation. In the context of our work, viewpoints offer several descriptions to the same Universe of Discourse (UoD). Each description is not a view, but a partial representation of data according to a given point of view. The various partial descriptions are supported by database schemas that together

provide the global schema of the same real world data. Objects can be described according to one or more descriptions, as a kind of role within a multiple data representation. Achieving such an approach requires a distributed environment and, more precisely, a federated database system that permits the integration and the collaboration of a collection of databases.

In this paper, we report an ongoing research we are engaged in [6]. Our work is aimed at extending object-oriented database technology to accommodate multiple and distributed modeling of data. The paper is structured as follows. Section 2 provides an overview of the viewpoint approach used in the several fields of computer science. A comparison of the integration of the viewpoint paradigm in database modeling is given in Section 3. In Section 4 and Section 5 we present the methodology and formalization of the MVDB (Multi-Viewpoint DataBase) model, respectively. The proposed model is an extension of the conventional object data model with the viewpoint mechanism. It allows developing a schema as a multiple description of an UoD. This description consists of translating several abstractions of this universe, using a basic formalism for the multiple data descriptions. Section 6 presents the consistency and objects evolution in the MVDB model and we give the general architecture of a federated database system, called MVDB system, that uses an adapted LAV approach to integrate viewpoint sources. Section 7 concludes our work.

2. The Viewpoint Approach

In computer science, most of data modeling systems don't deal with the variety of perceptions related to the same UoD and develop tools to create a single model for a single vision of the observed world. The viewpoint approach is opposed to this monolithic approach and makes it possible to model the same reality according to different points of view.

The viewpoint approach is constructed on the conjunction actor/information. Therefore, it is necessary to include the actor in the action. We thus define a viewpoint as "a conceptual manner binding, on the one hand an actor who observes and, on the other hand, a phenomenon (or a world) which is observed". Many actors can observe the same UoD and produce various viewpoints on it. These last can be considered in several manners illustrated in *Figure 1*.

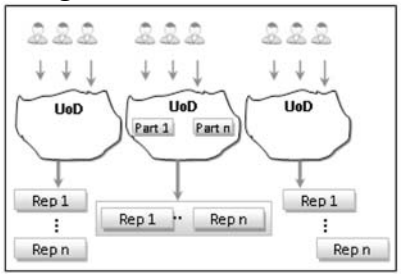

Fig.1. The different manners to consider viewpoints on an UoD.

Uniform viewpoints: in this case, all the actors have the same vision of the UoD and produce equivalent representations. For example, let us consider many research teams, each one uses a different data model and considers it as the best one to represent a project.

Complementary viewpoints: in this case, each actor sees a part of the UoD and provides a viewpoint on it. Each viewpoint is a partial and coherent representation. The various representations which rise from the various actors are complementary and their union is a complete and coherent representation of the UoD.

Comparable viewpoints: in this case, the actors produce comparable representations according to the generalization/specialization meaning. Within the framework of our study we are interested in the second interpretation of the relation "actor-world", which supposes that the various viewpoints on the same UoD are partial but complementary representations of it.

The viewpoint mechanism has been integrated into various contexts and used to solve different problems. Most works in the literature dealing with the viewpoint notion in object-oriented and conceptual modeling are much more pragmatic. In the following, we identify the main objectives in integrating viewpoints into computer systems. Note that

there is no single use of this concept that includes all of these objectives.

• The viewpoint as a means of providing multiple descriptions of an entity: the viewpoint concept seems to naturally result from the multiple views of objects of a specific study. As a matter of fact, a real world entity can have many behavioral contexts and many states from which the notion of multiple descriptions has been derived. Recently, the viewpoint paradigm has also been applied to web data in representing and viewing multidimensional information; that is information that may assume different facets under different contexts [5].

• The viewpoint as an approach for the modeling and distributed development of systems: many authors state that the modeling of complex systems as defined in cannot be handled with the same techniques as used for simple systems. However, the modeling of a complex system cannot be a centralized task based on a single formalism. Solutions based on logical systems are generally used to permit this correlation.

3. Related Works

In the field of databases, the concept of viewpoints is mainly investigated within the concept of views and roles in the object-oriented database community. Most of the research works propose enriching the monolithic vision of the traditional object-oriented approach in which an object belongs to one and only one hierarchy class. They deal with the objects evolution and with the existence of multiple views of the same data. In this section, we briefly examine some proposals which present roles and views, and then we present an overview of our viewpoint approach.

3.1. Views

Various view models have been proposed such as the multi-view model of [10] and the view model of [1] and of [7]. In these works, views are exploited to allow different applications to see the same database according to different viewpoints. The viewpoint concept here supports external schema, which is the third level of the ANSI architecture standard upon which the construction and the use of relational database systems and the later object-oriented ones are centered. Many problems arise, such as how a view schema (view class) is inserted in a global schema (class hierarchy) and whether an instance of a view owns an identity. A view can be treated as a database, but it does not preserve an object identity. Rundensteiner and Bertino [7] introduce the concepts of the multiview and schema view, respectively. These provide the capacity to restructure a database schema so that it meets the need of specific applications. They present support for view design by automating some tasks of the view specification process and by supporting automatic tools for enforcing the consistency of a view schema. Indeed, different views of the same object are allowed, depending on the context in which the object is considered. Here views preserve an object's identity, but the different instances of the same object are independent. All these models consider the viewpoint as a view defined with the aim of adapting an existing structure to new needs.

3.2. Roles

Objects with roles have increasingly been studied by several authors [8, 3, 10]. Roles are useful for supporting objects with multiple interfaces that can be dynamically extended to model entities which change their behavior, and the class they belong to over time. This task presents many problems such as uniqueness of objects identifier, strong typing, persistence, late binding, etc. in response to the role handling problem, several approaches have been introduced. In particular, the intersection-class-based and the role-hierarchy-based approaches are the most popular. The first approach simulates the objects multiple classification and dynamic restructuring by creating an intersection class to reflect the structure of a multiply-classified object. A separate class must thus be defined for every combination of roles. This simulation adheres to the constant that an object belongs to exactly one class at a time. This can present many problems: the class hierarchy may grow exponentially and the dynamic object classification is a tedious

task. The role hierarchy-based approach, however, has been adopted in many extended object-oriented database systems [10]. A role hierarchy is a tree of special types called role types. The root of this tree defines the time-invariant properties of an object. The other nodes represent types (roles) that an object can acquire and lose during its lifetime. The notion of roles is thus essential to support object extension, but is also useful to model situations where one real world entity may exhibit different behaviour in different contexts without changing its identity within a unique representation. Objects can therefore have several contexts, i.e. a kind of viewpoint that it acquires and loses dynamically.

4. The Methodology of the MVDB Model

The MVDB is an object-oriented data model with an extension by concepts and mechanisms which allow the multiple, evolutionary and distributed representation of a database schema. This representation confers to an UoD several partial and complementary representations. Each partial description is based on a first description of the entities and extends it according to a given viewpoint. The multiple and evolutionary representation overcomes the restriction of the single and fixed object instantiation link. The distributed representation fulfills the requirements of the current applications of the distributed and decentralized development of databases.

We adopted the object-oriented model as the common model for the various database schemas. This choice is justified by three principal motivations. First, the application of object-oriented concepts in system architectures provides a natural model for autonomous and distributed systems. Second, the object technology has been used in multidatabase systems to a finer level of granularity. Third, the expression and structuring power of the object-oriented approach goes with the objects modeling features in the MVDB model, such as the multi-instantiation mechanism that permits an object to have more than one instance.

The methodology of the MVDB model relies on the following ideas:

• the viewpoint concept is considered as an inherent concept of the data model and not as an augmented mechanism on it;
• a database schema is a multiple description of the same UoD according to various viewpoints. A database schema is thus viewed as a set of VP schemas, as shown in *Figure 2*. Each VP schema represents an aspect of the data description and is held by an independent database system;
• the VP schemas construction is based on a basic one called the referential schema. This last holds basic data on the real word entities shared by all the VP schemas;
• objects in the referential base are global. Global objects have a basic description in the referential base and one or more descriptions according to viewpoints;
• objects evolution is held by allowing entities to acquire or lose partial descriptions in the different viewpoint schemas while preserving their identities.

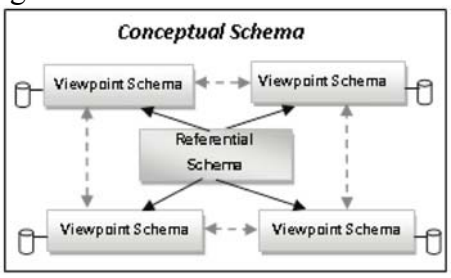

Fig.2. The viewpoint approach

We point out that object identity is a central notion in our approach. It is the same object described in many ways according of its membership in the various VP schemas. However, in order to ensure the components autonomy, local objects can be created and managed locally by VP databases. Local objects are objects with a single description according to one viewpoint and can't be accessed at the global level. VP databases are complementary and provide a global distributed database called multi-viewpoint database. A coherent exploitation of this global database is then recommended. Generally, these features are particularly needed in large complex applications of the industrial world. As a matter of fact, companies are logically distributed into offices, departments, working groups, etc.

Consequently we can deduce that the data are also already distributed. Each unit in the company must manage the relevant data for its operation and should be able, if necessary, to reach remote data that exist in the other units. The data in the various units are complementary and operated upon by collaborating users.

We illustrate the viewpoint approach through a simple modeling example. It concerns the representation of a laboratory's scientific staff (see Figure 3). This is composed of a referential schema and two viewpoint ones. The referential schema consists of the common information shared by all the viewpoints. We are particularly interested in the teaching and research activities of each member of the laboratory. Let us consider the Research VP and

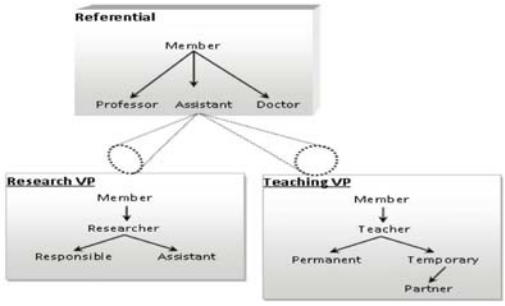

Fig.3. A multi-viewpoint modeling example.

the Teaching VP. Each viewpoint is an object-oriented schema that contains only information that is relevant to it. The Research VP, for example, is a hierarchical description of the laboratory's members according to their research activity. Each member can have, simultaneously, a basic description at the referential level and one or two viewpoint descriptions according to his/her teaching and research activities. For example, an one member is presented with oid "E1" in Figure 4, is a professor, permanent teacher and responsible of research topics. E11 and E12 are his identifiers at the VP schemas.

5. Formalization

The keystone of our modeling approach is the integration of the viewpoint paradigm. Thus, the conventional modeling concepts of an object-oriented database: schema, base (instance of schema) and objects are extended by the concepts of multi-viewpoint schema, multi-view point base and multi-viewpoint objects, respectively. Each one of these concepts contains two types of information:

1. Intrinsic information which represents basic and common objects description, shared by all the viewpoints.
2. Specific information which relates to objects description according to the various viewpoints. With these concepts, are added those of viewpoint schema, viewpoint base and viewpoint objects, which allow to model data at any viewpoint level. In this section, we develop the formal description of our data model. The presentation is inspired by [2].

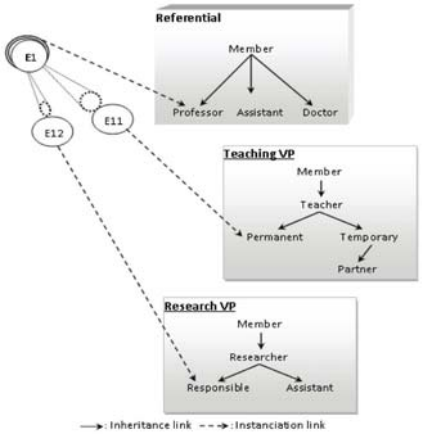

Fig.4. The multi-viewpoint object representation.

When applicable, each concept of the MVDB model comes with a syntactic formulation in a data definition language closely related to the O2 one [3]. Let MVDB (Sr, VP, C, O), be the specification of the data model signature, where:

- Sr is a referential schema name.
- VP is a set of viewpoint schema names,
- C is an infinite set of class names,
- O is an infinite set of object identifiers.

For any viewpoint, we specify:

- Svp is a viewpoint schema name, such that Svp $\in$ VP.
- Bvp is a viewpoint base name.
- Cvp is the set of classes in Svp, such that Cvp $\subset$ C.
- Ovp is the set of objects in a viewpoint base, such that Ovp $\subset$ O.

Example. The research-viewpoint schema refines the members' description by adding new attributes. All the members are concerned with this description here. The whole of the referential schema is thus imported.

The schema definition is:
Viewpoint schema Definition
Viewpoint Research-viewpoint from laboratory;
Base researchers-base;
Import-schema laboratory-schema class Member;
Import-schema laboratory-base name members;
Class Researcher from Member
Public type tuple (research-time : integer, research-Institution : string)
End;
Class Assistant . . .
Class Responsible . . .
End.

According to the property 1 on objects and their extension with the referent concept presented above, we give now the definition of an object in a viewpoint database called viewpoint object and in the referential one called multi-viewpoint object.

Definition 1. Viewpoint object.
A viewpoint object is a pair (Rl, Vl) where:
— Rl is its local referent
— Vl is its local state value.

Definition 2. Multi-Viewpoint object.
In a MVDB schema, an object is defined as a pair (Rg,Vg) where:
— Rg is its global referent
— Vg is its state value in the referential schema.

6. The MVDB Architecture

We have noticed above that the viewpoint approach to databases requires a distributed environment. Distributed systems [9, 5] have become increasingly important because of requests for organization and the growth of advanced techniques in the network management. These systems are characterized by three orthogonal dimensions: distribution, heterogeneity and autonomy. In this paper, we do not deal with the heterogeneity dimension. According to the autonomy dimension, [9] propose a classification most commonly applied to the distributed systems. These are divided into two families: non federated or tightly-coupled database systems and federated or loosely-coupled database systems. In tightly-coupled database systems all the various database schemas are integrated in only one global schema. The integration of the components makes these latter lose all their autonomy. Indeed, there is only one management level where all the operations are carried out in a uniform way. Then no distinction is made between the local and the global use of data. Thus, this approach does not meet the viewpoints structuring needs. As a matter of fact, a federated system consists of the integration of many autonomous and interdependent database systems. Thus, in contrast to the previous approach, a federated database does not support a global schema. Its main objective is to ensure the autonomy of the component databases and to privilege their management and their independent handling. The federation is an appropriate architecture to support the viewpoint approach. However, what about the data integration strategy that will be used?

In a federated system two strategies are used to integrate independent databases in a unified logical global schema: the Global-As-View (GAV) strategy that defines the global schema as a view over the local schemas and the Local-As-View (LAV) strategy that defines the local schemas as views over the global schema [9]. We are particularly interested in the LAV architecture that will be adapted to our architecture.

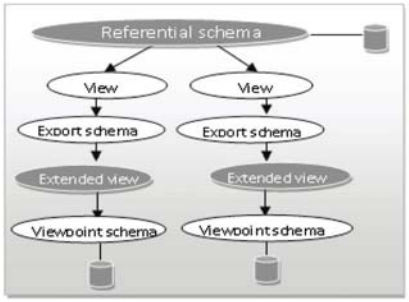

Fig.5. The LAV data integration approach.

The Local-As-View (LAV) strategy, presented in *Figure 5*, consists of defining the local sources as views over the global schema. This presents two principle advantages: a local change to a data source
is easily handled and the heterogeneity of the different components is supported. The LAV process is more adaptable to the data model we have defined above. However, in our case, local schema called viewpoint schema is an extended view over the global schema called the referential schema. We recall that a viewpoint schema is a partial description of data according to a viewpoint. A Local-As-Extended-View (LAEV) process is then used in our system (see Figure 6).

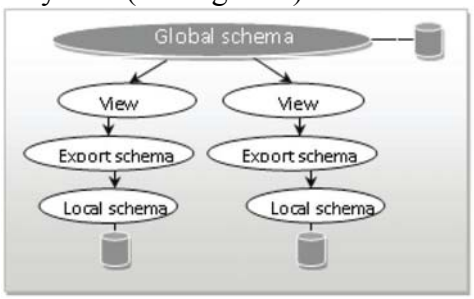

Fig.6. The LAEV data integration approach.

7. Conclusion

In this paper, we have proposed a structural object database model that integrates the viewpoint paradigm. This approach refers to the evolution, multiple description and distribution of objects. Also, it can make an undeniable contribution for the distributed design of complex databases. However, the same UoD can be described in a distributed fashion by different database schemas. Each one of these presents the entities according to a single viewpoint. A federated environment instead of a centralized one has been chosen to achieve our approach. Future work would concern the development of a data definition and manipulation language for the MVDB model, which is an extension of the OQL language. In addition, it would be interesting to develop an expression language to specify integrity constraints at the federation level.

References

[1] P. J. CHARREL, D. GALARETTA, C. HANACHI, B.ROTHENBURGER, Multiple Viewpoints for the Development of Complex Software. Proceedings of the IEEE Int'l Conference on Systems, Man and Cybernetics, (1993), pp. 556–561. Le Touquet, France.

[2] O. A. BUKHRES, A. K. ELMAGARMID, Object-Oriented Multidatabase Systems. Prentice-Hall, (1996), Englewood Cliffs, NJ.

[3] S. COULONDRE, T. LIBOUREL, An Integrated Object-Role Oriented Database Model. Data & Knowledge Engineering 42(1), (2002), pp. 113–141.

[4] H. NAJA, CEDRE: un mod`ele pour une repr´esentation multi-points de vue dans les bases d'objets. Doctoral thesis, University of Henri Poincar´e, Nancy 1, (1997).

[5] M. GERGATSOULIS, Y. STAVRAKAS, D. KARTERIS, A. MOUZAKI, D. STERPIS, A Web-Based System for Handling Multidimentional Information through MXML. Lecture Notes In Computer Science (LNCS 2151), (2001), pp. 352–365, Springer-Verlag.

[6] S. ABITEBOUL, A. BONNER, Objects and views, Proceedings of the Int'l Conference on Management of Data, ACM SIGMOD, (1991), pp. 238–247. Denver, Colorado.

[7] E. BERTINO, A View Mechanism for Object-Oriented Databases. Proceedings of the 3rd Int'l Conference on EDTB'92, (1992), pp. 136–151.

[8] A. ALBANO, R. BERGAMINI, R. GHELLI, R. ORSINI, An Object Data Model with Roles. Proceedings of the Int'l Conference on Very Large Database, (1993), pp. 39–51. Dublin, Ireland.

[9] FOUZIA BENCHIKHA, MAHMOUD BOUFAIDA, The Viewpoint Mechanism for Object-oriented Databases Modelling, Distribution and Evolution; Journal of Computing and Information Technology - CIT 15, 2007, 2, 95–110, doi:10.2498/cit.1000692

[10] G. GOTTLOB, M. SCHREFL, B. ROCK, Extending Object-Oriented Systems with Roles. ACM Transactions on Information Systems 14(3), (1996), pp. 268–296.