Journal of Computing and Information Technology - CIT 15, 2007, 2, 95–110
doi:10.2498/cit.1000692

The Viewpoint Mechanism for Object-oriented Databases Modelling, Distribution and Evolution

Fouzia Benchikha and Mahmoud Boufaida

LIRE Laboratory, Department of Computer Science, Mentouri University of Constantine, Algeria

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 databases, much work has been done towards extending models with advanced tools such as view technology, schema evolution support, multiple classification, role modelling and viewpoints. In particular, the integration of the viewpoint mechanism to the conventional object-oriented data model gives it flexibility and allows one to improve the modelling power of objects. The viewpoint paradigm refers to the multiple description, 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, viewpoint approach, referential schema, viewpoint schema, LAEV data integration approach, federated databases

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 appear. 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 classification, role modelling and the viewpoint paradigm. All these extensions require more flexible and powerful constructs than are currently supported by existing object-oriented models.

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 behaviour, 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 modelling 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 [12], knowledge representation [15], database systems [13, 24, 25, 26], web applications [16], 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 [1, 7, 33] 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 [4, 13, 17, 37] 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 modelling 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 modelling 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. In Section 7, we give the general architecture of a federated database system, called MVDB system, that uses an adapted LAV approach to integrate viewpoint sources. Section 8 concludes our work.

2. The Viewpoint Approach

In computer science, most of data modelling 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.

Currently, much interest has been shown in the viewpoint paradigm. This takes on various meanings according to how it's studied from the diverse standpoints of the different fields of computer science, eg. software engineering [12], knowledge representation [15, 31], database systems [14, 24], web applications [16], requirements engineering [23] and complex systems modelling [11]. Several terms have been assigned to this concept such as roles [27], aspects [30], perspectives [34], dimensions [16], and viewpoints [5, 12].

2.1. What's the Viewpoint Mechanism?

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.

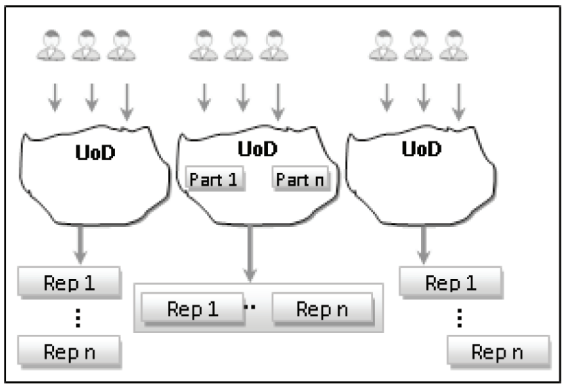

Figure 1. The different manners to consider viewpoints on an UoD.

1. 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.
2. 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.

3. Comparable viewpoints: in this case, the actors produce comparable representations according to the generalisation/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.

2.2. Why Using Viewpoints?

The viewpoint mecahnim 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 modelling 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 [16].
- **The viewpoint as an approach for the modelling and distributed development of systems**: many authors state that the modelling of complex systems as defined in [21] cannot be handled with the same techniques as used for simple systems. However, the modelling of a complex system cannot be a centralized task based on a single formalism. Different works suggest a distributed development approach based on viewpoints [14, 20]. Hence, every development process can be represented by correlated viewpoints. Solutions based on logical systems are generally used to permit this correlation. VBOOM (View Based Object-oriented Methodology) [20] is an example of an analysis/design method that integrates the viewpoint mechanism by defining the visual model concept. The need to use the viewpoint in modelling is also found in methods such as SADT [32].
- **The viewpoint as a mechanism for solving problems**: the viewpoint concept brought satisfactory and simple solutions to difficult problems found in different computer fields. In knowledge representation, for example, the viewpoint is introduced in the multiple classification of objects [5, 15, 31], in inherited value retrieval [28], in the modelling of independent concepts and in dealing with the multiple inherited conflicts [15]. In system modelling, explicitly considering the viewpoints of different designers in the production of a unique shared (single) model is an efficient means of improving the coherence of the modelling [11].
- **The viewpoint as a means of mastering system complexity**: several research works are based on the viewpoint concept with the principal objective of explicitly taking into account the complexity of the system. The result of the study is then held by dividing it into partial descriptions according to different and complementary aspects. Thus, in the context of a development environment, Schilling and Sweeney [36] describe multiple views as abstractions aimed at simplifying the complex structure of a system. Each view provides an interface adapted to a particular user (and/or developer) of the system. In addition, in the programming field, we find the EXPLAINER system [29] that describes programs according to different aspects (source program, graphical representation and so on).
- **The viewpoint as an advanced mechanism for object-oriented technology**: the use of object technology brought a real progress in the modelling of complex system through its powerful expression and its reutilisability. However, new needs have appeared such as considering the evolution of an object and its multiple and dynamic (re)classification. The strictness of the behavior and the state of an object have been reconsidered via the KRL perspectives [8], the CROME contexts [14] and the TROPES viewpoints [22].

In the context of our work, the viewpoint paradigm is used essentially as a means for object multiple descriptions (see Section 5) and evolution (see Subsection 6.2), as a mechanism for dealing with integrity constraint problems (see Subsection 6.1) and as an approach for the distributed development of a database schema (see Subsection 7.2). In the next section, we present an overview of related works integrating the viewpoint mechanism in database area.

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 [33] 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 centred. 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. Abiteboul [1] provides a general framework for view definition. A virtual class mechanism is used for instantiating views in object-oriented databases. Here, classes for views are explicitly defined where the attributes of these classes are really methods that retrieve information from where it is actually stored. A view can be treated as a database, but it does not preserve an object identity. Rundensteiner [33] 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 [4, 13, 17, 37]. Roles are useful for supporting objects with multiple interfaces that can be dynamically extended to model entities which change their behaviour, 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 [17, 37]. 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.

According to the above presentation of the two concepts of views and roles, we notice that the integration of the viewpoint concept in databases takes into account the actor and his vision to the system in a late stage. However, the viewpoint mechanism appears at data exploitation in the views case and represents the evolution of objects in the roles case. The aim of viewpoints is to take into account the actors earlier, i.e. at the data modelling and design step. The viewpoint approach that we consider deals with both the evolution and the multiple representation aspects within a unique paradigm. It also permits the novel dimension of distributed data description. To the best of our knowledge, no current object-oriented database system supports this aspect. In the next section, we present the methodology of our proposed MVDB model.

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 modelling 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 viewpoint is an abstraction of a certain perception (vision) of data. It does not correspond to a class, but it is a hierarchy of classes called ViewPoint schema (VP schema). 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;
- an exchange of information is supported between the partial descriptions of the schema, which are not isolated;
- 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.

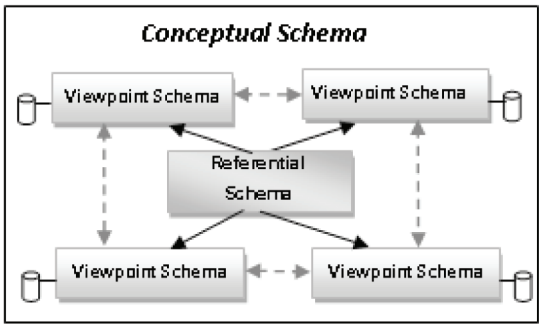

Figure 2. The viewpoint approach.

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 modelling 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 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, the member "Benali" presented with oid "E1" in Figure 4, is a *professor*, *permanent* teacher and *responsible* of research topics. E11 and E12 are Benali's identifiers at the VP schemas.

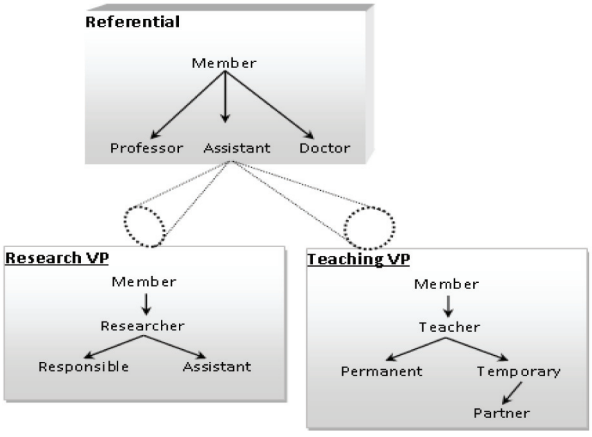

Figure 3. A multi-viewpoint modelling example.

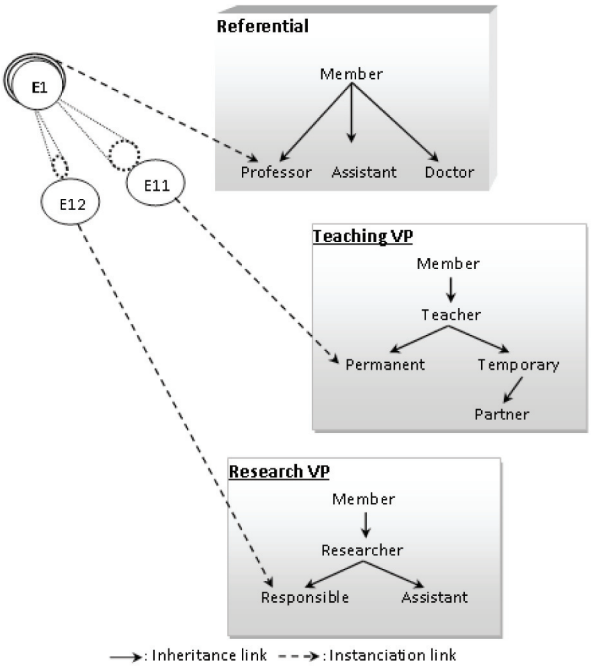

Figure 4. The multi-viewpoint object representation.

5. Formalization

The keystone of our modelling approach is the integration of the viewpoint paradigm. Thus, the conventional modelling concepts of an object-oriented database: schema, base (instance of schema) and objects are extended by the concepts of *multi-viewpoint schema*, *multi-viewpoint 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]. When applicable, each concept of the MVDB model comes with a syntactic formulation in a data definition language closely related to the O_2 one [3].

Let MVDB ($\mathcal{S}r$, $\mathcal{VP}$, $\mathcal{C}$, $\mathcal{O}$), be the specification of the data model signature, where:

- $\mathcal{S}r$ is a referential schema name.
- $\mathcal{VP}$ is a set of viewpoint schema names,
- $\mathcal{C}$ is an infinite set of class names,
- $\mathcal{O}$ is an infinite set of object identifiers.

For any viewpoint, we specify:

- $\mathcal{S}vp$ is a viewpoint schema name, such that $\mathcal{S}vp \in \mathcal{VP}$.
- $\mathcal{B}vp$ is a viewpoint base name.
- $\mathcal{C}vp$ is the set of classes in $\mathcal{S}vp$, such that $\mathcal{C}vp \subset \mathcal{C}$.
- $\mathcal{O}vp$ is the set of objects in a viewpoint base, such that $\mathcal{O}vp \subset \mathcal{O}$.

In the following, we first define formally some basic concepts of the conventional object-oriented database model: type, class hierarchy and subtyping. Our model supports three kinds of types: atomic types, constructed types and user types. Atomic types are: integer, string, bool and real. Constructed types are obtained from atomic types by recursive application of the following constructs: tuple, set and list. Given a class name n, n is a user type.

Assume the following sets:

- $\mathcal{D}om$ is a set of constants. $\mathcal{D}om = \text{Dom(integer)} \cup \text{Dom(string)} \cup \text{Dom(bool)} \cup \text{Dom(real)}$, containing the domain of each atomic type,
- $\mathcal{A}tt$: an infinite set of attribute names,

Definition 1. Type.

Let C be a finite subset of $\mathcal{C}$. The set of types over C, denoted Types(C), is defined as follows:

(a) any $\in$ Types(C).

(b) C $\subset$ Types(C). Class names are types.

(c) $\mathcal{D}om \subset$ Types(C).

(d) If $\tau \in$ Types(C), then $\{\tau\} \in$ Types(C). $\{\tau\}$ is a set type.

(e) If $\tau \in$ Types(C), then $[\tau] \in$ Types(C). $[\tau]$ is a list type.

(f) If $\tau 1, \ldots, \tau n \in$ Types(C) and a1, ..., an $\in \mathcal{A}tt$, then $<$a1:τ1, ..., an:τn$>\in$ Types(C). $<$a1:τ1, ..., an:τn$>$ is a tuple type.

Definition 2. Class hierarchy.

A class hierarchy is a tuple (C, ∂, $\prec$) where:

(a) C is a subset of distinct classes of $\mathcal{C}$.

(b) ∂ is a function from C to Types(C) which associates to each class ci of C a type.

(c) $\prec$ is a partial order over C, corresponding to the *Is a sub-class* link.

The sub-class relationships provide the conditions on types to establish if two classes are in a generalisation/specialization link. To formalize this concept, the subtyping relation is defined.

Definition 3. Subtyping.

Let (C, ∂, $\prec$) be a class hierarchy. The subtyping relation is defined as the smallest partial order $\leq$ over Types(C) such that:

(a) $\forall \tau \in$ Types(C), $\tau \leq$ *any*. *any* is the top of the hierarchy.

(b) $\forall$ c, c' $\in$ C, if c $\prec$ c' then c $\leq$ c'.

(c) $\forall \tau 1, \ldots, \tau m$, τ '1, ..., τ'n $\in$ Types(C) and $\forall$a1, ..., an, ... am $\in \mathcal{A}tt$, if n $\leq$ m and if τi $\leq \tau$'i for each i $\in$ [1,n] then $<$a1:τ1,..., an:τn, ... am:τm $>\leq<$a1:τ'1,..., an:τ'n$>$.

(d) $\forall \tau$, τ' $\in$ Types(C), if $\tau \leq \tau$' then $\{\tau\} \leq \{\tau'\}$.

(e) $\forall \tau, \tau' \in$ Types(C), if $\tau \leq \tau'$ then $[\tau] \leq [\tau']$.

Definition 4. Well-formed class hierarchy.

A class hierarchy (C, ∂, $\prec$) is well-formed if:

$\forall$c, c' $\in$ C, if c$\prec$ c' $\Rightarrow \partial(c) \leq \partial(c')$.

Relying on the previous definitions, we give now the formal definition of the MVDB concepts.

5.1. Schemas

In MVDB, a database schema is described by a referential schema and several viewpoint schemas.

5.1.1. Referential Schema

The referential schema is the basic schema that describes all the entities independently of any viewpoint.

Definition 5. Referential schema.

As any common object schema, the referential schema is a well-formed class hierarchy (see Definition 4)

$\mathcal{S}r = (Cr, \partial_r, \prec_r)$ where:

Cr is the finite set of class names in the schema such that: $Cr \subset C$. We notice that we don't deal with methods here.

Example 1. The laboratory-schema is the referential schema that models information about the laboratory's scientific staff presented in Figure 3. Each member is an object stored in the laboratory-base.

Referential schema Definition
Schema laboratory-schema;
Base laboratory-base
Class Member
Public type tuple (
family-name : string,
last-name : string,
age : integer **)**
End;
Class Profesor. . .
Class Assistent-profesor. . .
Class doctor. . .
End.
Name members = **set (**member**);**
Export-schema class Member;
Export-base name members;

5.1.2. Viewpoint Schema

A viewpoint schema is the customization of the whole or a part of the referential schema. Its definition is based on two operations: *projection* and *extension*, which are defined in the following.

Definition 6. Projection.

Let S = (C, ∂, $\prec$) and S' = (C', ∂', $\prec$') be two schemas. S' is a projection of S, denoted S' ∇ S, if the following conditions are verified:

(a) C' $\subseteq$ C.

(b) $\partial'_{|C} = \partial(\partial'_{|C}$ is the restriction of ∂' in S' on C in S).

(c) $\prec$ '$_{|C} = \prec$ ($\prec$ '$_{|C}$ is the restriction of $\prec$' in S' on C in S).

The projected schema S' is composed of some (or all) classes of S and preserves their type and their specialisation/generalization link in S.

Definition 7. Extension.

Let S = (C, ∂, $\prec$) and S' = (C', ∂', $\prec$') be two schemas. S' is an extension of S, denoted S' Δ S, if the following conditions are verified:

(a) C' $\subseteq$ C.

(b) $\partial'_{|C} = \partial(\partial'_{|C}$ is the restriction of ∂' in S' on C in S).

(c) $\prec$ '$_{|C} = \prec$ ($\prec$ '$_{|C}$ is the restriction of $\prec$' in S' on C in S).

(d) $\forall$c1, c2 $\in$ C', if c1 $\prec$' c2 $\Rightarrow$ (c1$\in$C $\wedge$ c2$\in$C) $\vee$ (c1 $\in$ (C'-C) $\wedge$ c2 $\in$ (C'-C)) $\wedge$ (c1 $\in$ C $\wedge$ c2 $\in$ (C'-C) $\wedge$ $\nexists$ c3 $\in$ C/ c3 $\prec$ c1)).

The hierarchy S' preserves the type and the subclass relationships of the classes of S (conditions (b) and (c)) and extends S by the creation of new classes as specialization of its classes 'sheets' (condition (d)). This constraint ensures the compatibility between the two hierarchies S and S'.

Definition 8. Viewpoint schema.

A viewpoint schema $\mathcal{S}_{vp} = (C_{vp}, \partial_{vp}, \prec_{vp})$ is an *extension* of a *projection* on the referential

schema $Sr = (C_r, \partial_r, \prec_r)$, i.e. $S_{vp} = \Delta\ (\nabla\ Sr)$) (as depicted in Figure 5).

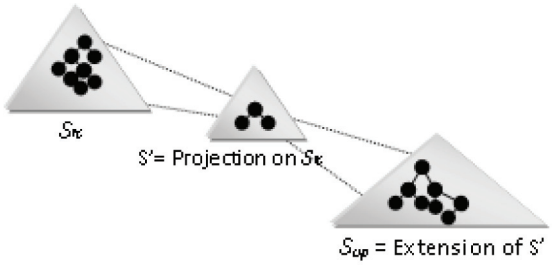

Figure 5. Viewpoint schema = Projection + Extension of the referential schema.

A viewpoint schema is obtained from two steps: first, a projection operation (∇) is carried out on the referential schema to select the part (or the whole) of it, which will be described according to the considered viewpoint. Then, an extension operation (Δ) of the resulting schema customizes the entities description according to the viewpoint. However, in order to support independence between the various viewpoint schemas, and to keep the specificity of each one, we choose a decentralized description. For that, we benefit from the "slicing technique" used in certain approaches described in [7, 33]. This technique consists of distributing the projected referential schema in the viewpoint schema.

Example 2. 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.

5.2. Objects

The various schemas (referential and viewpoints) hold all the persistent objects, instances of the different schema classes. Before giving a formal definition of them, we first present the extension to the object concept.

5.2.1. Objects Extension

In the original object model, an object corresponds to a pair (o,v) where o is the Object IDentifier (OID) and v its value. In this representation, each object in the database is assumed to have exactly a single class, that in which it was created. Such an assumption imposes some restrictions on the dynamic and multi-representation modelling of real world objects. These characteristics are crucial in advanced object-oriented technology. Indeed, multiple instantiation mechanism is proposed to overcome this rigidity. The multiple instanciation mechanism used up to now permits an object to belong to more than one class of the same database schema. In the context of our work, we address the two following object properties.

Property 1: An object is an instance of the referential schema and an instance of one or several viewpoint schemas.

Thus, an object has a basic description and may be described according to different viewpoints simultaneously. This is the most broadly accepted property of the viewpoint concept. Because an object in a viewpoint schema is seen as an instance of a viewpoint schema, it amounts to creating multiple descriptions of an object.

Property 2: The state of an object is viewpoint-oriented.

This means that the state of an object may vary depending on the viewpoint in which it is being described. This seems to suggest that each description of an object according to a viewpoint should be viewed as a separate instance of it. This property that allows the object multiple instantiations complements Property 1.

According to the afore mentioned properties, we can integrate a new concept called the *object referent*. We can distinguish a local object's referent from a global one.

Definition 9. Local object referent.

A local object referent, denoted by R_l, is the identification of the object in a viewpoint (local) database such that:

$R_l = (vp, O_l)$ where:

— vp is the viewpoint schema name,

— O_l is the viewpoint identification (OID).

Local object referents allow objects to be locally managed at the viewpoint level. It means that each VP database preserves its execution autonomy. However, a metadatabase is associated with this later to ensure local and global constraints handling when any update is done on objects (see the general architecture of the MVDB System in Figure 8).

Definition 10. Global object referent.

A global object referent, denoted by R_g, represents the identification of the object in the multi-viewpoint database such that:

$R_g = (O_g,, L(R_l))$ where:

— O_g is the referential OID of the object,

— L is the list of its viewpoint identifications that is: $L(R_l) = \cup_{vp} \in {}_{VP}(R_l)$.

The global referent is an important concept in our model. It permits the retrieval of a local referents list to access data of the same object in the VP databases.

5.2.2. MVDB Objects

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 11. Viewpoint object.

A viewpoint object is a pair (R_l, V_l) where:

— R_l is its local referent

— V_l is its local state value.

Definition 12. Multi-Viewpoint object.

In a MVDB schema, an object is defined as a pair (R_g, V_g) where:

— R_g is its global referent

— V_g is its state value in the referential schema.

5.3. Bases

The objects constitute the associated bases of the referential and the viewpoint schemas that are defined in the following.

5.3.1. Viewpoint Base

A viewpoint base contains viewpoint objects.

Definition 13. Viewpoint base.

A viewpoint base, denoted $\mathcal{B}_{vp}$, is a schema state specified commonly as a tuple $(\pi_{vp}, O_{vp}, \surd_{vp})$ where:

— π_{vp}: $C_{vp} \rightarrow O_{vp}$ is the function that associates to each class $c \in C_{vp}$ its object identifiers,

— O_{vp} is the set of the objects in $\mathcal{B}_{vp}$: $O_{vp} = \cup_{c \in Cvp}\{\pi_{vp}(c)\}$,

$\surd_{vp}$ is the function that associates a value to each object of $\mathcal{B}_{vp}$, such that: $\forall c \in C_{vp},\ \forall o \in \pi_{vp}(c)\,, \surd_{vp}(o) \in \mathrm{Dom}(\sigma_{vp}(c))$.

5.3.2. Referential Base

The referential base contains objects which are described at the global level.

Definition 14. Referential base.

Let $\mathcal{R}_r$ be the finite set of object referents of the referential schema $\mathcal{S}r$.

A referential base, denoted $\mathcal{B}r$, is a schema state specified as a tuple $(\pi_r, O_r, \surd_r)$ where:

— $\pi_r : C_r \rightarrow \mathcal{R}_r$ is the function that associates to each class $c \in C_r$ the object referents. C_r is the set of classes in $\mathcal{S}r$.

O_r is the set of the objects in $\mathcal{B}r$: $O_r = \cup_{c \in Cr}\{(\pi_r\,(c), \cup_{vp \in VP}(R_l))\}$,

— $\surd_r$ is the function that associates a global value to each object of $\mathcal{B}_r$, such that: $\forall c \in C_r,\ \forall o \in \pi_r(c),\ \surd_r(o) \in \mathrm{Dom}(\sigma_r(c))$.

The referential base and viewpoint bases constitute the multi-viewpoint base (extent) of the multi-viewpoint schema. They give the complete description of objects according to multiple viewpoints. Let us illustrate this by the following example.

Example 3. The associated bases of the referential and the viewpoint schemas presented in the above example are populated with objects. Each object has an instance in the 'laboratory-base' (referential base) and can possibly be instantiated in the 'researchers-base'. Instances are collected under the root of persistence of each schema as presented in follows.

Bases

laboratory-base((E1,{Benali, Mohamed,40}),
(research-vp, E11))
(E2, {Bencharif, Ali, 55}, nil))
researchers-base(E11,{Benali,Mohamed,40,
12,'Constantine-University'})

Once we have formally introduced the basic concepts of the MVDB model, we can finally define a database as an object-federated one composed of a referential part *Sr*, which is the core of the database augmented by a finite set of VP databases. A viewpoint component is defined to be a loosely coupled and locally managed object database.

6. The MVDB Consistency and Objects Evolution

In this section we focus a bit more on some aspects of the MVDB model such as: the consistency and objects evolution.

6.1. MVDB Consistency

The coherence is an essential and complementary functionality to every data model. However, unlike the traditional approach (mono viewpoint) where the integrity constraints are defined on the global schema, we distinguish in the viewpoint approach two types of constraints: local constraints which ensure the coherence at any viewpoint database and global constraints which ensure the global description coherence of the entities according to several viewpoints.

If the local constraints are well apprehended, it is difficult to take into account the global constraints. Within the framework of the MVDB model, the classical conflicts usually met in federated databases are solved by the viewpoint paradigm [6] such as names, semantic and structural conflicts. However, other types of conflicts are distinguished. These ones guarantee the compatibility of and the coordination between the different object descriptions in the system. Let us consider the following cases.

1. Mutual exclusion between viewpoint DBs: when the description of the entities by a viewpoint schema compromises their description by another viewpoint schema.

Example 4. Any temporary teacher does not have the right to acquire a research activity at the laboratory, and can't have a description according to this viewpoint.

2. Interdependency between viewpoint DBs: when the viewpoint schemas contain linked properties.

Example 5. Any permanent teacher whose research time exceeds twenty hours a week must reduce his official teaching time by 40 percent.

3. Referential integrity between viewpoint DBs: when the creation (or possibly the deletion) of a database entity requires a preliminary creation (or possibly the deletion) of one (or many) entity(ies) of another database.

Example 6. Any permanent teacher must be a member of a research group. This implies that the creation of any object instance according to the teaching viewpoint must generate the creation of the same object instance in the research viewpoint.

A multi-viewpoint base is coherent (here we are speaking about coherence with respect to the semantics of the applications and not about the implicit coherence induced by the model) if the referential base and each viewpoint base are coherent with respect to their schema, i.e. all the local constraints are checked, and if the gathering of the various bases is coherent, i.e. all the global constraints are satisfied.

6.2. Objects Evolution

In the MVDB model the multiple description of the objects is directly related to their evolution. While respecting the principle of the basic representation and the integrity constraints between the different viewpoints descriptions, an object can acquire new representations according to various viewpoints as it can lose others of them. This evolution, which is similar to the roles mechanism, consists of an addition or a deletion of an instantiation link between an object and the classes in the viewpoint schemas.

7. The MVDB Architecture

We have noticed above that the viewpoint approach to databases requires a distributed environment. Distributed systems [9, 10, 19, 35] 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, [35] 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?

7.1. Data Integration

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 [18]. We are particularly interested in the LAV architecture that will be adapted to our architecture.

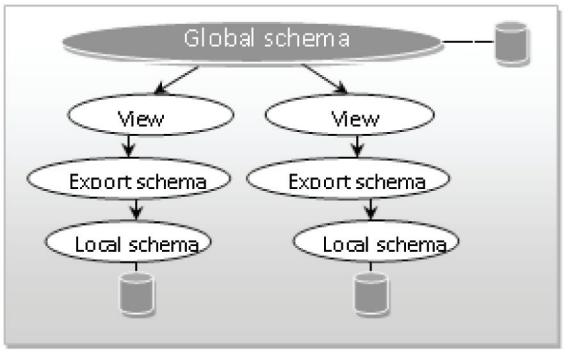

Figure 6. The LAV data integration approach.

The Local-As-View (LAV) strategy, presented in Figure 6, 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 7).

The next paragraph describes the basic architecture of the MVDB system which is a collection of local partial databases that cooperate in a federated environment.

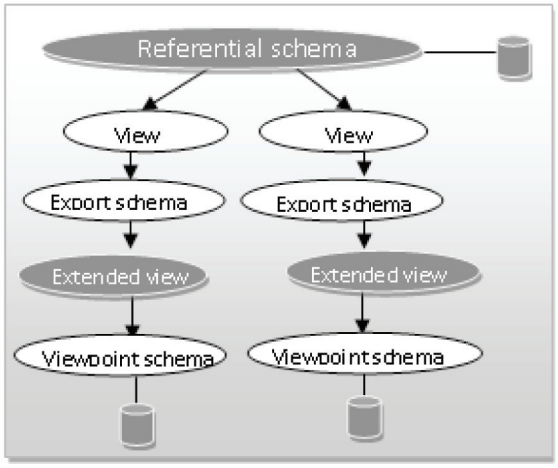

Figure 7. The LAEV data integration approach.

7.2. The Basic Architecture

According to the viewpoint approach, presented in Section 4, the global schema is designed in a decentralized way according to several viewpoints. Each viewpoint is held by an autonomous database. This autonomy promotes the independence of the component databases. It permits each one to keep a complete description of the entities according to a given viewpoint. Thus, the entities are described in a multiple but complementary way by several schemas. These latter share a basic description known as the referential.

The proposed architecture for the MVDB system is based on the federation following the LAEV process to integrate multiple autonomous viewpoint databases that show multiple descriptions of the same UoD. All the present schemas in this architecture, i.e. the local schemas, the referential schema and the external schemas are based on a unified common object model. The heterogeneity problem is therefore not dealt with here. The uniformity of the data model used is particularly important for managing of both the persistence and the identity of the objects in the federated base. The MVDB architecture is made up of three levels: the local level, the federated level and the external level (see Figure 8).

- The local level carries the partial independent object DBs called *viewpoint database systems*. Each system, which holds a particular entity description of the UoD, is autonomous. However, its local schema presents a complete data description according to a viewpoint. Moreover, a *metadatabase*, which stores visibility rules, is associated with each database in order to ensure autonomy of communication with the other bases.
- The federated level is the kernel of our federated database system. It is essentially made

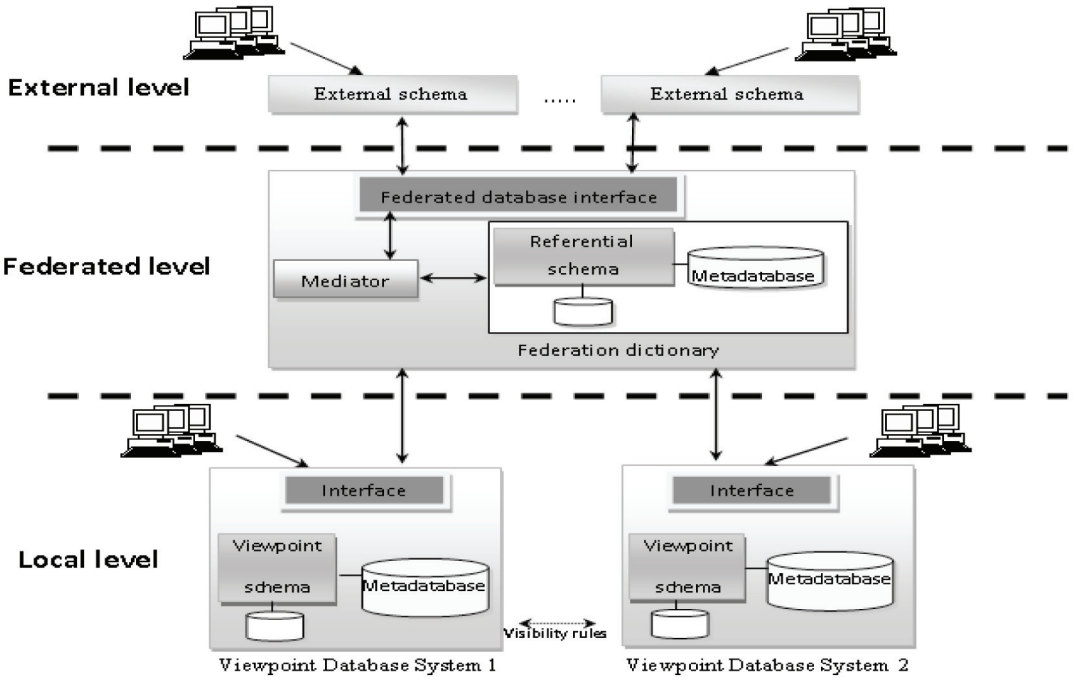

Figure 8. Global architecture of the MVDB system.

up of a *user interface*, a *mediator* and a *federation dictionary*. The user interface permits communication with the external level. The federation dictionary contains the *referential schema* and a metadatabase. Any database taking part in the federation imports a schema derived from the referential one and extends it with a particular description according to a given viewpoint. The derived schema can concern the entire basic schema if the partial description is related to all the entities of the UoD. The metadatabase is a component that has an important role in distributed data management. It stores two kinds of information: information relating to the types of data supported by the different viewpoint databases and information on the global constraints for solving integration conflicts during the exploitation. The metadatabase is used by the *mediator* in dealing with the users requests. The mediator is a processor that supports various functionalities: query propagation, query interpretation, result propagation, result interpretation and integrity checking. However, a constraint manager is integrated into the mediator and has the following roles:

1. It receives a notification message about an update that could possibly violate the object's integrity in a viewpoint database,
2. It constructs an execution plan of requests induced after integrity constraint checking. This plan will be executed within the transaction at the federated level, according to a two- phase- commit protocol,
3. The result of the constraint checking is then sent by the mediator to the appropriate database.

- The external level permits the exploitation of the federated system services. External schemas can maintain all the specificities of the multiple descriptions of data. The user can express his requests in terms of viewpoints on data.

8. 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] 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.

[2] S. ABITEBOUL, R. HULL, V. VIANU, *Foundations of Databases*, Addison-Wesley, Reading, Mass. (1995).

[3] F. BANCILHON, C. DELOBEL, P. KANELLAKIS, *Building an Object-Oriented Database System*. The story of O2, Morgan Kaufman, (1992), San Mateo, California.

[4] 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.

[5] F. BENCHIKHA, M. BOUFAIDA, Un modèle conceptuel pour une représentation multiple et évolutive des connaissances. *Proceedings of the 4th African Conference on Research in Computer Science*, (1998), pp. 793–804. Dakkar, Sénégal.

[6] F. BENCHIKHA, M. BOUFAIDA, L. SEINTURIER, The integration of the Viewpoint Mechanism in Federated Databases. *Proceedings of SAC'01*, ACM, (2001), pp. 280–284. Las Vegas. Nevada, United States.

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

[8] D. G. BOBROW, T. WINOGRAD, An Overview of KRL, a Knowledge Representation Language. *Cognitive Science*, Vol. 1, (1977).

[9] Y. BREITBART, A. SILBERSCHATZ, Multidatabase update issues. *Proceedings of SIGMOD Int'l Conference on Management of Data*, (1988), pp. 135–142. Chicago, Illinois, United States.

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

[11] N. CARN, *Représentation Orientée Objet de Système Opérationnel avec application au domaine spacial*. INP thesis, (1992), Toulouse, France.

[12] 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.

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

[14] L. DEBRAUWER, *Des vues aux contextes pour la structuration fonctionnelle de bases de données à objets en CROME*. Doctorat thesis, University of sciences and technologies, Lile, France, (1998).

[15] L. DEKKER, FROME: *Représentation Multiple et Classification d'Objets avec Points de Vues*. Doctoral thesis, University of sciences and technologies, Lile, France, (1994).

[16] 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.

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

[18] A. HALEVY, Logic-based techniques in data integration. *Proceedings of the Logic Based Artificial Intelligence*, (2000).

[19] D. HEIMBIGNER, D. MCLEOD, A Federated Architecture for Information Systems. *ACM Transactions on Office Information Systems* 3(3), (1985), pp. 253–278.

[20] A. KRIOUILE, *VBOOM, une méthode orientée objet d'analyse et de conception par points de vue*. Doctoral thesis, University of Mohamed V, Rabat, Maroc, (1995).

[21] J. L. LEMOIGNE, *La modélisation des systèmes complexes*. Dunod Edition, (1990).

[22] O. MARINO, *Raisonnement classificatoire dans une représentation à objets multi-points de vue*. Doctoral thesis, University of Joseph-Fourier, Grenoble, France, (1993).

[23] T. MENZIES, S. EASTERBROOK, B. NUSEIBEH, S. WAUGH, An Empirical investigation of multiple viewpoint reasoning in requirements engineering. *Proceedings of the 4th Int'l Symposium on Requirements Engineering* (RE'99), (1999), Limerick, Ireland.

[24] H. NAJA, *CEDRE: un modèle pour une représentation multi-points de vue dans les bases d'objets*. Doctoral thesis, University of Henri Poincaré, Nancy 1, (1997).

[25] G. T. NGUYEN, D. RIEU, Database Issues in Object-Oriented Design. *Proceedings of the 4th International Conference TOOLS*, (1991), pp. 73–86, Paris, France.

[26] M. PAPAZOGLOU, B. KRAMER, A Database Model for Object Dynamics. *The VLDB Journal 6*, (1997), pp. 73–96.

[27] B. PERNICI, Objects with roles. *Proceedings of Office Information Systems*, (1990), pp. 205–215, Cambridge, Massachussets.

[28] A. PONS, Formalisation logique d'un mécanisme d'héritage crédule avec points de vue. *Proceedings of Actes des Journées RPO, Edition EC2*, (1992), pp. 73–85, La Grande Motte.

[29] C. RATHKE, D. F. REDMILES, *Multiple Representation Perspectives for Supporting Explanation in Context*. Technical Report CU-CS-645-93, University of Colorado, Department of Computer Science, Boulder, Colorado. (1993)

[30] J. RICHARDSON, P. SCHWARTZ, Aspects: Extending Objects to support Multiple, Independent Roles. *Proceedings of the ACM SIGMOD Int'l Conference on Management of Data*, (1991), pp. 298–307. Denver, Colorado.

[31] D. RIEU, G. T. NGUYEN, A. CULET, J. ESCAMILLA, C. DJERABA, Instanciation Multiple et Classification d'Objets. *VII^{èmes} Journées Bases de Données Avancées*, Lyon, (1991).

[32] D. ROSS, K. E. SCHAMAN, Structured Analysis for Requirements Definition. *IEEE Transactions* 3(1), (1977), pp. 6–15.

[33] E. RUNDENSTEINER, Multiview: a Methodology for Supporting Multiple Views in Object-oriented Databases. *Proceedings of the 18th VLDB Conference*, (1992), pp. 187–198, Vancouver, British Columbia, Canada.

[34] E. SCIORE, Object Specialisation. *ACM Trans. Information Systems* 7(2), 1989, pp. 103–122.

[35] A. SHETH, J. LARSON, Federated Database Systems for Managing Distributed, Heterogeneous and Autonomous Databases. *ACM Computing Surveys* 22(3), 1990, pp. 183–236.

[36] J. J. SHILLING, P. F. SWEENEY, Three Steps to Views: Extending the Object-Oriented Paradigm. *Proceedings of OOPSLA'89*, 1989, pp. 353–361. New Orleans, Louisiana, United States.

[37] L. WANG, M. ROANTREE, Designing Roles For Object-Relational Databases. *Proceedings of the 5th International Workshop on Engineering Federated Systems* (EFIS'2003), (2003). Coventry, UK.

Received: April, 2005
Accepted: November, 2005

Contact addresses:
Fouzia Benchikha
LIRE Laboratory
Department of Computer Science
Mentouri University of Constantine
25000 Constantine, Algeria
e-mail: f_benchikha@yahoo.fr

Mahmoud Boufaida
LIRE Laboratory
Department of Computer Science
Mentouri University of Constantine
25000 Constantine, Algeria
e-mail: boufaida_mahmoud@yahoo.fr

FOUZIA BENCHIKHA is an Assistant Professor in the Computer Science Department of the University of Skikda in Algeria. Her current research activities are conducted at the LIRE Laboratory at the University of Constantine. Her research interests include advanced information systems, distributed databases and software engineering.

MAHMOUD BOUFAIDA is a Professor at the Department of Computer Science of the Mentouri University of Constantine in Algeria. He currently heads the research group "Information Systems and DataBases" of the LIRE Laboratory at the University of Constantine. His current research interests include cooperative information systems, databases, multi-agent systems and software engineering.