Viewpoints: a Framework for Object-Oriented Database Modelling and Distribution

Fouzia Benchikha, Mahmoud Boufaida, Lionel Seinturier

VIEWPOINTS: A FRAMEWORK FOR OBJECT ORIENTED DATABASE MODELLING AND DISTRIBUTION

Fouzia Benchikha[1]*, Mahmoud Boufaida[2]* and Lionel Seinturier[3]

*[1]LIRE Laboratory, Departement of Computer Science, Mentouri university of Constantine
25000 Constantine, Algeria
Email: f_benchikha@yahoo.fr
*[2]LIRE Laboratory, Departement of Computer Science, Mentouri university of Constantine
25000 Constantine, Algeria
Email: boufaida_mahmoud@yahoo.fr
[3]LIP6 Laboratory, Pierre et Marie Curie University, 75252 Paris, France
Email: lionel.seinturier@lip6.fr

ABSTRACT

The viewpoint concept has received widespread attention recently. Its integration into a data model improves the flexibility of the conventional object-oriented data model and allows one to improve the modelling power of objects. The viewpoint paradigm can be used as a means of providing multiple descriptions of an object and as a means of mastering the complexity of current database systems enabling them to be developed in a distributed manner. The contribution of this paper is twofold: to define an object data model integrating viewpoints in databases and to present a federated database system integrating multiple sources following a local-as-extended-view approach.

Keywords: Object data model, Viewpoint mechanism, Referential schema, Viewpoint schema, Federated databases.

1 INTRODUCTION

The rising popularity of distributed databases and their expansion towards new applications supporting the increased complexity and irregularity of the real-word entities requires the development of advanced database models. 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 such as O2 and ODMG ones.

However, within earlier object-oriented models, 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. Unfortunately, conserving this single class paradigm seems unsuitable for coping with the complex representation of real-world entities. Indeed, in the new distributed and co-operative applications, it's difficult to work out a single representation of objects, which would be appropriate to every participants of the same project. Thus, it would be desirable for objects to have multiple descriptions, which take account of several points of view, while each one keeps its specificity and allows the sharing and the exchange of information. 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 (Charrel, Galaretta, Hanachi & Rothenburger, 1993), knowledge representation (Dekker, 1994), database systems (Debrauwer, 1998) (Naja, 1997), web applications (Gergatsoulis, Stavrakas, Karteris, Mouzaki & Sterpis, 2001), etc. In DataBases (DBs), we notice few works on the integration of the viewpoint concept to data models. Most of these works consider the view and the role mechanisms. Views (Abiteboul & Bonner, 1991) (Bertino, 1992) (Rundensteiner, 1992) are external schemas that provide the user with part of the global schema, a kind of *viewpoint* on the description of its entities. Roles (Albano, Bergamini, Ghelli & Orsini, 1993) (Coulondre & Libourel, 2002) (Gottlob, Schrefl & Rock, 1996) (Wang & Roantree, 2003) 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 of 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 viewpoint descriptions are supported by database schemas that together provide the global schema of the same real world data called the "multi-viewpoints database schema". Objects can be described according to one or more descriptions, as kind of role within a multiple data representation. Achieving such an approach requires a distributed environment that permits the integration and the collaboration of a collection of databases.

The significance of this paper is the description of the impact of the viewpoint mechanism on the modelling of large-scale databases and its integration into a distributed database environment. In particular, federated or loosely-coupled database systems (Heimbigner & McLeod, 1985) (Sheth & Larson, 1990) are more easily adapted to our approach because their principal objective is to ensure the autonomy of the component databases and to cope with their management and their handling. These goals suit those of the viewpoint approach. However, recent works (Levy, 2000) (Manolescu, Florescu & Kossmann, 2001) classify data integration systems according to the way the schema of the local data sources is related to the unified global schema. Two approaches are presented in (Levy, 2000): 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. We are particularly interested in the LAV architecture that will be adapted to our architecture.

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, we propose the MVDB (Multi-Viewpoint DataBase) model, which is an extension of the conventional object data model with the viewpoint concept. The proposed model allows the development of a database 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. In Section 5, we present the general architecture of a federated database system, called MVDB system that uses an adapted LAV approach to integrate viewpoint sources. Section 6 concludes our work.

2   THE VIEWPOINT APPROACH: AN OVERVIEW

In recent years, 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 (Charrel, et al, 1993), knowledge representation (Dekker, 1994) (Rieu, et al, 1991), database systems (Debrauwer, 1998) (Nguyen & Rieu, 1991), web applications (Gergatsoulis, et al, 2001), requirements engineering (Menzies, et al, 1999) and complex systems modelling (Carn, 1992). Several terms have been assigned to this concept such as roles (Pernici, 1990), aspects (Richardson & Schwartz, 1991), perspectives (Sciore, 1989), dimensions (Gergatsoulis, et al, 2001), and viewpoints (Benchikha & Boufaida, 1998) (Charrel, et al, 1993).

Currently, interest in the viewpoint mechanism has grown continuously. It 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. In this case it is defined as the fact of conferring several partial descriptions to the same UoD each of which describes it in a given viewpoint. The various partial descriptions are complementary and together provide a complete and a coherent global description of the entities. 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 (Stavrakas, Gergatsoulis & Mitakos, 2000).

➢   **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 (1989) 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 (Rathke & Redmiles, 1993) that describes programs according to different aspects (source program, graphical representation and so on).

➢ **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 Lemoigne (1990) can not be handled with the same techniques as used for simple systems. However, the modelling of a complex system can not be a centralized task based on a single formalism. Different works suggest a distributed development approach based on viewpoints (Debrauwer, 1998; Kriouile, 1995). 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) (Kriouile, 1995) 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 (Ross & Schaman, 1977).

➢ **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 has been reconsidered via the KRL perspectives (Bobrow & Winograd, 1977), the CROME contexts (Debrauwer, 1998) and the TROPES viewpoints (Marino, 1993).

➢ **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 (Benchikha & Boufaida, 1998) (Dekker, 1994) (Rieu et al, 1991), in inherited value retrieval (Pons, 1992), in the modelling of independent concepts and in dealing with the multiple inherited conflicts (Dekker, 1994). 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 (Carn, 1992).

In the context of our work, the viewpoint paradigm is used essentially as a means of providing multiple descriptions of an object (see Section 4.2), as a mechanism for dealing with integrity constraint problems (see Section 4.3) and as an approach for the distributed development of a database schema (see Section 5.2). In the next section, we present an overview of viewpoints in the area of databases.

3   THE VIEWPOINT CONCEPT IN DATABASES

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 we compare them with the viewpoint concept. Figure 1 presents in a conceptual manner roles, views, and viewpoints in a database schema.
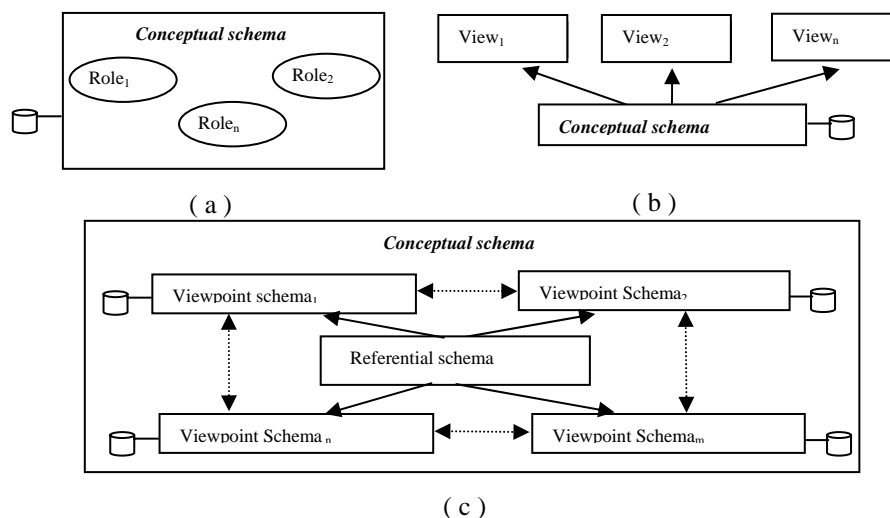

Figure 1. Roles, views and viewpoints approaches

3.1   Roles

Objects with roles have increasingly been studied by several authors (Albano, and al, 1993) (Coulondre & Libourel, 2002) (Gottlob, and al, 1996) (Wang & Roantree, 2003). A comparison of these proposals can be found in (Papazoglou & Kramer, 1997).

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 object's 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 dynamic object classification is a tedious task. The role hierarchy-based approach, however, has been adopted in many extended object-oriented database systems (Gottlob, et al, 1996) (Wang & Roantree, 2003). 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. Roles commonly refer to the evolution of an object's behaviour while viewpoints, which we consider, provide the multiple descriptions of the object (Figure 1, (a), (c)). Objects in our study can be viewed according to several interdependent and complementary descriptions. Each description of the object's structure and behaviour depends on the point of view concerned. However, roles can be a complementary evolutionary aspect to viewpoints, in the sense that an object can dynamically change its behaviour over time within a viewpoint's representation.

3.2   Views

Various view models have been proposed such as the multi-view model of Rundensteiner (1992) and the view model of Abiteboul and Bonner (1991) and of Bertino (1992). 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 into the global schema (class hierarchy) and whether an instance of a view owns an identity. Abiteboul 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 the information from where it is actually stored. A view can be treated as a database but it does not preserve an object's identity. Rundensteiner and Bertino 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 of the 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. In our study we argue that the view and the viewpoint concepts concern the exploitation step and the design step respectively (Benchikha, Boufaida & Seinturier, 2001). Thus, while views provide users with a schema that is structured appropriately for their applications, viewpoints must be directly related to an object's description and deal with multiple instantiations of the same object (Figure 1, (b), (c)).

In conclusion, we state that roles, views and viewpoints are common since they all allow an object to be extended. However, while roles permit an object to dynamically change its type, and views allow an object to be seen as if it has a different structure, viewpoints deal with both the evolution and the multiple representation aspects within a unique paradigm. It also permits the novel dimension of distributed data descriptions. To the

best of our knowledge, no current object-oriented database system supports this aspect. Our MVDB model allows the construction of distributed object databases based on viewpoints.

4   THE MVDB MODEL

In this section we present the basic notions and concepts of our model.

4.1   Basic notions

The MVDB model is based on the object-oriented paradigm within the DBs framework. We adopted this object 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, 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 object modelling features in MVDB, such as the multi-instantiation mechanism that permits an object to have more than one instance.

The model relies on the following ideas:
- A database schema is a multiple description of the same UoD. It is viewed as a set of ViewPoint schemas (VP schemas), as shown in Figure 1 (c). 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 referential schema that holds basic data on the real word entities shared by all the VP schemas. Any VP schema can describe the whole or a part of the referential data according to a given point of view;

- 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.

We wish to point out that object identity is a central notion in our approach. It is the same object described in many ways according to its membership of the various VP schemas. VP databases are complementary and provide a global distributed database called a multi-viewpoint database. A coherent exploitation of this global database is then recommended. This modelling approach confers a decentralized vision of a database schema, facilitates the parallel work of several designers and leads to more powerful data structuring.

Generally these features are particularly needed in the 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 exists 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 2). 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 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.
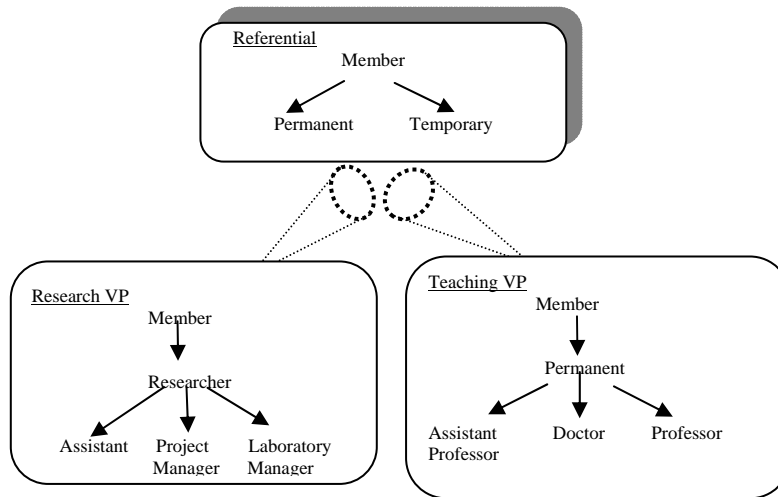
Figure 2. A multi-viewpoint modelling example

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. The member "Benali", for example, is a *permanent* member, *Doctor* and *Project Manager*.

4.2   The formal model

An Object-oriented DataBase (ODB) according to database technology, contains two fundamental types of information: data, represented by the object's state (the database *extent*), and metadata, implemented in the database schema. An ODB schema is organized as a class hierarchy according to the sub-class relationships. These relationships provide the conditions to establish if two classes are generalized/specialized. MVDB proposes an extension of the conventional object model to support the viewpoint paradigm.

Let be MVDB (S_r, VP, C, O), the specification of the data model signature, where:
- S_r is a referential schema name.
- VP is a set of viewpoint schema names,
- C is a set of class names,
- O a set of objects.

In the following, we define formally the basic concepts of the model.

4.2.1 Schemas

In MVDB, a multi-viewpoints database or a database schema is described by a referential schema and several viewpoint ones. The referential schema is the basic schema that describes all the entities independent of any viewpoint. A viewpoint schema is the customization of the whole or a part of the referential schema. By default, we consider that the database schema is defined via its referential schema definition.

Definition 1. Referential schema
The referential schema is defined as any common object schema $S_r = (C_r, \propto_r)$ where:
C_r ($C_r \in C$) is the finite set of class names in the schema and $\propto_r$ is the sub-classing relationship, which is a strict partial ordering among C_r.
S_r is well-formed if:
$\forall (c, c') \in Cr \times C_r, c \propto c' \Rightarrow (Ext(c) \subseteq Ext(c'))$
where Ext is the extension of a class.

Example 1. The laboratory-schema is a database schema that describes the laboratory's scientific staff presented in Figure 2. It is defined by its referential schema that models information about the laboratory's members. Each member is an object stored in 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 Permanent from Member
               Public  type  tuple (
                  Salary  :  real  )
End;
Class Temporary from Member
               Public  type  tuple (
                  Nb-hours: integer)
End;
End.
Name  members  =  set (member);
Export-schema laboratory-schema;
Export-base  laboratory-base;
```
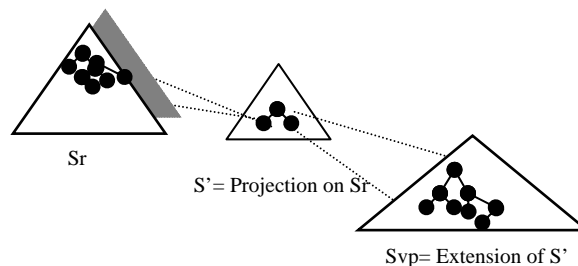
Definition 2. Viewpoint schema

Let $S = (C, \propto)$ and $S' = (C' , \propto')$ two schemas.

S' is a projection of S (conversely, S is an extension of S' on C), denoted $S' \nabla S$ (conversely $S \Delta S'$) if: $(C' \subseteq C)$ and $(\sigma' \mid C' = \sigma)$

A viewpoint schema Svp is an extension of a projection S' on the referential schema Sr (as depicted in Figure 3).

Sr

S'= Projection on Sr

Svp= Extension of S'

Figure 3. Viewpoint schema = Projection + Extension of the referential schema

A viewpoint schema is obtained from two steps: firstly a projection operation (∇) is carried out on the referential schema to select a 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 descriptions 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 Bertino (1992) and Rundensteiner (1992). This technique consists of distributing the projected referential schema in the viewpoint schema.

Example 2. The research-vp schema refines the member's description according to the research point of view. All the members are concerned with this description here. The schema definition is:

```
┌─────────────────────────────────────────────────────────────────┐
│  Viewpoint schema Definition                                      │
│  Schema Research-vp  from laboratory-schema;                      │
│  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 Project Manager ….                                        │
│  Class Laboratory Manager ….                                     │
│  End.                                                             │
└─────────────────────────────────────────────────────────────────┘
```

Remark: The teaching VP schema, presented in Figure 2, concerns only the description of part of the laboratory schema i.e. the permanent members. Temporary members do not carry out teaching activity.

4.2.2 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.

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-representational modelling of real word objects. These characteristics are crucial in advanced object-oriented technology. Indeed, a multiple instantiation mechanism is proposed to overcome this rigidity. The multiple instantiation 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 3. 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 a viewpoint OID (Object IDentification) for the object,
An object in a viewpoint database becomes a pair (r_l, v_l) where v_l is its local state value.

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 MVDB System in Figure 6).

Example 3. "Benali" is a member of the laboratory staff who carries out a research activity. The local referent of this object at the research-vp database is: (research-vp, oidvpr1) with oidvpr1, the local identifier of the object. The object instance at this level can be as follows: (oidvpr1, {Benali, Mohamed, 40, 12, 'Constantine-University'}).

Definition 4. Global object referent
A global object referent, denoted by r_g, represents the identification of the object in the multi-viewpoint database so 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: $L(r_l) = U_{vp \in VP} (r_l)$.
An object becomes a pair (r_g , v_g) where v_g is its global state value.

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.

Example 4. Let consider "Bencherif" a permanent member at the laboratory. He carries out both teaching and research activities. The global referent of this object is: (oid2, ((research-vp, oidvpr2),(teaching-vp, oidvpt1))). The object instance at the global level can be: ((oid2, {Bencherif, Ali, 55, 10000}),((research-vp, oidvpr2),(teaching-vp, oidvpt1)).

4.2.2 Bases

The objects constitute the associated bases of the referential and the viewpoint schemas. These bases constitute the multi-viewpoint base (extent) of the multi-viewpoint schema. They give a complete description of objects according to multiple viewpoints. They are defined in the following.

Definition 5. Referential base
Let R_r be the finite set of the objects referent of the referential schema S_r and VP the finite set of the viewpoints schema names.
A referential base, denoted B_r, is a schema state specified as a tuple $(\pi_r, O_r, \sqrt{}_r)$ where:
   $- \pi_r : C_r \rightarrow R_r$ is the function that associates to each class $c \in C_r$ the objects referent,
   $- O_r$ is the set of the objects in $B_r : O_r = U_{c \in Cr} \{(\pi_r (c), U_{vp \in VP} (rl))\}$,
   $- \sqrt{}_r$ is the function that associates a global value to each object of B_r, such us : $\forall c \in C_r, \forall o \in \pi_r (c)$,
       $\sqrt{}_r(o) \in \sigma_r(c)$.
where B_r describes the UoD entities.

Example 5. The associated base of the referential schema, presented in Example 1 is populated with objects. Each object has an instance in the 'laboratory-base' (referential base) and may or may not be instantiated in the research and/or the teaching VP. Instances are declared at the root of the schema as presented in the following:

```
Referential base:
    laboratory-base (
        ((oid1, {Benali, Mohamed, 40}) , ( (research-vp , oidvpr1))),
        ((oid2, {Bencherif, Ali, 55, 10000}),((research-vp, oidvpr2),(teaching-vp, oidvpt1))),
        ((oid3, {Benyoucef, Ahmed, 32}), nil)
        )
```

Definition 6. Viewpoint base
Let O_{vp} be the finite set of the objects identifier of a viewpoint schema S_{vp}.
A viewpoint base, denoted B_{vp}, is a schema state specified commonly as a tuple $(\pi_{vp}, O_{vp}, \sqrt{}_{vp})$ where:
   $-\pi_{vp} : C_{vp} \rightarrow O_{vp}$ is the function that associates to each class $c \in C_{vp}$ the objects identifier,
   $-O_{vp}$ is the set of the objects in $B_{vp} : O_{vp} = U_{c \in Cvp} \{\pi_{vp} (c)\}$,
   $-\sqrt{}_{vp}$ is the function that associates a value to each object of B_{vp}, such as : $\forall c \in C_{vp}, \forall o \in \pi_{vp} (c)$,
       $\sqrt{}_{vp}(o) \in \sigma_{vp}(c)$.

Example 6. The associated base of the research-vp schema, presented in Example 2 is:

Viewpoint-base:
> researchers-base (
> (oidvpr1, ⟨ Benali, Mohamed, 40 , 12, 'Constantine-University' ⟩),
> (oidvpr2, {Bencherif, Ali, 55, 10, 'Constantine-University'}
>)

In the following we focus a bit more on an essential and complementary functionality of every data model: consistency.

4.3   MVDB consistency

Unlike the traditional approach (mono viewpoint) where the integrity constraints are defined in the global schema, in the viewpoint approach we distinguish two types of constraints.

- Local constraints: they help to ensure the local coherence of the entities in a viewpoint database independently of the other bases.

- Global constraints: they help to ensure coherence of the global description of entities according to several viewpoints.

If the local constraints are well understood, it is difficult to take into account the global constraints. Classically, the principal conflicts found in federated databases are the names, the semantic and the structural conflicts. In our work, these can be solved by the viewpoint mechanism.

- Naming conflicts: traditionally, the solving of this type of conflict is done by assertions specifying the synonyms and the homonyms. In our context, the existence of the referential solves any conflict arising from a problem of synonymy. Thus, all the common properties are described by the referential schema. On the other hand, a conflict arising from homonyms is solved by the viewpoint mechanism itself. As a matter of fact, two distinct homonym constructions can be differentiated by prefixing them, for example, by the name of the VP schema.

- Semantic and structural conflicts: they are few or lacking, in a database schema designed according to various viewpoints. Nevertheless, each VP schema describes an aspect of the data in a semantically different way to the other descriptions. In addition, the referential permits a representation, and in the same way an unified structure of the real world entities that will have different descriptions according to different viewpoints.

However, within the framework of MVDB, we distinguish other types of conflicts. These ones guarantee the compatibility of and coordination between the different object descriptions in the system. Let us consider the following cases.

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

      Example 7. Any laboratory manager does not have the right to acquire a teaching activity at the laboratory, and therefore can't have a description according to this viewpoint.

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

      Example 8. Any teacher whose research time exceeds twenty hours a week must reduce his official teaching time by 40 percent.

3.   Referential integrity between VP 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 9. Every 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 following conditions are satisfied:

1. The referential base is locally coherent with respect to its schema, i.e. the object set checks the local constraints of the referential schema.
2. Each viewpoint base is coherent with the corresponding viewpoint schema, i.e. each viewpoint objects check the viewpoint schema constraints.
3. The gathering of the various bases is coherent, i.e. all the global constraints are satisfied.

5   GLOBAL ARCHITECTURE

We have noticed above that the viewpoint approach to databases requires a distributed environment. Distributed systems (Breitbart & Silberschatz, 1988) (Bukhres & Elmagarmid, 1996) (Sheth & Larson, 1990) 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.

Regarding the issue of autonomy, Sheth and Larson (1990) 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 the next paragraph, we explain the use of the federation as the appropriate architecture for our approach.

5.1   Federation as an appropriate architecture for the viewpoint approach

In tightly-coupled database systems all the different database schemas are integrated into only one global schema. This phase of integration should remove all the inconsistencies, the errors and the redundancies resulting from these view schemas. The integration of the components makes them lose their autonomy. Indeed, there is only one management level where all the operations are carried out in a uniform way. Therefore no distinction is made between the local and the global use of data.

Let us notice that the objective of a global integration schema is not to take into account the users descriptions. The conceptual schema resulting from the integration does not differ in its form from the one resulting from a direct design. Relative specificities according to the users are lost at the end of the design's integration. Thus, this approach does not meet the viewpoints structuring needs. We will see that the federated approach, on the other hand, preserves this multi-viewpoint description of data.

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, their management and independent handling.

The autonomy component is therefore the principal characteristic of a federated system. All the transaction management and integrity constraint checking protocols must preserve it. This perspective follows directly the principal motivations of the viewpoint approach presented above. Two strategies used to integrate independent databases in a federated system in a unified logical global schema are presented in Levy (2000) according to an ascending or a descending process (see Figure 4).
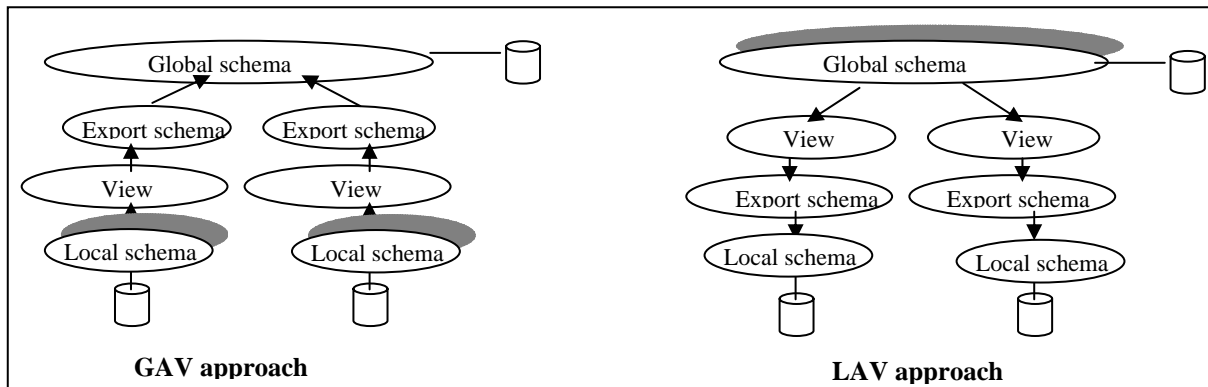
Figure 4. Data integration approaches

The ascending approach called Global-As-View (GAV) defines the global schema as a view over the local schemas. This suits the federated databases reference architecture presented by Sheth and Larson (1990). Thus, exported schemas are considered as views that are integrated and carried out at the federated level over a federated schema. In such an approach the schema evolution of local databases is a difficult problem. The opposite strategy known as Local-As-View (LAV) 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 VP description of data according to a viewpoint. A Local-As-Extended-View (LAEV) process is then used in our system (see Figure 5).
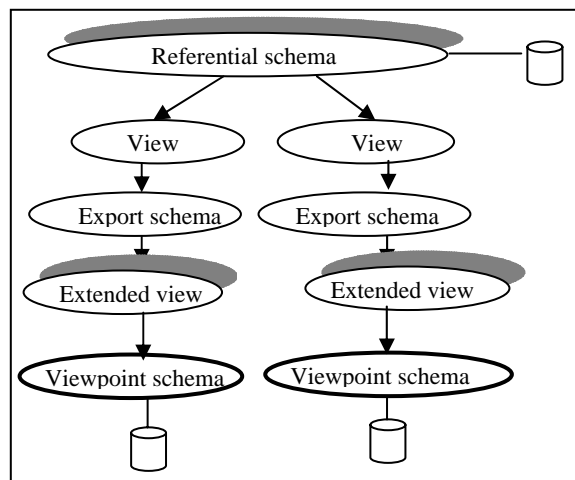
**Figure 5.** LAEV data integration approach

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

5.2   The basic architecture of the MVDB System

According to the viewpoint approach, presented in Section 4, the global schema is designed in a decentralized way according to several viewpoints. However, the database administrator initially defines a referential schema upon which the viewpoint schemas are created. At the viewpoint level, the local administrators define the viewpoint schemas. Each viewpoint schema gives a complete description of objects according to a given point of view. An integrated database is thus constructed as well as a metadatabase. The latter results from the interoperation of all the administrators and contains assertion rules dealing with global interdependency constraints to ensure the distributed database's consistency.

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.

The proposed architecture for the MVDB system is based on 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 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 6).
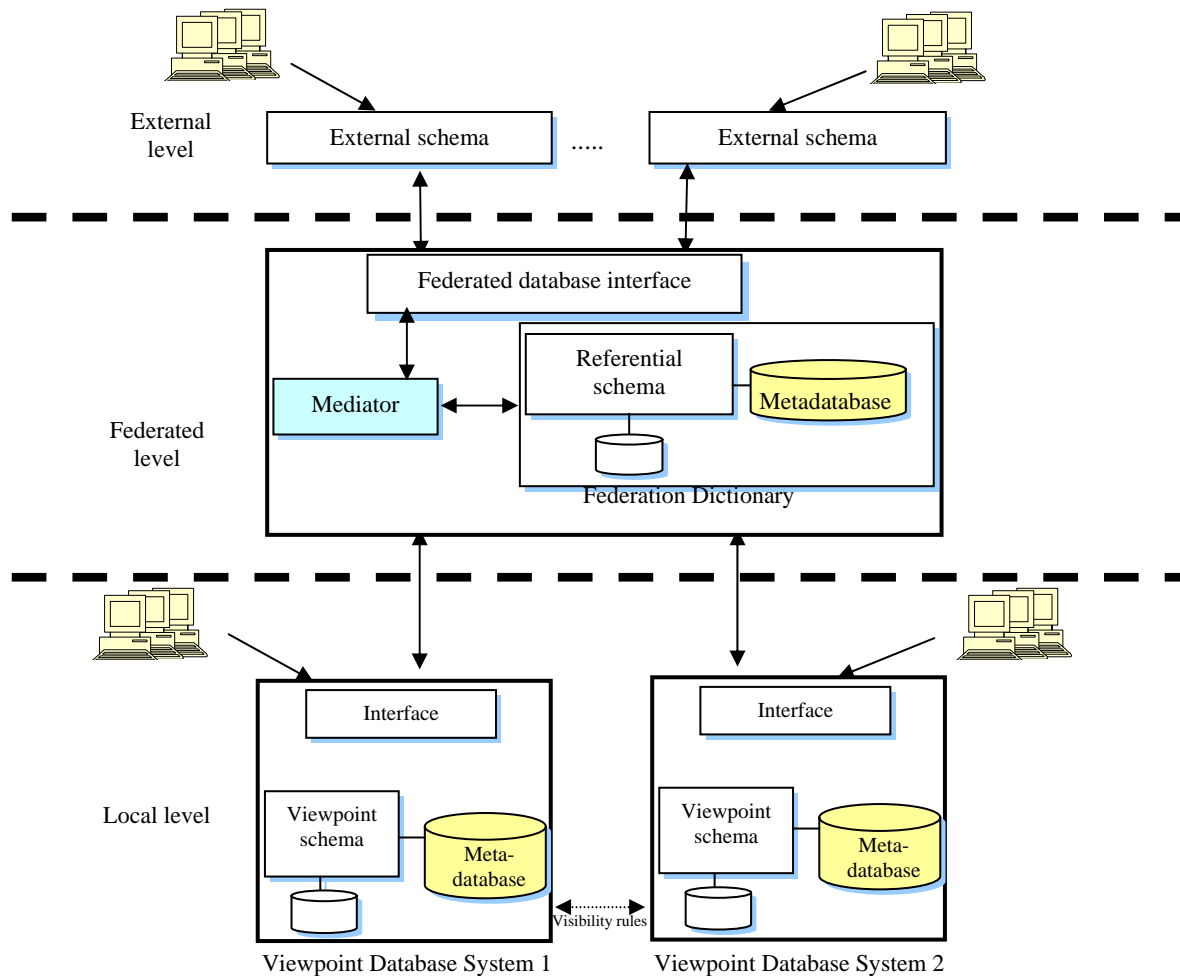
Figure 6. Global architecture of MVDB

➢ **The local level:** this level carries the partially 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:** the federated level is the kernel of our federated database system. It is essentially made 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 VP 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 activity that could possibly violate the object's integrity in a viewpoint database,

2. It constructs an execution plan of requests 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:** this 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.

6   CONCLUSION

In this paper, we have investigated the integration of the viewpoint mechanism into the database field. This mechanism can make an undeniable contribution to 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.

The proposed structure of the object model, MVDB, by integrating the viewpoint paradigm resolves the evolution, multiple description and distribution of objects. Objects are therefore characterized by:

• Distribution: a global object's representation is distributed throughout different databases. Each database represents a particular description of objects according to a particular viewpoint.

• Multiplicity: objects can be described simultaneously in multiple ways according to their membership in the various viewpoint schemas. These can be independent or not.

• Evolution: objects can acquire or lose viewpoint descriptions during their lifetime in the federated system. Global constraints integrity must thus be handled.

• Identity: object identity is a central notion in the MVDB model. It is the same object with a unique global identity that is described in multiple databases. A local identification is assigned to every object's viewpoint description to allow components autonomy.

• State: an object state is identified by the global state at the federated level and by the various viewpoint states at the local level.

Indeed, it would be interesting to generalize this approach, starting from an existing set of DBs. Then the aim would be to find the common referential, the viewpoint databases and their interdependencies.

It would also be interesting to study the behavioural aspects of objects that would be useful for constraint management. However, an object-oriented database management system may considerably increase its semantic power if the object's communication behaviour is incorporated into its object model. Thus, there would be no centralized place (constraint manager) where constraints can be stated, reasoned about, and maintained; every object would be responsible for maintaining the corresponding constraints.

Future work would concern further exploration of the data definition and the manipulation language's description. The later is an extension of the OQL language for dealing with the multi-viewpoint aspect of objects.

7   REFERENCES

Abiteboul, S. & Bonner, A. (1991) Objects and views. *Proc. The International Conference on Management of Data. ACM SIGMOD* (pp. 238-247). Denver, Colorado.

Albano, A., Bergamini, R., Ghelli, R. & Orsini, R. (1993) An Object Data Model with Roles. *Proc. 19th International Conference on Very Large Data Bases* (pp. 39-51). Dublin, Ireland

Benchikha, F. & Boufaida, M. (1998) Un modèle conceptuel pour une représentation multiple et évolutive des connaissances. *Proc. The 4th African Conference on Research in Computer Science* (pp. 793-804). Dakkar, Sénégal.

Benchikha, F., Boufaida, M. & Seinturier, L. (2001) The integration of the Viewpoint Mechanism in Federated Databases. Proc. *ACM, SAC'01* (pp. 280-284). Las Vegas, Nevada, United States.

Bertino, E. (1992) A View Mechanism for Object-Oriented Databases. *Proc. The 3rd International Conference on EDTB'92* (pp. 136-151). Vienna, Australia.

Bobrow, D.G. & Winograd, T. (1977) An Overview of KRL, a Knowledge Representation Language. *Cognitive Science*, 1.

Breitbart, Y. & Silberschatz, A. (1988) Multidatabase update issues. *Proc. SIGMOD International Conference on Management of Data* (pp. 135-142), Chicago, Illinois, United States.

Bukhres, O.A. & Elmagarmid, A.K. (1996) *Object-Oriented Multidatabase Systems*. Englewood Cliffs, NJ: Prentice-Hall.

Cardelli, L. & Wegner, P. (1985) On understanding types, data abstraction, and polymorphism. *ACM Computer Survey*. 17(4), 471-522.

Carn, N. (1992) Représentation Orientée Objet de Système Opérationnel avec application au domaine spacial. INP thesis, Toulouse, France.

Charrel, P. J., Galaretta, D., Hanachi, C., & Rothenburger, B. (1993) Multiple Viewpoints for the Development of Complex Software. *IEEE International Conference on Systems, Man and* Cybernetics (pp. 556-561). Le Touquet, France.

Coulondre, S. & Libourel, T. (2002) An Integrated Object-Role Oriented Database Model. *Data & Knowledge Engineering 42*(1), 113-141.

Debrauwer, L. (1998) Des vues aux contextes pour la structuration fonctionnelle de bases de données à objets en CROME. Doctoral thesis, University of sciences and technologies, Lile, France.

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

Gergatsoulis, M., Stavrakas, Y., Karteris, D., Mouzaki A., & Sterpis, D. (2001) A Web-Based System for Handling Multidimentional Information through MXML. *Lecture Notes In Computer Science (LNCS 2151)* (pp. 352-365). Amsterdam, The Netherlands: Springer-Verlag.

Gottlob, G., Schrefl, M. & Rock, B. (1996) Extending Object-Oriented Systems with Roles. ACM *Transactions on Information Systems 14(3),* 268-296.

Heimbigner, D., & McLeod, D. (1985) A Federated Architecture for Information Systems. *ACM Transactions on office Information Systems 3(3)*, 253-278.

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

Lemoigne, J.L. (1990) *La modélisation des systèmes complexes*. Paris: Dunod.

Levy, A.Y. (2000) Logic-Based Techniques in Data Integration. In Jack Minker (Ed.), *Logic Based Artificial Intelligence*. Norwell, MA: Kluwer Publishers.

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

Manolescu, I., Florescu D., & Kossmann, D. (2001) Answering XML Queries over Heterogeneous Data Sources. Proc. *The 27th VLDB Conference*. Roma, Italy.

Menzies, T., Easterbrook, S., Nuseibeh, B., & Waugh, S. (1999) An Empirical investigation of multiple viewpoint reasoning in requirements engineering. *Proc. The fourth International Symposium on Requirements Engineering (RE'99)*. Limerick, Ireland.

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

Nguyen, G. T., & Rieu, D. (1991) Database Issues in Object-Oriented Design. *Proc. The 4th International Conference TOOLS* (pp. 73-86). Paris, France.

Papazoglou, M. & Kramer, B (1997) A Database Model for Object Dynamics. *The VLDB Journal 6*(2) 73-96.

Pernici, B. (1990) Objects with roles. *Proc. Office Information Systems* (pp. 205-215). Cambridge, Massachussets.

Pons, A. (1992) Formalisation logique d'un mécanisme d'héritage crédule avec points de vue. *Proc. Actes des Journées RPO, Edition EC2* (pp. 73-85), La Grande Motte.

Rathke, C., & Redmiles, D.F. (1993) Multiple Representation Perspectives for Supporting Explanation in Context. *Technical Report CU-CS-645-93*. Boulder, Colorado: University of Colorado, Department of Computer Science.

Richardson, J., & Schwartz, P. (1991) Aspects: Extending Objects to support Multiple, Independent Roles. *Proc. ACM SIGMOD International Conference on Management of Data* (pp. 298-307). Denver, Colorado.

Rieu, D., Nguyen, G. T., Culet, A., Escamilla, J., & Djeraba, C. (1991) Instanciation Multiple et Classification d'Objets. VIIèmes Journées Bases de Données Avancées, Lyon, France.

Ross, D., & Schaman, K.E. (1977) Structured Analysis for Requirements Definition. *IEEE Transactions 3*(1)*,* 6-15.

Rundensteiner, E. (1992) Multiview: a Methodology for supporting Multiple Views in Object-oriented Databases. *Proc. The 18th VLDB Conference* (pp. 187-198). Vancouver, British Columbia, Canada.

Sciore, E. (1989) Object Specialisation. *ACM Trans. Information Systems 7*(2), 103-122.

Sheth, A. & Larson, J. (1990) Federated Database Systems for Managing Distributed Heterogeneous, and Autonomous databases. *ACM Computing Surveys 22*(3), 183-236.

Shilling, J.J. & Sweeney, P.F. (1989) Three Steps to Views: Extending the Object-Oriented Paradigm. *Proc. OOPSLA'89* (pp. 353-361). New Orleans, Louisiana, United States.

Stavrakas, Y. Gergatsoulis, M. & Mitakos, T. (2000) Representing context-dependent information using Multidimentional XML. Proc. *The 4th European Conference ECDL'2000, Lecture Notes in Computer Science (LNCS 1923)* (pp. 368-371). Lisbon, Portugal: Springer-Verlag.

Wang, L. & Roantree, M. (2003) Designing Roles For Object-Relational Databases. *Proc. The 5th International Workshop on Engineering Federated Systems (EFIS'2003)* (pp. 106-116). Coventry, UK.