# On Extending a Semantic Data Model with Some Aspects of Rules and Objects

Mahmoud Boufaïda
LIRE Laboratory
Computer Science Department
University of Constantine
Route d'Aïn-El-Bey
25000, Constantine, Algeria
Tel & Fax : 213.4.92.28.35
boufaida@ist.cerist.dz

Zizette Boufriche-Boufaïda
LIRE Laboratory
Computer Science Department
University of Constantine
Route d'Aïn-El-Bey
25000, Constantine, Algeria
Tel & Fax : 213.4.92.28.35

## Abstract

Expert systems, databases and object-oriented languages have known important developments, each one in its field. The database domain provides models allowing the specification of data schemes. The object-oriented systems offer mechanisms such as inheritance, encapsulation and message sending. They also provide abilities to model the real world objects and the operations that must be executed. The deductive aspects of expert systems facilitate the expression of events and integrity constraints that the database must satisfy at any time. The purpose of this paper is to present a conceptual modeling which integrates both features of deductive databases and mechanisms of object-oriented systems into a uniform framework. The defined model is based on an extended semantic data model and organizes the information systems through object-oriented and rule-based paradigms.

**Proceedings of the 5th KRDB Workshop**
**Seattle, WA, 31-May-1998**

(A. Borgida, V. Chaudhri, M. Staudt, eds.)

http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-10/

## 1 Introduction

In the last few years, databases, object-oriented systems and expert systems have grown quite isolated from each other. However, these domains can be complementary on several aspects. The database field provides models, allowing the specification of a data schema in which it becomes possible to express some constraints related to the consistency in the schema.

These constraints are automatically respected in the database management system. The object-oriented systems and the expert systems do not provide this ability of management of integrity constraints but offer other interesting mechanisms. The first ones offer abilities to model the real world objects and the operations that must be executed. They also provide means of data protection via encapsulation and message sending. Furthermore, the inheritance and the polymorphism play a big role in the structuring and the evolution of data models. The second ones, through associated reasoning mechanisms, allow one to product dynamically new knowledge. The deductive aspects facilitate the expression of events and integrity constraints that the database must satisfy at any time. The techniques used in the expert systems have been designed for exploiting a sole program than to allow the specification of several programs which can interact efficiently via a shared knowledge base.

Several works were concentrated on the integration of object-oriented systems/databases[Ber93], object-oriented systems/expert systems[Bou93], expert systems/databases[Sto92]. These three paradigms have different degrees of adaptability for different tasks, that makes their simultaneous use interesting in the development of complex systems in various domains such as banking, computer-integrated manufacturing, educational and medical fields. Nevertheless, various problems emerge from these different types of coupling, due to the lack of a global data model in which the aspects inherent to the compatibility have to be seriously studied.

The purpose of this paper is to present an approach allowing the conceptual modeling of advanced database applications. This modeling integrates important features of deductive and object-oriented databases into a uniform approach. To perform this integration, it is necessary to base the definition of the presented model on all these concepts. This model combines approaches to semantic data modeling[Hul87][Nav92] and organizes the information systems through object-oriented and rule-based paradigms.

Our approach extends the semantic model defined by Smith and Smith[Smi77] to support production rules and object-oriented concepts. Fundamentally, the idea in Smith and Smith model is to enrich the Entity/Relationship data model[Che76] to support all abstractions of data. The Smith and Smith model leads to a general data model with classes and object types. It is based on the notions of aggregation and generalization/specialization.

We expect from this extended model, the following features :

- inheritance of properties and methods, and support of complex object,
- object identity and operator overloading,
- expressive power of constraints,
- modification, gain of modularity and reuse of specification.

The remainder of the article is organized as follows. Related work in Section 2, surveys existing proposals to the integration of deductive and objects paradigms. Section 3 gives an overview of the new model and describe its basic principles. Section 4 presents the modeling of objects. In Section 5, we find a specification of rules. A description of the interaction between these two concepts (object and rule) is provided in Section 6. Finally, the last section contains some concluding aspects of this work.

## 2 Related work

Several research proposals have attempted to combine object-orientation, databases and logical languages [Que94][Liu96][Ber95][Dia91]. Logic was used to formalize notions underlying object-oriented data models. Some approaches concern deductive databases which allow the expression of traditional databases by means of recursion, and declarative querying with a logical form. Others consist of employing logical languages as query for object-oriented databases. Examples of languages used in such approaches are ROCK&ROLL, F-LOGIC, ROL and CORAL[Ber95]. In these approaches, there are no aspects of schema updating and evolution.

Object-oriented databases allow data modeling of traditional databases by means of object identity, complex objects, classes, class hierarchy and inheritance. Management systems of these databases provide advantages like handling complex objects, inheritance, relationships among sets of entities, associations between object behaviors, object definition at the schema level and so on. The notion of object is much more flexible than that of record or tuple, allowing one to one correspondence between real world objects and database objects. Examples of such systems are IRIS, EXODUS, ORION, O2[Ber93].

It exists a considerable amount of work in the areas of data modeling and object-oriented modeling. The work related to object-oriented modeling has mostly been reported in object-oriented analysis and design literature. Data modeling has been the subject of intense research since the beginning of database management system technology. The work on data modeling in the late 1970s and early 1980s was focused on extending relational and semantic data models[Nav92]. Although many of these models introduced features that are supported by object-oriented models, complete object-oriented data models were not introduced until the mid 1990s[Cze92]. Object-oriented models can be classified into two groups : object-oriented extensions of relational and semantic data models[Hul87], and database extensions of object-oriented programming languages[Ber93]. Pure object-oriented models do not include facilities such as rules, events and triggers in the modeling process although these aspects are used in several programming languages for modeling interactions[Sto92][Bou93][Dia91].

Unfortunately, deductive databases lack powerful data modeling mechanisms, whilst the object-oriented ones, lack logical semantics and declarative query language.

## 3 Description of the two-layered model

The Smith & Smith semantic data model is known as a static one, in the sense that it does not provide mechanisms for defining actions on real world objects. The fundamental abstractions that this model supports are classification, aggregation and generalization. Our approach extends this model by incorporating operations within classes. The goal of providing this new possibility in a semantic model lets it, not only, supporting the full object-oriented paradigm but ensures the cooperation among database objects and rules. The rules which are introduced by labels are invoked by database objects through encapsulated operations and are asked to ensure database consistency.

The most remarkable characteristic of our model is its proposal of constraint expression in a declarative language for keeping separated logic in rules and control in objects, whose the method activation determines the effect on the database. So, the conceptual modeling of an information system, involves a schema which is based on two interrelated levels (See Figure 1):

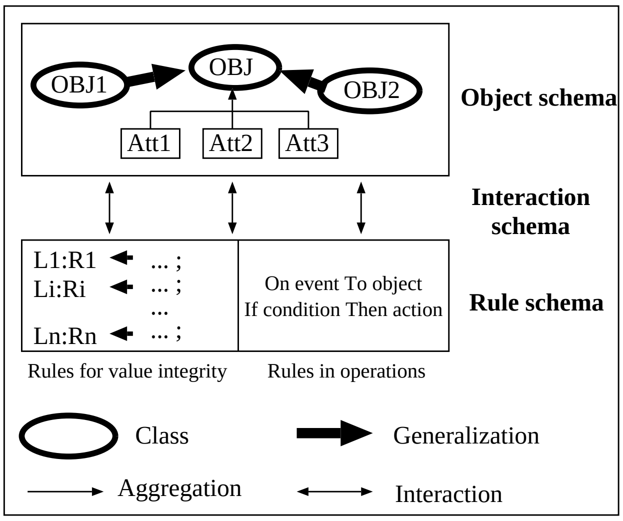

Figure 1: A global view of the defined model

- the first level (**Object model**) corresponds to objects with their operations. Each object can activate a rule. The object model deals with complex objects and object identity, taking into account an inheritance hierarchy among classes. It allows aggregation links to be modeled and incorporates also the notion of methods for deriving new information from object states. The object attributes can be reasoned about in the formalism because the model supports declarative specification through rules.

- the second level (**Rule model**) contains rules. The rule model represents the dynamic aspect of these objects.

These two layers of description of the real world concepts allow a distinction between the objects and the rules which manage the consistency of the information system. The articulation between these two levels (**Interaction model**) uses a message sending mechanism. It corresponds to the events emitted by operations which exist in the object model. These events fire a rule or a rule packet. The originality of the chosen representation is based on the priority given to the modularity and the reuse of component specification.

The semantic model extended with operations and rules leads to a better design because most modifications in the functionality of applications would not require reprogramming objects but would only require modifications of interactions and activities. Moreover, an object can be used independently of the rules because no knowledge of any other object is built into it.

Integrating a production rule facility into a database provides a uniform mechanism for a number of advanced database features, including integrity constraint enforcement, derived maintenance, triggers. In addition, a database management system with rule-processing capabilities provides a useful platform for large and efficient knowledge-base and expert systems.

## 4 Object model

In our model, an object is an abstraction of an entity of the domain world and is characterized by its state and a set of operations that triggers a rule or a rule packet in the schema rule. The state of an object is defined in terms of its attributes.

In order to present the object model, we have to explicit the basic notions of object and type and also to describe the different constructors which are defined. The most used concepts are shortly described in the following :

- an **object** is the most elementary components of the conceptual view of the modeled world. It is an instance of a class which is created dynamically during computation. Each instance (of an object class) is uniquely identified by an **identifier**,

- a **class** contains sets of objects of a given type that share certain **attributes**. A class defines a collection of instances of the same type,

- **methods** corresponding to operations are associated to a class. The methods use the rule labels to trigger them.

According to this model, an information system consists of classes with associated messages. Messages are sent to a set of rules. Three constructors are proposed :

- the **aggregation** describes a composite object which is composed from basic instances : an object can be composed from other objects,

- the **generalization** allows to manipulate types which are different structurally. The object description may also be embedded in a specialization hierarchy. A specialization describes a more detailed aspect of a conceptual entity,

- the **collection** contains sets of objects of some type.

Let us give an example of the previous concepts (See Figure 2):
PERSON is Class
{
name is string ; /* attribute type */
age is integer ; /* attribute type */
address is aggregate ; /* aggregation constructor composed of the attributes : number, street and town*/
children is collection of PERSON; /* collection constructor */
}
EMPLOYEE is PERSON /* generalization constructor */
{
department is string; /* attribute type */
salary is real; /* attribute type */
}

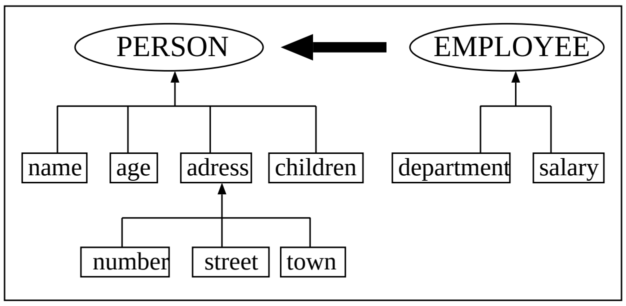

Figure 2: Example of classes and their attributes

## 5 Rule model

This model allows one to represent as well rules for value integrity as rules in operations. In any domain, there are certain rules that need to be enforced. When a rule is violated, corrective actions may need to be taken. The rule construct, therefore, should specify the invalid states of the domain and the corrective actions that should be taken when such states occur. A rule may span multiple objects.

We propose two primary rule structures which have different levels of capability. The first type (**integrity rules**) supports special-purpose rules for expressing properties of objects and maintaining value integrity. The second type (**query and updating rules**) supports rules whose triggering is based on events and fires only when some of them occurs such as the issuance of a particular database updating.

Typically, in artificial intelligence systems, an inference engine cycles through all the rules in the system, matching the conditions parts of the rules with data in working memory. Of all the rules that match (the condition set), one is selected using some conflict resolution policy, and this selected rule is fired. That is, its conclusion part is modified. The conclusion part is set to true. The cycle continues until no more rules match.

### 5.1 Integrity rules

These rules are defined in order to check the value integrity in the database. Such rules are attached to methods. A rule in this case, is composed of a list of conditions on the object attributes and returns a true or a false value indicating whether it is satisfied or not. In rule-based systems using objects, the conditions are usually written in a quadruplet form (Object Attribute Operator Value). In the current model, this form will be extended as shown in this section. The methods defined locally to the objects, constitute the rule action parts. They will not be executed if the related rules are not satisfied. The most important problem here is the specification of a binding between the condition and the action of a rule. Some systems specify no binding at all. Others use special names in the rule action to refer to data matching the rule condition. In our model, an explicit linkage which makes use of special labels is employed to logically link the condition and action parts.

The general form of this first kind of rules (based on first-order logic) in the rule model is the following :
**L :**
**Rj <—– O1 C(a1, a2, ..., an) ; O2 C(a1, a2, a3, ..., an) ; Oi C(a1, a2, ..., an) ;... Om C(a1, a2, ..., an);**

- L is the label of the rule,

- Rj is an intentional deduction of the rule. Its right part is a conjunction of conditions (separated by semicolon). Rj is set to true if all the conditions in the rule are true,

- Oi C(a1, a2, ..., aj, ..., an) is a set of conditions which refers to the attributes a1, a2, ..., aj, ..., an of the object Oi.

The intuitive meaning of a rule in the rule model is that : if a condition on attributes of object1 is true and condition on attributes of object2 is true and ...., and a condition on attributes of objectn is true then the rule or the rule packet returns true and the operation which has triggered them is accepted.

The model does not support strict encapsulation in the sense that it allows to directly access the attribute values through the rules. We only disallow the modification of object attributes from outside the object.

We give below an example of database objects and the associated rules :

```
EMPLOYEE is PERSON
{
department is string ;
salary is real ;
....................
ModifySalary(percent) : CheckSalary
{salary = salary * percent}
Display ()
{print (name, salary, ...) ;}
........
}
DEPARTMENT is Class
{
name is string ; /* attribute type *
.........
}
```

In this example, PERSON is a generalization of EMPLOYEE, where ModifySalary() is a method defined over objects of the EMPLOYEE class. The rule packet schema associated with EMPLOYEE is labeled with CheckSalary.

CheckSalary :

? Dept DEPARTMENT

? Emp EMPLOYEE

R1 <—– Dept name = " Production " ; Emp salary < 50.000 ; Emp department = Dept name

R2 <—– Dept name = " Management " ; Emp salary < 45.000 ; Emp department = Dept name

The rule 1 (respectively 2) means that the salary of an employee belonging to the Production (respectively Management) department can't exceed 50.000 (respectively 45.000). When a tentative of modifying an employee salary occurs, the rule packet labeled with CheckSalary is triggered for the instance Emp. The rule checks whether the employee salary and department are under the specified conditions. If at least one of the rules occurs, it returns true and the object Emp will see his salary raised.

## 5.2 Query and updating rules

These rules are related to query and updating operations which are defined in a class. They are production rules with events, conditions and actions. they are similar to those employed in artificial intelligence programming. The systems that support this kind of rules also allow different degrees of complexity, from simple selection patterns on one relation to a combination of selections and joins. The rule construct should specify the invalid states of the domain and the corrective actions that should be taken when such states occur. A rule may span multiple objects.

The defined rules in this category are similar than those used in the HiPAC (High Performance ACtive) project. The objective of this project is to design an active object-oriented database management system with introducing the concept of ECA (Event Condition Action) rules as formalism. HiPAC uses an object-oriented data model where rules are objects of a class defined in the model. Our approach offers means for separating rules and real world objects.

The database production rules for this second category are of the form :

**On event To object If condition Then action**

An event may be an updating, retrieval, insertion or deletion event. The action is one of the methods defined in the objects. Either the On event, If condition, or both can be present As an example, consider the following rules :

?Emp EMPLOYEE

On update To Emp
If Emp name = " John " Then Emp.ModifySalary(1.1)

On update To Emp
Emp.Modifysalary(0.9)

On retrieve To Emp
If Emp salary < 10.000 Then Emp.Display()

On delete To Emp
If Emp name = " Bob " Then Emp.Delete()

The first rule is triggered when the updated employee name is " John ". The rule's action modifies the employee salary after verifying the labeled rules. The second one allows all the employee salary decreases. The third rule logs all the salary whose salary is under 1 0.000 when the fourth one delete the employee whose name is " Bob".

## 6 Interaction level

In this section, we briefly review the key concepts of events which are a basis for determining the object interactions involved in the two layers. Events are emitted by database object operations (methods) in the object model. Then, we consider four kinds of events:

- **retrieval events** which allow the query of object information existing in the database.
- **insertion events** which capture the notion of inserting a new object into a class or giving an initial value to an object attribute.
- **deletion events** which correspond to deleting an object from a class or removing an attribute value of an object.
- **updating events** which capture the notion of changing an attribute value.

The interaction model corresponds to the events emitted by operations (methods) in the object model. These events fire a rule or a rule packet and they are a basis for determining the object interactions involved in the two layers. An event is a message transmitted by a method which must be defined for each relevant event of an object class.

The functions of these methods are :

- to transmit the relevant event to a rule or a rule packet,
- to receive the result of the transmitted event,
- to produce the target events directly induced by the received event, thus creating or destroying object instances, updating object attributes, checking integrity constraints.

The interaction specified by an event requires the triggering of the associated rule after the execution of the operation. If the rule execution returns true, the method is performed else it will be inhibited. An event allows the object model to establish a type of interaction among objects that occur frequently, but it is difficult to express without proper modeling constructs. An event is transmitted to the rule schema when an object reaches to be changed.

In conclusion, the interaction model is a mean for linking the object and rule levels. In a direction, it corresponds to a set of events, and in the opposite direction, it is the result given by the rule.

## 7 Conclusion

We have described the main components of an uniform approach to conceptual modeling of an information system. Basic aspects of this approach are objects, rules, and interaction. The global model consists of a two-layered model which is referred to as the Object-Rule-Interaction model. The object layer uses a class construct, aggregation, collection, generalization/specialization. The rule layer uses conditions and triggers for checking integrity and updating operations. The interaction between these two layers is defined by event constructs. Class construct is used to specify object type. Rules and operations establish dynamic interaction among instance objects based on the properties of the objects at type level. Events are invoked by changes on objects. They allow rules to be fired and constitute a link between the two layers. The results of such a representation are objects that are more reusable and simple for developing applications which can be easily modified and extended.

The proposed model has mainly the following advantages :

- the expressive power with classification, generalization, rules, triggers and event,
- the reuse of existing components in the object schema and the rule schema,
- the ease of use and modification of the different concepts which are employed,
- the possibility of expressing constraints in a declarative language,
- the ability of keeping separated logic in rules and control in objects whose the method activation determines the effect on the database.

Databases become more complex with aspects such multimedia. So, extending data models to include rules and objects could produce some interesting results with regard to intelligent retrieval and reasoning techniques.

Our work will be extended by considering the translation from the conceptual level to the operational one, with choosing a database language such as CO2[Lec89]. Then, we can define a global software architecture supporting all the defined concepts.

## References

[Sto92] M. Stonebraker. The Integration of Rules Systems and Database Systems. *IEEE Transactions on Knowledge and data Engineering*, 4(5):415–422, October 1992.

[Cze92] B. Czejdo and M.C Taylor. Integration of Information Systems Using an Object-Oriented Approach. *Computer Journal*, 35(5):501–513, 1992.

[Ber93] E. Bertino and L. Martino. Object-Oriented Database Systems : Concepts and Architecture. *Addison-Wesley*, 1993.

[Que94] C. Quer and A. Olive. Determining Object Interaction in Object-Oriented Deductive Conceptual Models. *Information Systems*, 19(3):211–227, 1994.

[Bou93] M. Bouziane and C. Hsu. A Rule Based Model for Data and Knowledge Interaction in Multiple Systems Environments. *International Journal On Artificial Tools*, 2(4):485–509, 1993.

[Liu96] M. Liu. ROL : A Deductive Object Based Language. *Information Systems*, 21(5):431–457, 1996.

[Hul87] R. Hull and R. King. ROL : Semantic Database Modeling : Survey, Applications and Research Issue. *ACM Computing Surveys*, 19(3), September 1987.

[Ber95] E. Bertino, G. Gurrini and D. Montest. Towards Deductive Object Databases. *Theory and Practice of Object Systems*, 1(1):19–39, 1995.

[Dia91] O. Diaz, N. Paton and P. Gray. Rule Management in Object-Oriented Databases : A Uniform Approach. *17th International Conference on Very Large Data Bases*, 317–326, September 1991.

[Nav92] S.B. Navathe. Evolution of Data Modeling for Databases *Communications of the ACM*, 35(9):112–123, 1992.

[Smi77] J.M. Smith and D.C.P. Smith. Database Abstractions : Aggregation and Generalization. *ACM Transactions on Database Systems*, 2(2):105–133, 1977.

[Lec89] C. Lecluse and P. Richard. The O2 Database Programming Language. *5th International Conference on Very Large Databases*, 411–422, 1989.

[Che76] P.P. Chen. The Entity-Relationship Data Model : Towards a Unified View of Data. *ACM-TODS*, 1(1), 1976.