An Active and Deductive Object-Oriented Data Model

Giovanna Guerrini
Dipartimento di Informatica e Scienze dell'Informazione
Università di Genova, Italy

Thesis Advisor:

Prof. Elisa Bertino
Dipartimento di Scienze dell'Informazione
Università di Milano, Italy

Thesis Proposal

Phd in Computer Science
Dipartimento di Informatica e Scienze dell'Informazione
Università di Genova, Italy

Summary

The aim of the thesis is to investigate the problems deriving from the introduction of active and deductive rules in object-oriented data models. This research topic falls at the confluence of three among the most promising research areas in the database field, namely object-oriented databases, active databases and deductive databases. These three research areas, though moving along different directions, try to overcome the inadequacy of traditional data models and database languages (in particular, of the relational ones) with respect to advanced applications.

Object-oriented database systems overcome the limitations of relational ones with respect to the emerging data-intensive applications due to their ability of directly representing in the model complex objects and storing in the database not only data but also the operations that can be performed on them. Deductive rules enhance a database system with inferential capabilities, providing the support for derived data. They allow to represent in a database not only data, but also *knowledge*, making it possible to infer additional information from the stored data, providing also the capability of expressing recursively defined data. Deductive rules can moreover be used for restructuring and repartitioning the format of data, so that applications can customize shared data, through the definition of views. An active database system is a database system in which some operations are automatically executed when specified events happen and particular conditions are met. Thus, active rules enhance traditional database systems with reactive capabilities, providing powerful and efficient mechanisms for database integrity constraints, authorization, statistic gathering, monitoring and alerting.

This thesis will investigate how deductive and active rules can be effectively integrated in an object-oriented data model. Our goals can be summarized as follows:

- definition and formalization of a reference object-oriented data model;
- investigation of rule-based query and method definition languages;
- introduction of triggers in the reference object-oriented data model;
- investigation of integrity constraints peculiar to object-oriented data models;
- definition of a view mechanism for object-oriented databases;
- investigation of architectural issues concerned with rule support in object-oriented databases.

Finally, we plan to conclude the thesis with a discussion on how object-oriented data models and rules should be revisited in order to support temporal capabilities. Time is indeed an important aspect of most real-world phenomena, but conventional database systems do not offer the possibility of dealing with temporal data.

Contents

1 Introduction

The area of data models has always been a very active research area that has a considerable impact on data management technologies. Indeed, the evolution of data models has marked the various generations of database management systems (DBMS). In particular, since the definition of the relational model, research has progressed to develop "new-generation" data models. A number of modeling requirements can be devised for those new models, that include the possibility of defining and manipulating complex objects, user-defined operations, type hierarchies, mechanisms for supporting views and integrity constraints, inferential and reasoning capabilities, reactive capabilities. A number of research and development directions in the data model area can be identified, that try to address the above requirements; some of those directions are based on the integration of the database technology with programming language and artificial intelligence technologies. To date, three main directions can be devised in the area of data models: object-oriented data models, integrating database technologies with object-oriented programming; deductive data models, integrating database technologies with logic programming; active data models, resulting from the introduction in the database field of the production rule paradigm developed in artificial intelligence. Object-oriented data models [BM93] are establishing themselves as the most promising among those directions, mainly because of industrial interest that has resulted in the development of a large number of object-oriented database systems and prototypes. Moreover, several significant application have been developed on object-oriented DBMS, from areas such as geographical information systems, telecommunications, multimedia systems, spatial databases.

Object-oriented data models, however, still lack inferential and reasoning capabilities, and reactive capabilities. Thus, there is a growing interest in enriching the object-oriented data model with active and/or deductive rules. The use of rule-based languages in database systems has a number of advantages: these languages are easy to use and to understand, indeed, they are high-level languages if compared to traditional programming and database languages. Moreover, they have proved their effectiveness in extending *relational-based* languages with inferential and reactive capabilities. Therefore, an active, deductive, object-oriented data model seems to meet all the requirements of next generation database systems.

Deductive rules [CGT90] are the usual rules of logic programming, exploited in the database field to define derived or intensional data, that is, data not stored in the database. Indeed, those rules allow to derive new information from the facts explicitly stored in the database and allows to express in a convenient way recursively defined data. A deductive rule has the form

$$template \leftarrow query$$

where *template* specifies the structure of the derived data, and *query* is the expression that computes the derived data. An example of deductive rules is the following, specifying, in a Datalog-like syntax, a recursive view that starting from an extensional relation `parent` computes an intensional relation `ancestor`:

$$\texttt{ancestor(X,Y)} \leftarrow \texttt{parent(X,Y)}$$
$$\texttt{ancestor(X,Y)} \leftarrow \texttt{parent(X,Z)}, \texttt{ancestor(Z,Y)}$$

Deductive rules can be added to object-oriented databases for different purposes. First of all, the deductive rule language (which usually is a subset of a logic programming language) can be used as method definition language and as query language, bringing advantages of declarativity and uniformity. Moreover, rules can be used to express the population of derived classes and to define derived attribute values. In addition, deductive rules can be used to define inter-class relationships (views).

Active databases [CW96] comes from the introduction in the database field of the notion of active rule (or *production rule* or *trigger*) from artificial intelligence. An active rule is a syntactical construct to define the reaction of the system. Active rules are usually specified by means of the *event-condition-action* (ECA) paradigm, that is, they have the form

ON event IF condition DO action

Events are monitored and their occurrence activates (triggers) the rule, a condition is a declarative formula that must be satisfied in order for the action to be executed; an action is a sequence of database operations (possibly causing state changes). An example of active rule for integrity constraint enforcement is the following, referring to a database containing a relation `employee` with attributes `manager, salary,#`:

```
ON new employee or updated employee.salary
IF employee (X), employee (Y), X.manager = Y.#, X.salary ≥ Y.salary
DO update employee set X.salary = Y.salary
```

An active database system is able to autonomously perform certain operations in response to certain events occurring or certain conditions being satisfied. By contrast, in conventional database systems (passive ones) data is created, retrieved, modified and deleted only in response to explicit invocations by users or application programs. Active database systems are significantly more powerful than their passive counterparts: they can efficiently performs functions that in passive database systems must be encoded in applications, and they facilitate applications that are beyond the scope of passive database systems. Active rules can be used for integrity constraint monitoring and enforcement, for situation monitoring and alerting, for the specification of consequential actions or user notification upon certain situations.

The advantages of adding active rules to relational databases are well recognized. A relational database does not provide comprehensive means to formally state the semantics of data, the high-level semantic operations on data and the integrity constraints. Semantics is by contrast coded in application programs. Constraints and rules are hidden in application programs and it is thus difficult to enforce and maintain them. Since object-oriented databases incorporate behavior as well as data, it is less obvious why active rules are still useful in object-oriented systems. In relational active database systems, indeed, the behavior is specified either by rules or by applications; in object-oriented active database systems, by contrast, behavior can be specified by rules, by applications or by methods. However, specifying the semantics of data through rules has three important advantages over coding it into methods. First, the behavior represented by methods must be explicitly invoked by the user or by applications, while active rules are autonomously activated. Second, the semantics represented by a single rule, needs often to be replicated in the code of several methods. Consider for instance integrity constraints: since a single constraint can be potentially violated by the execution of different methods, the code for enforcing the constraint should be replicated in all those methods. Finally, a specialized trigger support, internal to the database system, supports a more efficient active behavior processing compared to the approach where the active behavior is coded into methods.

The integration of rules in an object-oriented data model introduces several interesting issues. First of all, the power and effectiveness of rule-based query and method definition languages have not been fully investigated yet. Moreover, the introduction of rules in a very rich data model, such as the object-oriented one, requires understanding how rules interact with object-oriented modeling constructs, such as inheritance, and how methods and rules can be combined. The aim of the thesis is to define an active and deductive object-oriented data model, with views and integrity constraints. Issues that will be addressed include: the formal definition of a reference object-oriented data model, typing of object-oriented rule languages, behavior refinement for methods expressed by rules, static analysis of transactions, integration of triggers with inheritance hierarchies, efficient rule support in object-oriented database systems.

The remainder of this proposal is organized as follows. Next section surveys related work. Section 3 outlines the research directions we plan to explore in the thesis, while Section 4 reports the research activities we have carried on so far. Finally, Section 5 concludes the proposal. Appendix A proposes a possible structure for the thesis, while Appendix B presents a work plan, with the explicit indication of the dependencies among different tasks to be performed during the thesis work.

2 Related Work

In this section we survey proposed approaches to the integration of rules in object-oriented data models. First, we introduce the basic notions of object-oriented databases and discuss the features of recently proposed object-oriented data models. Then, we discuss the efforts to incorporate deductive and active rules in object-oriented data models. Finally, we discuss the view models for object-oriented databases and integrity constraint support in object-oriented database systems.

2.1 Object-Oriented Data Models

Object-oriented DBMSs (OODBMSs) [BM93] combine the expressive power of object-oriented programming language with effective support for persistent data typical of database management systems. The object-oriented data paradigm can be briefly characterized by a number of concepts:

- Each real-world entity is modeled by an object. Each object is associated with a unique identifier, immutable during the object lifetime, that makes the object distinguishable from other objects.
- Each object has a set of instance attributes (properties) and methods (operations); the value of an attribute can be an object or a set of objects. The set of attributes of an object and the set of methods represent the object structure and behavior, respectively.
- The attribute values represent the object status. This status is accessed or modified by sending messages to the object to invoke the corresponding methods.
- Objects sharing the same structure and behavior are grouped into classes. A class represents a template for a set of similar objects. Each object is an instance of some class.
- A class can be defined as a specialization of one or more classes. A class defined as specialization is called a subclass and inherits attributes, messages, and methods from its superclass(es).

Note that there are many variations with respect to these basic concepts, especially when comparing OODBMSs and object-oriented programming languages. In fact, we use them mainly as a basis for our discussion, and we do not claim them to be a definition of the object-oriented paradigm. However, despite all differences, it has been widely recognized that this paradigm offers several advantages. First of all, information hiding: an object implementation may change without impacting other objects or applications using the services provided by the object. The inheritance mechanism favors re-usability of both object interfaces and implementations. Moreover, in most models, a subclass may override the definition of inherited methods and attributes. Therefore, inheritance lets a class specialize another class by *additions* and *substitutions*.

Specifically in the database field, the object-oriented paradigm brings other important advantages, in addition to the above ones. First, it allows complex objects to be directly represented by the model, without having to flatten them into tuples, as in the case of relational systems. Second, an OODBMS centralizes both data and operation on data, whereas and a traditional DBMS only centralizes data. Therefore, in a traditional DBMS the high-level semantic operations on data are dispersed among application programs. By contrast, application programming in an OODBMS is simplified, since it often consists of invoking and assembling predefined operations - the methods.

To date, several commercial OODBMSs have been developed. The most known ones are O_2 [D^+90], GemStone [BMO^+89], Iris/OpenODB [F^+89], Orion/Itasca [K^+89]. Among them, only Iris support a limited form of integrity constraints (through the use of key/non key qualifiers), intensionally defined attributes, declarative method implementation and the possibility of expressing views through functions not associated with a specific type. Moreover, extensions of O_2 with views [SdSAD94] and triggers [CCS94] are being independently investigated. More recent proposals of object-oriented data models incorporate, though to different extents, rules. The TM data model [BdBZ93], developed at the University of Twente, supports a deductive query language and allows declarative method implementation as well as integrity constraint specification. The *Cocoon* data model [SS90], from the University of Ulm, allows the specification of limited forms of integrity constraints and supports a view mechanism. Finally, the *Ode* database system [GJ91], developed at AT & T Bell Labs, is a prototype of active OODBMS, supporting integrity constraints and triggers.

The field of object-oriented databases suffers for the absence of a common reference data model, with a formal (that is, precise) definition. There have been, however, some formal definitions of object-oriented data models. Most of them originate from the work of Beeri [Bee89], who has been the first to clarify several issues concerning object-oriented data models. Other proposals formalize a core object-oriented data model as a basis for covering other issues, such as particular forms of integrity constraints in [ST93], behavior specification in [KM91], an algebra and object communication mechanisms in [Liu92]. Also the model developed in [VW91b] may be seen as a basis for the development of an algebra [VW91a]. However, those models do not adequately cover all features of a real object-oriented data model. Other models are concerned with specific aspects. In [ACO85] type hierarchies and subtyping problems are considered, but other aspects are not covered. In [SO90] a simple object model is formally specified as a basis for addressing query issues. The proposals in [GKS91] and [Pet94] mainly concern the specification of object behavior.

As far as languages are concerned, two different kinds of languages are of interest in object-oriented databases: the query language (as in relational databases) and the method definition language. An important requirement of the query language is declarativity which results in ease of use and provides potential for query optimization. Different declarative query languages for object-oriented databases have been proposed [AB88, BNPS92] and some preliminary analysis on their expressive power have been performed. The use of deductive languages as query languages for object-oriented databases has also been proposed, as we will see in Section 2.2. For what concerns method definition language, in most object-oriented databases methods are specified in an object-oriented programming language (for instance, SmallTalk for GemStone, an object-oriented extension of C, CO_2, for O_2, C++ for Ode). The use of a rule-based method definition language has not been much investigated yet. The use of rule-based method definition languages has several advantages: declarativity, set-orientedness, the possibility of exploiting efficient query execution techniques. Another important advantage is that the behavior of methods expressed through rule-based languages can be clearly analyzed in terms of the set of updates performed.

2.2 Deductive Rules in Object-Oriented Databases

Several research proposals have attempted to combine object-orientation, databases and logical languages. We refer the reader to [BGM95] for a survey of deductive object-oriented data models. A first research direction uses logic to formalize the notions underlying object-oriented data models, resulting in the so called logics for objects. These proposals originate from Maier's O-logic [Mai86], and evolve in C-logic [CW89] and the revised O-logic [KW89]. These proposals are also strictly related to the LOGIN language [AKN86]. The final effort in this direction, which extends the previous ones, is *Frame Logic* (F-logic) [KL90]. It extends previous works with higher-order features to accommodate methods within complex terms. The motivations underlying those approaches is the development of a logical framework for representing complex objects, thus overcoming the absence of formal foundations for object-oriented data models. Those approaches attempt to define model-theoretic semantics for object-oriented logics. The approaches in this area consider (complex) objects as terms, take into account a notion of schema; no update and evolution aspects are considered.

Another research direction is that of employing logical languages as query languages for object-oriented databases, leading to the definition of object-oriented logical languages primarily used for querying object-oriented databases. The first approach in this direction has been COL (*Complex Object Language*) [AG90], a Datalog extensions to deal with structured objects. It is a query language whose corresponding data model is not a 1NF relational model, rather an extended relational model. IQL (*Identity Query Language*) [AK89] is an extension of COL with the concepts of object identity and inheritance. It is centered on the concept of object identifier (oid) as a powerful primitive for database query languages. IQL is obtained by enriching Datalog to obtain an object based language with sufficient expressive power to invent oid. Other proposals along this direction are LLO (*Logic Language for Objects*) [LO91], a declarative query language including also methods, and DTL (*DataTypeLog*) [BB93], the query language of the TM data model.

The last direction we consider[1] is that of extending Datalog to specify object-oriented databases. These systems have a built-in deductive approach, that is, they do not use logic only as a query language rather they deeply use logic in the specification of the data model. Thus, these systems can be regarded as object-oriented extensions of Datalog. In such extensions, the logical component is used to specify the schema of the database, and a distinction is made between base relations and derived relations. Examples of such systems are LOGRES [CCCR$^+$89], COMPLEX [GLR90] and Datalog^{meth} [ALUW93].

[1] Other proposals exploit deductive rules to define intensionally populated classes. These proposals will be considered in Section 2.4.

2.3 Active Rules in Object-Oriented Databases

Most of the research and development efforts on active databases have focused on the relational model. Recently, however, several approaches have been proposed incorporating active rules into object-oriented database systems. One of the earliest object-oriented active database projects is HiPAC [MD89, DBC96], started in 1987. HiPAC provides an expressive active database rule language with flexible execution semantics, supporting a full complement of coupling modes through a nested transaction model. There are some projects on active object-oriented databases that are follow-on of HiPAC: among them, the *Sentinel* [AMC93] and the REACH projects [BBKZ93]. One of the best known active object-oriented database systems is Ode [GJ91, GJS92, GJ96], which extends the O++ database programming language with facilities for expressing rules in the form of constraints and triggers. A prototype implementation of Ode has recently been completed [LGA96]. Other projects integrating active rules in object-oriented data models are Adam [DPG91], NAOS [CCS94], SAMOS [GD92] and TriGS [KRSR94]. Among the considered systems, NAOS and TriGS are extensions of commercial object-oriented database systems with active capabilities. All the proposals support ECA rules; however, in some systems, such as Ode, conditions are part of the event specification as a mask which qualifies the event and can refer to parameters of the method call defining the event.

The active rule languages supported by the different systems present some differences. First of all, different kinds of events can be monitored: method calls, database primitives, temporal events; triggering events may also be *composite*, that is, combinations of other events. Some systems allow events to be parameterized; when a parameterized event occurs, values related to the event are bound to the event parameters, and these parameter values can be referenced in rule conditions or actions. Some systems only consider as an implicit parameter the identifier of the object receiver of the event, that is, the object on which the rule is executed. As far as the condition component is concerned, in all languages the condition is a predicate over the database. If the rule language supports parameterized events, the condition specification language includes a mechanism for referencing the values bound to the event parameters. In some systems, values related to the condition can be passed to the action. In the most common approach for parameter passing, the condition is expressed as a query returning data, that are then passed to the rule action. Moreover, because in triggers it can be useful to refer to past database states, some systems support this possibility. The considered systems differ not only with respect to the supported rule language, but also for the rule execution semantics. Different rule activation times are supported, thus different rule processing granularities (as soon as the triggering event occurs or at the end of the transaction), and there are different behaviors with respect to cascade rule triggering and to conflict resolution for selecting a rule to execute among the triggered ones.

There are several factors not present in relational database systems that complicate the extension of object-oriented database systems to include active behavior. Among them, let us mention that in object-oriented data models, in contrast to a fixed number of predefined primitive events of the relational model, every method/message is a potential event. Moreover, issues related to scope, accessibility, and visibility of object states with respect to rules are not yet fully clear. Among existing systems some force rule definitions to be attached to classes. Attaching rule definitions to classes enhances modularization and allows an efficient detection of relevant events while there are sometimes useful rules triggered by events spanning sets of objects possibly from different classes (untargeted rules). Some important, open issues concern trigger inheritance and overriding. Because inheritance is a central notion of the object-oriented data model, the definition of a proper approach for rule inheritance is crucial. However, those issues have been largely neglected by current implementations and research proposals, or only simplistic solutions are adopted. Those solutions are inadequate to model the large variety of rule semantics, that arise in practical applications. The approach taken by the majority of the systems for rule inheritance is to simply apply all rules, defined in a class, to the entire extent of the class, that is, to all the instances as well as to the members of the class[2]. No rule overriding is supported by those systems. Only one system [KRSR94] supports rule overriding, though in a completely uncontrolled way.

2.4 Views in Object-Oriented Databases

In the relational model, a view is a virtual (that is, not physically stored) relation, defined by a query on one or more stored relations. Relational languages are closed, that is, the result of a query expressed in a relational language is a relation, thus the relation returned by such a query represents the view content. Thus, relational views can be used in (almost) any query in which a relation may appear. Moreover, authorizations may be granted and revoked on views as on ordinary relations. At the same time, views can be used to define external schemas, thus customizing the database to different applications. The definition of a view mechanism has been recognized to be a fundamental aspect also for the practical development of object-oriented applications. In this new context, views should be still used as shorthand in queries, should support the integration of heterogeneous databases, should be the basis for content-dependent authorization and, finally, should support the simulation of schema changes. Indeed, views allow to dynamically modify a database schema but yet retaining its older versions, a very important utility for advanced applications.

Unfortunately, the definition of an object-oriented view mechanism does not come for free from the relational approach. Indeed, the object-oriented model is far more complex than the relational one. Whereas a relational schema consists of a set of independent relations, in an object-oriented schema classes are connected by inheritance relationship. Therefore, an object-oriented view model must face the problem of determining the position of a view in the class hierarchy. We refer to this problem as *view integration problem*. Three kinds of solutions have been proposed to solve it:

1. Views are automatically positioned in the class hierarchy by integration algorithms [AB91, Run92, SLT90].

2. The user explicitly specifies the position of the view in the class hierarchy [KKS92].

3. The view model extends the object-oriented schema with a new dimension: the *view derivation relationship* between a base class and a view, whose semantic is “the view is derived from the base class” [Ber92].

As a second problem, objects have an identity. A view, at the data level, should be a class. However, we can distinguish two distinct kinds of views [HS91]:

- *object-preserving views*: they are views only extracting elements from existing classes; the instances of these views are identified by the identifiers of the extracted base objects.

- *object-generating views*: they are views creating new objects; the instances of these views must be identified by newly generated object identifiers.

[2]We distinguish between instances and members of a class as follows. An object is an instance of a class if this class is the most specialized class in the inheritance hierarchy to which the object belongs. An object is member of a class if it is an instance of this class or an instance of any subclass of this class.

A number of approaches [SS89, SLT90, SJJN93] only consider object-preserving views. Thus, views result in providing different views of existing objects. This is particularly useful for supporting objects with multiple interfaces and context-dependent behavior. In this sense, object-preserving views are similar to *roles* [ABGO93, RS91]. However, this kind of views is not powerful enough for supporting all kinds of database reorganization. By contrast, object-generating views are proposed in [Ber92] for supporting schema evolution, so that the evaluation of a query always returns new objects.

2.5 Integrity Constraints in Object-Oriented Databases

Integrity constraints are declarative specifications of properties that must be satisfied by any database instance. In object-oriented database systems, constraints are typically used to restrict the admissible extent of a class, to impose restrictions on the state of individual objects, or to specify general laws that must be obeyed by sets of objects possibly belonging to different classes. Among the most common types of constraints let us mention referential integrity, ensuring the existence of each referenced object, and the exclusiveness and dependency constraints associated with composite objects. The notion of composite object refers to the semantic notion of "part-of" [KBG89]. This notion models the fact that an object (called component object) is part of another object (called composite object). This semantic notion may influence the way certain operations are performed; for example, the deletion of a composite object may imply the deletion of all its component objects.

Integrity maintenance in object-oriented database systems has so far received limited attention. Most mechanisms for constraint enforcement supports very limited types of constraints. This support consists usually in either forbidding or rolling back the operations that violate constraints. However, this "everything or nothing" approach, by which whenever an inconsistent instance is produced, the transaction that caused the inconsistency is rolled back, causes high run-time overhead for consistency checking and rollback. Moreover, it leaves the burden of writing consistent transactions to the user. A more flexible approach to enforcement is to extend the system with capabilities for checking whether constraints have been invalidated by a transaction and repairing integrity violations, for instance by performing compensating actions.

There are presently two approaches to ensure that the database satisfies its constraints after certain actions. The first one is based on modifying methods in accordance to the specified integrity constraints. This direction is followed by the works of Schewe et al. [ST93, STSW92] and by the TM data model [BdBZ93, GdBA94]. Under this approach, integrity maintenance is obtained by an appropriate method modification. Those approaches, however, modify methods in different ways. In the approach of Schewe et al., given some arbitrary methods and some constraints, a *consistent specialization* of the methods with respect to the constraints is built. This consistent specialization is a new set of methods, performing the "same" actions, but without violating the constraints. This approach handles a limited set of constraints. By contrast, in the approach used in TM [GdBA94], for each update method an adorned version is created that checks the constraints of the class to which the method refers. For each method, an adorned method is created such that if it calls an update method, it calls its adorned version. Constraint checks are translated into preconditions on the actual data manipulation. Designers are allowed to define corrective actions upon expected constraint invalidation. This can be done on a per-method and per-constraint basis and may result in several adorned method versions, depending on application requirements.

The second approach relies on the use of active rules for integrity enforcement. Indeed,

	Our model	GemStone	Iris	O2	Orion	TM	Ode	Cocoon
Reference		[BMO+89]	[F+89]	[D+90]	[K+89]	[BdBZ93]	[GJ91]	[SS90]
User-defined value types	YES	NO	NO	YES	NO	YES	NO	NO
Value collections	YES	NO	NO	NO	NO	NO(1)	NO	NO
Class attributes and class methods	YES	YES	NO	NO	YES	YES	NO	NO
Multiple inheritance	YES	NO	YES	YES	YES	YES	YES	YES
Object migration	YES	NO	NO	NO	NO	NO	NO	YES
Multiple class direct membership	YES	NO	YES(2)	NO	NO	NO	NO	NO
Composite objects	YES	NO	NO	NO	YES	NO	NO	NO
Referential integrity	YES	YES	NO	YES	NO	YES	YES	YES
Integrity constraints	YES	NO	L.F.	NO	NO	YES	YES	L.F.
Intensionally defined attributes	YES	NO	YES	NO	NO	NO	NO	NO
Declarative method implementation	YES	NO	YES	NO	NO	YES	NO	NO
Triggers	YES	NO	NO	NO	NO	NO	YES	NO
Views	YES	NO	YES(3)	NO	NO	NO(1)	NO	YES

Legenda: L.F. limited form

(1) Value classes and views could be simulated in TM by using the module mechanism provided by this language [BdBZ93].

(2) In Iris arbitrary types can be added to an object. To avoid conflicts, however,two different types of an object must not have different methods with the same name.

(3) Views in Iris are supported as functions implemented in OSQL which are not associated with a specific type.

Table 1: Comparison of the reference data model with other o-o data models

integrity constraints can be expressed by triggers, activated by any event likely to violate the constraint. The condition of the trigger tests for the actual occurrence of violations; if a violation is detected, then the action of the trigger either issues a rollback command or performs database manipulations for repairing constraint violations. This approach is the one taken by the Ode object-oriented database system [GJ91], by the proposal for constraints in the O_2 object-oriented database system [MP91], by the works of Urban et al. [UD90, KU91] and those of Ceri et al. [CFP94, FPT92]. The set of triggers representing the constraints on the database must be however analyzed (either at design time or run-time) to ensure that they exhibit the desired behavior with respect to termination and determinism (uniqueness of final state) [FPT92, KU91].

3 Research Plan

In this section we illustrate the main research directions we plan to explore in the thesis. The features of the data model we aim at developing are compared in Table 1 with those of existing object-oriented data models.

3.1 Formal Definition of a Reference Object-Oriented Data Model

The first step for investigating the problems resulting from the introduction of rules in an object-oriented data model is the choice of a reference data model. As stated in Section 2.1 there is no general accepted reference object-oriented data model, and formal definitions exist only for *toy* data models, not covering all the features of the models of real object-oriented databases. Thus, the first effort is the formal definition of a reference object-oriented data model, to obtain a formal basis on which the issues related to rules can be addressed. Such definition is quite a challenging task due to the richness of the target data model. Indeed the model should cover a number of modeling features not covered by already developed models, among them let us mention the distinction between values and objects (and thus between classes and types at the intensional level), the support for named and constrained value types as well as for explicitly populated value collections, the presence of class attributes and class methods, the possibility of defining attributes intensionally. Another important characteristic of our reference model is the support for object migration, that is, the possibility for an object to change its class membership. As a second step, typing problems need to be considered in the context of the developed object-oriented data model. This means considering restrictions on the redefinition of inherited features, by giving a precise characterization of type refinement, and choosing of an attitude towards type checking (static or dynamic).

A peculiarity of our reference object model is that objects can belong to many most specific classes, that is, an object can be instance of multiple classes. There are different motivations for this. The most intuitive one is that in real world objects do belong to several most specific classes; in a model with multiple inheritance this can be handled by defining a subclass of all the involved classes, this solution, however, leads to a lot of artificial subclasses, sometimes referred to as *intersection classes*. Moreover, there are models supporting derived classes[3], that is, classes whose extents are not explicitly manipulated; rather those classes are implicitly populated in that a population predicate is associated with the class specifying sufficient and necessary conditions for an object to belong to the extent of the class. In such models, it is difficult to ensure that an object belong to a unique most specific class, because it depends on the population predicates being disjoint. The need of overcoming the restriction of the exclusive link between an object and its most specific class has been stressed in [DDH92].

When objects belong to several most specific classes, conflicts among different definitions may arise. Indeed, if an object has several most specific classes, the object takes the union of the features of all the classes to which it belongs. Such conflicts resemble conflicts due to multiple inheritance. However, conflicts due to multiple inheritance can be detected at compile time whereas an object may become and cease to be an instance of a class at run time; the situation is thus more complicate for multiple class direct membership. If an object can reside in more than one class, there are additional problems in determining *the* class of an object which implies the method implementation to be executed. There is a fundamental choice to be taken for solving these conflicts: whether or not taking care of the context from which the feature is referred. Of course, a context-dependent approach is more suitable for view support. Different dispatching techniques for objects with multiple most specific classes must be analyzed, and compared with respect to a number of dimensions, among which we would like to mention the efficiency of dispatching, the completeness of the technique, and whether or not the technique allows for static type checking.

3.2 Rule-based Query and Method Definition Languages

A main goal of the thesis is to investigate the effectiveness of rule-based query and method definition languages. The approach we plan to take is to exploit the deductive rule language as query language, and the active rule language as method definition language, where the

[3]Note that also object-preserving views can be seen as derived classes.

event activating the execution of the trigger implementing a method is the call of the corresponding method. In the context of this goal, a type system for rules must be defined. This type system is exploited to check the type correctness of queries and operations. We remark that typing issues in object-oriented rule languages have not been much investigated (an exception is the DTL query language [BB93]).

3.2.1 Query Language

As far as the query language is concerned, we have two primary goals. First, we plan to compare our rule-based query language with well-accepted query languages for object-oriented databases [AB88, BNPS92, BP91] in terms of expressive power. Subsets of our query language excluding or limiting recursion, will be considered, too. Transformations between the languages will be investigated.

A second goal is the investigation of the problem of query containment (or subsumption) in the context of (restrictions of) our reference query language. A query Q is said to contain, or subsume, a query Q' if for any database state the answer set of Q contains in the answer set of Q'. Query subsumption has been deeply investigated in the context of relational query languages [Ull89], but little work has been done for object-oriented query languages [BJNS94]. This problem is however very relevant for our model. Indeed, relational query subsumption is mainly employed for (semantic) query optimization. In addition to this application, object-oriented query subsumption is related to two important issues in active, deductive, object-oriented data models, namely trigger refinement and view inheritance relationships. Indeed, for a trigger semantics to be preserved in subclasses, it must be ensured that trigger overriding is allowed provided that the condition of the refined trigger subsumes the condition of the inherited one. This ensures that the refined trigger is activated each time the inherited trigger would be. Query subsumption is also related to view support. Indeed, for an object-oriented view mechanism to be consistent with the basic principles of the object-oriented paradigm, views should be organized in an inheritance hierarchy. To ensure that the extent of a view is contained in the extents of its superviews, the query defining the view must be subsumed by the queries defining its superviews.

3.2.2 Method Definition Language

As far as the method definition language is concerned our primary goal is to address the practical effectiveness of a rule-based method definition language. To this aim, we must specify both the syntax and the semantics of such a language. The basic idea is to express methods as *condition* $\rightarrow$ *action* constructs, where the condition is a declarative formula and the action is a sequence of update operations and method calls. Methods with an empty action part are side-effect free and can be invoked in conditions of other methods (and in any formula of our deductive language). The most relevant aspect of such a rule-based language is its set-orientedness, since the action is executed on the set of bindings satisfying the condition. This implies the possibility of exploiting efficient query evaluation techniques when executing methods. To the best of our knowledge, the only work concerned with set-oriented methods in object-oriented databases is [ACPvdB95].

The practical usefulness of such a method definition language will be investigated, and the possibility of introducing constructs (such as sequence, parallel, non-deterministic choice) for combining the various rules defining a method could be considered. Since an important advantage of methods expressed through rule-based languages is that method behavior can be clearly characterized in terms of the set of updates performed, this offers potential for investigating issues such as behavior preserving for method overriding and static analysis of transactions.

3.3 Triggers

For what concerns the active component of our model, a first goal of the thesis is to clarify a neglected issue in existing active object-oriented data models, namely trigger inheritance and overriding. As we have anticipated in Section 2.3, existing systems provide only simplistic solutions with respect to trigger inheritance and only one system [KRSR94] supports trigger overriding. In our opinion, there are several cases in which rule overriding is useful. However, we believe that rules should not be overridden in a uncontrolled way, rather they should only be *refined* in subclasses, that is, they can be overridden provided that (*i*) the redefined rule is triggered each time the overridden one would do; (*ii*) the redefined rule does at least what the overridden one would do. Syntactical condition ensuring that will be defined.

Trigger inheritance, moreover, require to revisiting notions such as trigger priority and method selection in trigger actions in the context of inherited triggers. Moreover, the influence of multiple inheritance and multiple class direct membership on triggers will be considered, too. For multiple inheritance, the main issue is how to order triggers (on the same events) inherited from different superclasses; this could be achieved by imposing a total order on classes, or by allowing a class to modify the relative priorities of triggers in its superclasses. Note moreover that in these cases it is likely to be rather frequent that a trigger becomes redundant. Conditions for trigger redundancy with respect to a class will also be defined.

Another important neglected aspect in active object-oriented databases concerns trigger semantics. Little work indeed has been carried on towards the specification of a semantic framework for triggers. This, and the fact that the execution model varies from system to system, constitute a serious obstacle to the diffusion of active database systems. Some preliminary work towards the clarification of trigger semantics has been reported in [FT94, CC95]. The definition of a semantics suitable for the active component of our model is one of our goals, too.

3.4 Integrity Constraints

As far as integrity aspects are concerned, in the thesis we are mainly interested in identifying classes of integrity constraints peculiar to object-oriented data models. Two important categories of constraints are respectively related to the two main relationships among classes in an object-oriented data model, namely aggregation and inheritance relationships. The most relevant constraint related to the aggregation relationship is referential integrity. Referential integrity ensures that whenever the value of an attribute of an object is another object, the referenced object exists. Violations to referential integrity must be reported on attempts to delete a referenced object. Violations to referential integrity can be repaired by forbidding the deletion, by setting to null the value of the attribute containing the reference or by cascade deleting also the object containing the reference.

A less obvious class of constraints related to the aggregation relationship concerns composite object references. A "part-of" semantic relationship can indeed be superimposed on a reference between a pair of objects. The part-of relationship models the notion that an object is a part of another object. A set of component objects which form a single logical entity is called a composite object [KBG89]. Two different kinds of constraints are related to composite object references, namely exclusiveness and dependency. An exclusiveness constraint specifies that an object can be component of only a composite object (possibly, with respect to a certain class or through a certain attribute) while a dependency constraint specifies whether the existence of the component object is independent from the existence of the composite objects containing it, that is, if the deletion of a composite object implies the deletion of its components.

Integrity constraints related to the inheritance relationship include constraints stating that a set of subclasses should be a covering of a class (that has thus no proper instances), or that two classes must have disjoint extents. Other interesting constraints are related to object migration, to be restricted to semantically meaningful class changes [Su91, Zdo90].

One of the aims of the thesis is to investigate triggers for the enforcement of this kind of constraints and to study the behavior of the set of obtained triggers with respect to confluence and termination[AWH92]. A very preliminary attempt in this direction, concerned with referential integrity in the relational model, is reported in [BCP94].

3.5 Views

For what concerns view support, we aim at developing a view mechanism for object-oriented databases well-suited to all view applications, namely external schemas, authorization and schema evolution. As we have anticipated in Section 2.4, the existing view models are tailored to specific applications. This has a strong influence on the model, since, for instance, schema evolution requires object-generating views while object-preserving views are well suited for external schemas and authorization. A basic characteristic of the view model we aim at defining is the support for both object-generating and object-preserving views. Note that the support of object-preserving views is strictly related to multiple class direct membership discussed in Section 3.1.

Concerning the view integration problem, the better solution seems extending the schema with a new dimension: the *view derivation relationship* between a base class and a view, whose semantic is "the view is derived from the base class" [Ber92]. Indeed, both the other approaches present some disadvantages. The approach of automatically positioning views in the class hierarchy by integration algorithms [AB91, Run92, SLT90] may cause the inheritance hierarchy to become very large because of the introduction many intermediate classes, not semantically meaningful. The approach of letting the programmer explicitly specify the position of the view in the class hierarchy [KKS92], results in a mix of classes and views in the same hierarchy that may cause confusion in the user.

The view derivation relationship is orthogonal to the inheritance relationship. An inheritance relationship among views is also introduced. This relationship organizes views in a ISA hierarchy, in the same way as classes are organized. The main difference between the ISA hierarchy on base classes and the one on views is that the inheritance hierarchy on views must take into account the query component of the view. This is due to the fact that a view is not explicitly populated, as a class; rather its population is derived from the population of its root classes. Thus, to ensure that the instances of a view are a subset of the instances of its superviews, the restriction must be imposed that the query of a view must be subsumed by the queries of its superviews.

The problem of view update propagation will be addressed in the thesis, too. The view update problem has been widely investigated in the relational context. In object-oriented data models, the existence of object identifiers simplifies the problem, since it is always possible to establish a mapping among a view instance and the base objects from which it has been derived. A preliminary investigation of the problem, limited to object-preserving views, is reported in [SLT91]. Moreover, object-oriented data models offer the possibility of specifying in methods how to propagate ambiguous updates, such as deletions on views defined as joins, which are forbidden in the relational context.

Finally, the use of triggers for view handling is an interesting possibility, as suggested in [CW91] for the relational context. If views are materialized (that is, stored in the database) rules can monitor dynamic changes to base data and modify relevant views. If, by contrast, virtual views are supported, rules can dynamically detect queries on virtual views and transform them into queries on base data, by composing the user query with the query defining the view. Finally, rules can propagate view updates to base data.

3.6 Architectural Aspects

In the thesis, we will address how the features we support in the proposed data model can be efficiently implemented. Indeed, performance is a very important requirement for practical database system usage. In particular, we plan to investigate how an efficient trigger run-time support can be realized, and to address some architectural problems concerning object migrations, multiple class direct membership and dispatching, trigger inheritance, view handling.

3.7 Temporal Capabilities

Conventional database systems do not offer the possibility of dealing with time-varying data. The content of a database represents a *snapshot* of the reality in that only the current data are recorded, without the possibility of maintaining the complete history of data over time. If such a need arises, data histories must be managed at application program level, thus, making the management of data very difficult, if at all possible. To overcome this lack in the past years there has been a growing interest in extending data models with temporal capabilities [TCG+93]. Although many temporal extensions to the relational data model have been proposed, there is no comparable amount of work in the context of object-oriented data models. The last goal of the thesis will be to discuss some of the problems one has to face when adding temporal capabilities to an active, deductive, object-oriented data model. Notions such as object consistency and object equality must be revisited; moreover, the need arises to handle in a uniform way temporal and non-temporal domains and to define suitable restrictions on domain refinement in subclasses. Then, we will address how methods, triggers and deductive rules can be extended with time. A temporal integrity constraint language will allow, among other things, to express constraints based on past histories of objects.

4 Current Status of Research

The aim of this section is to discuss the work already developed with respect to the directions presented in Section 3. The state of the research for the specific topics is summarized in Table 2. The state of the research quantifies the amount of research already done, with a number between 0 and 3. For topics we have started investigating, a reference is given to the papers and reports presenting the preliminary results. In particular, for what concerns the method definition language, we have investigated the effectiveness of a preliminary rule-based language in an interesting application domain, for expressing interference queries on hierarchical terrain models. For the two topics for which most of the planned work has been completed, namely, the definition of a reference object-oriented data model and the definition of a view model for it, we present in Table 3 and Table 4, respectively, a comparison with already existing proposals.

Basic Goal	Research State (0 ...3)	Reference
reference object-oriented data model	2	[GBB94, BG95b]
query language	0	
method definition language	1	[GM95]
triggers	1	[BG95a]
integrity constraints	0	
views	2	[BCGMG95]
architectural aspects	1	[CGMR95]
temporal capabilities	1	[BFG96]

Table 2: State of the research

Let us mention, moreover, that our previous work on deductive object-oriented databases [BCG93, BGM94, BGM95] helped us in clarifying to which extent deductive rules can be integrated in object-oriented data models. Similarly, the techniques for transaction analysis and optimization in the context of deductive languages supporting updates [BCGM94] could be useful for analyzing method behavior in our model.

5 Conclusions

If we reason on the evolution of database technology, we observe a progressive attempt of subtracting semantics from applications and adding it to the generic, system-controlled component of the database. First, applications were decoupled from the physical data representation on files (physical independence); next, applications were decoupled from the actual schema structure used in the database (logical independence). The next step in this evolution can be considered as the achievement of knowledge independence, that is, the decoupling of applications from generic, application-independent, shared knowledge. Object-oriented data models, deductive data models, and active data models can be seen, though to different extents, going towards this direction. However, the areas of active, deductive and object-oriented database systems have primarily developed as separate research areas. Although significant results have been achieved in each area individually, there are weaknesses in each type of database system that can be overcome by combining the strengths of each. The successful integration of these areas has the potential to provide the kind of powerful database processing environment that will be required by database applications of the future.

Thus, the integration of active and deductive rules in an object-oriented data model is a very relevant topic of great current interest. This thesis will result in a proposal towards this goal. We plan to use rule-based query and method definition languages in our reference object-oriented data model, clarifying their expressive power and their practical usefulness. Issues related to a clean integration of rules with object-oriented modeling capabilities, such as methods and inheritance, will be clarified. Moreover, the use of triggers for integrity constraint enforcement and of deductive rules for view support will be investigated. Finally, the thesis will address architectural aspects concerned with rule support in object-oriented database systems, since the efficiency of rule processing is essential to get active, deductive, object-oriented databases be used in practise.

	[AHV95, LRV90]	[ST93]	[KM91]	[VW91b]	[Liu92]	Our model
Complex Objects	Partially*	YES	Partially*	YES	YES	YES
Inheritance	Single	Multiple	No	Single	Multiple	Multiple
Classes & Types	Types only	Both	Sorts only	Both	Both	Both
Relations	NO	NO	YES	NO	NO	NO
Schema - Instance Levels	YES	YES	YES	YES	YES	YES
Feature Refinement	YES	NO	NO	Attributes only	YES	YES
Type System	NO	NO	NO	NO	NO	YES
Class Features	NO	NO	NO	NO	NO	YES
Named Types	NO	NO	NO	NO	NO	YES

Legenda: * Partially denotes the non-complete orthogonality in applying constructors to arbitrary values

Table 3: Comparison of our o-o data model with other formally defined o-o data models

	[Ber92]	[Run92]	[SLT90]	[SdSAD94]	Our model
data model	general	MultiView	COCOON	O_2	ref. o-o model
query language	object-oriented predicate calculus	Object algebra	Object algebra	O_2	deductive rules
external schemas	NO	YES	NO	YES	YES
schema evolution	YES	YES	L.F.	L.F.	YES
view integration problem	separate view hierarchy	views inserted in class hierarchy	views inserted in class hierarchy	separate view hierarchy	separate view hierarchy
augment class definition	YES	YES	NO	NO	YES
object-preserving/ object-generating	generating	generating	preserving	both*	both

Legenda: L.F. limited form
* Actually, a query may return a set of tuples that are converted to new objects outside the query language.

Table 4: Comparison of the developed view model with other o-o view models

References

[AB88] S. Abiteboul and C. Beeri. On the Power of Languages for the Manipulation of Complex Objects. Technical Report 846, INRIA, Rocquencourt, 1988. To appear in *Journal of ACM*.

[AB91] S. Abiteboul and A. Bonner. Objects and Views. In J. Clifford and R. King, editors, *Proc. of the ACM SIGMOD Int'l Conf. on Management of Data*, pages 238–247, 1991.

[ABGO93] A. Albano, R. Bergamini, G. Ghelli, and R. Orsini. An Object Data Model with Roles. In R. Agrawal, S. Baker, and D. Bell, editors, *Proc. Nineteenth Int'l Conf. on Very Large Data Bases*, pages 39–51, 1993.

[ACO85] A. Albano, L. Cardelli, and R. Orsini. Galileo: A Strongly-Typed, Interactive Conceptual Language. *ACM Transactions on Database Systems*, 10(2):230–260, June 1985.

[ACPvdB95] M. Andries, L. Cabibbo, J. Paredaens, and J. van den Bussche. Applying an Update Method to a Set of Receivers. In *Proc. of the Fourteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, 1995.

[AG90] S. Abiteboul and S. Grumbach. COL: A Logic Based Language for Complex Objects. In F. Bancilhon and P. Buneman, editors, *Advances in Database Programing Languages*, pages 347–377. ACM Press, 1990.

[AHV95] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases.* Addison-Wesley, 1995.

[AK89] S. Abiteboul and P. Kanellakis. Object Identity as a Query Language Primitive. In *Proc. of the ACM SIGMOD Int'l Conf. on Management of Data*, pages 159–173, 1989.

[AKN86] H. Aït-Kaci and R. Nasr. Login: a Logic Programming Language With Built-In Inheritance. *Journal of Logic Programming*, 3:185–215, 1986.

[ALUW93] S. Abiteboul, G. Lausen, H. Uphoff, and E. Waller. Methods and Rules. In P. Buneman and S. Jajodia, editors, *Proc. of the ACM SIGMOD Int'l Conf. on Management of Data*, pages 32–41, 1993.

[AMC93] E. Anwar, L. Maugis, and S. Chakravarthy. A New Perspective on Rule Support for Object-Oriented Databases. In P. Buneman and S. Jajodia, editors, *Proc. of the ACM SIGMOD Int'l Conf. on Management of Data*, pages 99–108, 1993.

[AWH92] A. Aiken, J. Widom, and J. Hellerstein. Behavior of Database Production Rules: Termination, Confluence, and Observable Determinism. In M. Stonebraker, editor, *Proc. of the ACM SIGMOD Int'l Conf. on Management of Data*, pages 59–68, 1992.

[BB93] R. Bal and H. Balsters. A Deductive and Typed Object-Oriented Language. In S. Tsur, S. Ceri, and K. Tanaka, editors, *Proc. Third Int'l Conf. on Deductive and Object-Oriented Databases*, number 760 in Lecture Notes in Computer Science, pages 340–359, 1993.

[BBKZ93] H. Branding, A. Buchmann, T. Kudrass, and J. Zimmerman. Rules in an Open System: the REACH Rule System. In N. Paton and M. Williams, editors, *Proc. First International Workshop on Rules in Database Systems*, Workshops in Computer Science, pages 111–126, 1993.

[BCG93] E. Bertino, B. Catania, and G. Guerrini. An Overview of LOL: a Deductive Language for Object Bases. In A. Makinouchi, editor, *Proc. International Symposium on Next Generation Database Systems and Their Applications*, pages 69–76, 1993. Invited Paper.

[BCGM94] E. Bertino, B. Catania, G. Guerrini, and D. Montesi. Transaction Optimization in Rule Databases. In J. Widom and S. Chakravarthy, editors, *Proc. Fourth International IEEE Workshop on Research Issues in Data Engineering - Active Database Systems*, pages 137–145, 1994.

[BCGMG95] E. Bertino, B. Catania, J. Garcia-Molina, and G.Guerrini. A Formal Model of Views for Object-Oriented Database Systems. Submitted for publication, 1995.

[BCP94] E. Baralis, S. Ceri, and S. Paraboschi. Declarative Specification of Constraint Maintenance. In P. Loucopoulos, editor, *Proc. Thirteenth Int'l Conf. on the Entity-Relationship Approach*, number 881 in Lecture Notes in Computer Science, pages 205–222, 1994.

[BdBZ93] H. Balsters, R. A. de By, and R. Zicari. Typed Sets as a Basis for Object-Oriented Database Schemas. In *Proc. Seventh European Conference on Object-Oriented Programming*, number 707 in Lecture Notes in Computer Science, 1993.

[Bee89] C. Beeri. Formal Models for Object Oriented Databases. In W. Kim et al., editors, *Proc. First Int'l Conf. on Deductive and Object-Oriented Databases*, pages 370–395, 1989.

[Ber92] E. Bertino. A View Mechanism for Object-Oriented Databases. In M. L. Brodie and S. Ceri, editors, *Proc. Third Int'l Conf. on Extending Database Technology*, pages 136–151, 1992.

[BFG96] E. Bertino, E. Ferrari, and G. Guerrini. A Formal Temporal Object-Oriented Data Model. In P. Apers, editor, *Proc. Fifth Int'l Conf. on Extending Database Technology*, Avignon (France), March 1996. To Appear.

[BG95a] E. Bertino and G. Guerrini. A Formal Model of Triggers for the Chimera Active Object Language. Submitted for publication, 1995.

[BG95b] E. Bertino and G. Guerrini. Objects with Multiple Most Specific Classes. In W. Olthoff, editor, *Proc. Ninth European Conference on Object-Oriented Programming*, number 952 in Lecture Notes in Computer Science, pages 102–126, 1995.

[BGM94] E. Bertino, G. Guerrini, and D. Montesi. Deductive Object Databases. In M. Tokoro and R. Pareschi, editors, *Proc. Eighth European Conference on Object-Oriented Programming*, number 821 in Lecture Notes in Computer Science, pages 213–235, 1994.

[BGM95] E. Bertino, G. Guerrini, and D. Montesi. Towards Deductive Object Databases. *Theory and Practice of Object Systems*, 1(1):19–39, Spring 1995. Special Issue: Selected Papers from ECOOP '94.

[BJNS94] M. Buchheit, M. Jeusfeld, W. Nutt, and M. Staudt. Subsumption Between Queries to Object-Oriented Databases. In M. Jarke, J. Bubenko, and K. Jeffery, editors, *Proc. Fourth Int'l Conf. on Extending Database Technology*, number 779 in Lecture Notes in Computer Science, pages 15–22, 1994. Extended version in *Information Systems*, 19(1).

[BM93] E. Bertino and L.D. Martino. *Object-Oriented Database Systems - Concepts and Architecture*. Addison-Wesley, 1993.

[BMO+89] R. Breitl, D. Maier, A. Otis, J. Penney, B. Schuchardt, J. Stein, E. H. Williams, and M. Williams. The GemStone Data Management System. In W. Kim and F. H. Lochovsky, editors, *Object-Oriented Concepts, Databasases, and Applications*, pages 283–308. Addison-Wesley, 1989.

[BNPS92] E. Bertino, M. Negri, G. Pelagatti, and L. Sbattella. Object-Oriented Query Languages: the Notion and the Issues. *IEEE Transactions on Knowledge and Data Engineering*, 4:223–237, 1992.

[BP91] J. van den Bussche and J. Paredaens. The Expressive Power of Structured Values in Pure OODBs. In *Proc. of the Tenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 291–299, 1991.

[CC95] T. Coupaye and C. Collet. Denotational Semantics for an Active Rule Execution Model. In T. Sellis, editor, *Proc. Second International Workshop on Rules in Database Systems*, number 985 in Lecture Notes in Computer Science, pages 36–50, 1995.

[CCCR+89] F. Cacace, S. Ceri, S. Crespi-Reghizzi, L. Tanca, and R. Zicari. The Logres project: Integrating Object-Oriented Data Modelling with a Rule-Based Programming Paradigm. Technical Report TR 89-039, Dipartimento di Elettronica e Informazione, Politecnico di Milano, 1989.

[CCS94] C. Collet, T. Coupaye, and T. Svensen. Naos: Efficient and Modular Reactive Capabilities in an Object-Oriented Database System. In *Proc. Twentieth Int'l Conf. on Very Large Data Bases*, pages 132–143, 1994.

[CFP94] S. Ceri, P. Fraternali, and S. Paraboschi. Constraint Management in Chimera. *IEEE Data Engineering Bulletin, Special Issue on Database Constraint Management*, 17(2):4–8, June 1994.

[CGMR95] S. Castangia, G. Guerrini, D. Montesi, and G. Rodriguez. Design and Implementation for the Active Rule Language of Chimera. In N. Revell and A M. Tjoa, editors, *Proc. Sixth International Conference and Workshop on Database and Expert Systems Applications*, pages 45–54, 1995.

[CGT90] S. Ceri, G. Gottlob, and L. Tanca. *Logic Programming and Databases.* Springer-Verlag, Berlin, 1990.

[CW89] W. Chen and D. S. Warren. C-Logic of Complex Objects. In *Proc. of the Eighth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 369–378, 1989.

[CW91] S. Ceri and J. Widom. Deriving Production Rules for Incremental View Maintenance. In G. M. Lohman, A. Sernadas, and R. Camps, editors, *Proc. Seventeenth Int'l Conf. on Very Large Data Bases*, 1991.

[CW96] S. Ceri and J. Widom. *Active Database Systems - Triggers and Rules for Advanced Database Processing.* Morgan-Kaufmann, 1996. To Appear.

[D+90] O. Deux et al. The Story of 0_2. *IEEE Transactions on Knowledge and Data Engineering*, 2(1):91–108, 1990.

[DBC96] U. Dayal, A. Buchmann, and S. Chakravarthy. The HiPAC project. In S. Ceri and J. Widom, editors, *Active Database Systems*. Morgan-Kaufmann, 1996. To Appear.

[DDH92] K. Davis, G. Dong, and A. Heuer. Discussion Report: Object Migration and Classification. In U. Lipeck and B. Thalheim, editors, *Proc. Fourth International Workshop on Foundations of Models and Languages for Data and Objects - Modelling Database Dynamics*, Workshops in Computer Science, pages 223–227, 1992.

[DPG91] O. Diaz, N. Paton, and P. Gray. Rule Management in Object Oriented Databases: A Uniform Approach. In G. M. Lohman, A. Sernadas, and R. Camps, editors, *Proc. Seventeenth Int'l Conf. on Very Large Data Bases*, pages 317–326, 1991.

[F$^+$89] D. H. Fishman et al. Overview of the Iris DBMS. In W. Kim and F. H. Lochovsky, editors, *Object-Oriented Concepts, Databases, and Applications*, pages 219–250. Addison-Wesley, 1989.

[FPT92] P. Fraternali, S. Paraboschi, and L. Tanca. Automatic Rule Generation for Constraint Enforcement in Active Databases. In U. Lipeck and B. Thalheim, editors, *Proc. Fourth International Workshop on Foundations of Models and Languages for Data and Objects - Modelling Database Dynamics*, Workshops in Computer Science, pages 153–173, 1992.

[FT94] P. Fraternali and L. Tanca. A Structured Approach for the Definition of the Semantics of Active Databases. Technical Report 94-073, Dipartimento di Elettronica e Informazione, Politecnico di Milano, 1994. To appear in *ACM Transactions on Database Systems*.

[GBB94] G. Guerrini, E. Bertino, and R. Bal. A Formal Definition of the Chimera Object-Oriented Data Model. Technical Report IDEA.DE.2P.011.01, ESPRIT Project 6333, May 1994. Submitted for publication.

[GD92] S. Gatziu and K. Dittrich. SAMOS: an Active Object-Oriented Database System. *IEEE Data Engineering Bulletin, Special Issue on Active Databases*, 15(4):23–26, December 1992.

[GdBA94] P. Grefen, R. de By, and P. Apers. Integrity Control in Advanced Database Systems. *IEEE Data Engineering Bulletin, Special Issue on Database Constraint Management*, 17(2):9–13, June 1994.

[GJ91] N. Gehani and H. Jagadish. Ode as an Active Database: Constraints and Triggers. In *Proc. Seventeenth Int'l Conf. on Very Large Data Bases*, pages 327–336, 1991.

[GJ96] N. Gehani and H. V. Jagadish. Active Database Facilities in Ode. In S. Ceri and J. Widom, editors, *Active Database Systems*. Morgan-Kaufmann, 1996. To Appear.

[GJS92] N. Gehani, H. Jagadish, and O. Shmueli. Event Specification in Active Object-Oriented Databases. In *Proc. of the ACM SIGMOD Int'l Conf. on Management of Data*, pages 81–90, 1992.

[GKS91] G. Gottlob, G. Kappel, and M. Schrefl. Semantics of Object-Oriented Data Models - The Evolving Algebra Approach. In *Proc. Int'l Workshop on Information Systems*, Lecture Notes in Computer Science, pages 144–160, 1991.

[GLR90] S. Greco, N. Leone, and P. Rullo. COMPLEX: An Object-Oriented Logic Programming System. *IEEE Transactions on Knowledge and Data Engineering*, 4(4):344–359, August 1990.

[GM95] G. Guerrini and P. Marzano. Interference Queries on Hierarchical Terrain Models in a Deductive Object-Oriented Database Language. Technical Report DISI-TR-95-14, Dipartimento di Informatica e Scienze dell'Informazione, Università di Genova, 1995.

[HS91] A. Heuer and P. Sander. Preserving and Generating Objects in the Living in a Lattice Rule Language. In *Proc. Seventh IEEE Int'l Conf. on Data Engineering*, pages 562–569, 1991.

[K+89] W. Kim et al. Features of the ORION Object-Oriented Database System. In W. Kim and F. H. Lochovsky, editors, *Object-Oriented Concepts, Databasases, and Applications*, pages 251–282. Addison-Wesley, 1989.

[KBG89] W. Kim, E. Bertino, and J. Garza. Composite Objects Revisited. In J. Clifford, B. Lindsay, and D. Maier, editors, *Proc. of the ACM SIGMOD Int'l Conf. on Management of Data*, 1989.

[KKS92] M. Kifer, W. Kim, and Y. Sagiv. Querying Object-Oriented Databases. In M. Stonebraker, editor, *Proc. of the ACM SIGMOD Int'l Conf. on Management of Data*, pages 393–402, 1992.

[KL90] M. Kifer and G. Lausen. F-Logic: A Higher-Order Language for Reasoning about Objects, Inheritance, and Schema. In *Proc. of the ACM SIGMOD Int'l Conf. on Management of Data*, pages 134–146, 1990.

[KM91] A. Kemper and G. Moerkotte. A Formal Model for Object-Oriented Databases: The First Step. Technical report, University of Karlsruhe, 1991.

[KRSR94] G. Kappel, S. Rausch-Schott, and W. Retschitzegger. Beyond Coupling Modes: Implementing Active Concepts on Top of a Commercial ooDBMS. In E. Bertino and S. Urban, editors, *Proc. Int'l Symp. on Object-Oriented Methodologies and Systems*, number 858 in Lecture Notes in Computer Science, 1994.

[KU91] A. Karadimce and S. Urban. Diagnosis Anomalous Rule Behaviour in Databases with Integrity Maintenance. In *Proc. Third International Workshop on Foundations of Models and Languages for Data and Objects*, pages 77–102, 1991.

[KW89] M. Kifer and J. Wu. A Logic for Object-Oriented Logic Programming (Maier's O-Logic Revisited). In *Proc. of the Eighth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 379–393, 1989.

[LGA96] D. Lieuwen, N. Gehani, and R. Arlein. The Ode Active Database: Trigger Semantics and Implementation. In *Proc. Twelfth IEEE Int'l Conf. on Data Engineering*, 1996. To Appear.

[Liu92] L. Liu. *A Formal Approach to Structure, Algebra and Communication of Complex Objects*. PhD thesis, Katholieke Universiteit Brabant Tilburg, November 1992.

[LO91] Y. Lou and Z. M. Ozsoyoglu. LLO: An Object-Oriented Deductive Language with Methods and Methods Inheritance. In *Proc. of the ACM SIGMOD Int'l Conf. on Management of Data*, pages 198–207, 1991.

[LRV90] C. Lecluse, P. Richard, and F. Velez. 0_2, an Object-Oriented Data Model. In F. Bancilhon and P. Buneman, editors, *Advances in Database Programming Languages*, pages 257–276. Addison-Wesley, 1990.

[Mai86] D. Maier. A Logic for Objects. In *Proc. of the Workshop on Foundation of Deductive Database and Logic Programming*, pages 6–26, 1986.

[MD89] D. McCarthy and U. Dayal. The Architecture of an Active Data Base Management System. In *Proc. of the ACM SIGMOD Int'l Conf. on Management of Data*, pages 215–223, 1989.

[MP91] C. Medeiros and P. Pfeffer. Object Integrity Using Rules. In P. America, editor, *Proc. Fifth European Conference on Object-Oriented Programming*, number 512 in Lecture Notes in Computer Science, pages 219–230, 1991.

[Pet94] R. J. Peters. *TIGUKAT: A Uniform Behavioral Objectbase Managent System.* PhD thesis, Department of Computing Science, University of Alberta, April 1994.

[RS91] J. Richardson and P. Schwarz. Aspects: Extending Objects to Support Multiple, Independent Roles. In J. Clifford and R. King, editors, *Proc. of the ACM SIGMOD Int'l Conf. on Management of Data*, pages 298–307, 1991.

[Run92] E.A. Rundensteiner. A Methodology for Supporting Multiples Views in Object-Oriented Databases. In *Proc. Eighteenth Int'l Conf. on Very Large Data Bases*, pages 187–198, 1992.

[SdSAD94] C. Souza dos Santos, S. Abiteboul, and C. Delobel. Virtual Schemas and Bases. In M. Jarke, J. Bubenko, and K. Jeffery, editors, *Proc. Fourth Int'l Conf. on Extending Database Technology*, number 779 in Lecture Notes in Computer Science, pages 81–94, 1994.

[SJJN93] M. Staudt, M. Jarke, M. Jeusfeld, and H. Nissen. Query Classes. In S. Tsur, S. Ceri, and K. Tanaka, editors, *Proc. Third Int'l Conf. on Deductive and Object-Oriented Databases*, number 760 in Lecture Notes in Computer Science, pages 283–295, 1993.

[SLT90] M. Scholl, C. Laasch, and M. Tresch. Views in Object-Oriented Databases. In *Proc. Second International Workshop on Foundations of Models and Languages for Data and Objects*, pages 37–58, 1990.

[SLT91] M. Scholl, C. Laasch, and M. Tresch. Updatable Views in Object-Oriented Databases. In M. Kifer et al., editors, *Proc. Second Int'l Conf. on Deductive and Object-Oriented Databases*, number 566 in Lecture Notes in Computer Science, pages 189–207, 1991.

[SO90] D. Straube and M. Ozsu. Queries and Query Processing in Object-Oriented Database Systems. *ACM Transactions on Information Systems*, 8(4):387–430, October 1990.

[SS89] J. Schilling and P. Sweeney. Three Steps to Views: Extending the Object-Oriented Paradigm. In *Proc. Fourth Int'l Conf. on Object-Oriented Programming: Systems, Languages, and Applications*, pages 353–361, 1989.

[SS90] M. Scholl and H. Schek. A Relational Object Model. In S. Abiteboul and P.C. Kanellakis, editors, *Proc. Third Int'l Conf. on Database Theory*, number 470 in Lecture Notes in Computer Science, pages 89–105, 1990.

[ST93] K. D. Schewe and B. Thalheim. Fundamental Concepts of Object Oriented Databases. *Acta Cybernetica*, 11(1-2):49–83, 1993.

[STSW92] K.-D. Schewe, B. Thalheim, J. Schmidt, and I. Wetzel. Integrity Enforcement in Object-Oriented Databases. In U. Lipeck and B. Thalheim, editors, *Proc. Fourth International Workshop on Foundations of Models and Languages for Data and Objects - Modelling Database Dynamics*, Workshops in Computer Science, pages 174–195, 1992.

[Su91] J. Su. Dynamic Constraints and Object Migration. In G. M. Lohman, A. Sernadas, and R. Camps, editors, *Proc. Seventeenth Int'l Conf. on Very Large Data Bases*, pages 233–242, 1991.

[TCG+93] A. Tansel, J. Clifford, S. Gadia, S. Jajodia, A. Segev, and R. Snodgrass. *Temporal Databases: Theory, Design, and Implementation.* Database Systems and Applications Series. Benjamin/Cummings, 1993.

[UD90] S. Urban and L. Delcambre. Constraint Analysis: a Design Process for Specifying Operations on Objects. *IEEE Transactions on Knowledge and Data Engineering*, 2(4), December, 1990.

[Ull89] J.D. Ullman. *Priciples of Database and Knowledge-Base Systems.* Computer Science Press, 1989.

[VW91a] G. Vossen and K. U. Witt. FASTFOOD: A Formal Algebra over Sets and Tuples for the FOOD Object-Oriented Data Model. Technical report, Arbeitsgruppe Informatik, Justus Liebig Universitat, Giessen, August 1991. TR n. 9103.

[VW91b] G. Vossen and K. U. Witt. Objectbase Schemata and Objectbases in the FOOD Model. Technical report, Arbeitsgruppe Informatik, Justus Liebig Universitat, Giessen, June 1991. TR n. 9101.

[Zdo90] S. Zdonik. Object-Oriented Type Evolution. In F. Bancilhon and P. Buneman, editors, *Advances in Database Programming Languages*, pages 277–288. Addison-Wesley, 1990.

A Structure of the Thesis

The thesis to be developed could be structured in the following chapters:

1. **Introduction** motivating the work with respect to the research area;
2. **Related Work** surveying the pertinent literature, giving the starting point for the work;
3. **Reference Object-Oriented Data Model** introducing the formal definition of the reference object-oriented data model, and some kinds of model-specific integrity constraints;
4. **Query and Method Definition Languages** proposing rule-based query and method definition languages, and discussing their effectiveness;
5. **Triggers** introducing triggers in the model, and discussing how triggers can be employed to express model-specific integrity constraints;
6. **Views** proposing a view model for object-oriented databases, discussing view update propagation and view handling through triggers;
7. **Temporal Capabilities** addressing some issues arising from the introduction of the time dimension in an object-oriented data model with rules;
8. **Conclusions** summarizing the contribution of the thesis and outlining further developments.

B Work Plan

We can identify the following tasks to be performed for completing the thesis work, referring to the corresponding goals outlined in the research plan:

1. comparison of different dispatching techniques for objects with multiple most specific classes;
2. (a) investigation of the expressive power of the query language;
   (b) study of query subsumption;
   (c) precise definition of the method definition language;
   (d) study of behavior refinement and static analysis of transactions;
3. (a) integration of triggers with inheritance hierarchies;
   (b) definition of a semantic framework for triggers;
4. (a) identification of classes of integrity constraints peculiar to object-oriented data models;
   (b) expression of those integrity constraints through triggers;
   (c) analysis of the behavior of such triggers;
5. (a) completion of the proposed view mechanism;
   (b) propagation of view updates;
   (c) view handling through triggers;
6. investigation of architectural aspects;

7. (a) identification of issues in temporal object-oriented data models;
   (b) identification of issues in temporal rule languages.

We can identify the following dependencies:

- $2b \rightarrow 2d, 3a, 5a$;
- $2c \rightarrow 2d$;
- $3a \rightarrow 4b$;
- $4b \rightarrow 4c$;
- $3a \rightarrow 5c$.

The investigation of architectural aspects will be carried on in parallel with the definition of the corresponding features, while temporal capabilities will be investigated after all the other work will have been completed.