STRUCTURAL MODEL OF REAL-TIME DATABASES[*]

Nizar Idoudi, Claude Duvallet and Bruno Sadeg
LITIS, UFR des Sciences et Techniques, 25 rue Philippe Lebon BP 540, 76 058, Le Havre Cedex, FRANCE
{*Nizar.Idoudi, Claude.Duvallet, Bruno.Sadeg*}*@univ-lehavre.fr*

Rafik Bouaziz, Faiez Gargouri
MIRACL-ISIMS, BP 1030, 3018, Sfax, Tunisie
{*Raf.Bouaziz, Faiez.Gargouri*}*@fsegs.rnu.tn*

Keywords: RTDB, Real-Time Objects, Sensor Attributes, Derived Attributes, Evolutionary Stereotype, UML profile.

Abstract: A real-time database is a database in which both the data and the operations upon the data may have timing constraints. Our objective in this paper is to model easily real-time database structures using the Model-Driven Engineering paradigm, specially the UML2 language. For that purpose, we propose an UML2.0 profile, entitled UML-RTDB, which is based on UML2.O Profiles package, in order to cope with the specific need of designing real-time databases.

1 INTRODUCTION

A real-time database is a database in which both the data and the operations upon the data may have timing constraints (Ramamritham, 1993). Real-time database operations involve gathering data from the environment, processing the gathered information in the context of previously acquired information, and providing timely responses. The operations also involve processing not only archival data but also temporal data which loses its validity after a certain time duration. Timing constraints of data express how old data can still be considered valid. In general, temporal consistency of data has two aspects: *absolute* and *relative*. The absolute temporal consistency restricts the age of a single data item, while relative temporal consistency restricts the relative ages of a groups of data items with respect to each other (Ramamritham, 1993). Both the time semantics of the data and the response-time requirements imposed by the environment define the transaction timing constraints and may be expressed as either the *periods* or *deadlines*.

Two of the most widely-studied models for real-time databases are the relational and the object-oriented models. However, due to the nature of many real-time applications that must handle complex real-world objects with short deadlines, many researchers believe that the object-oriented model is more natural and powerful than the relational model (Kim, 1995). Several research projects on real-time databases have adopted the object-oriented model for building their prototype systems (Wolfe et al., 1997) (Stankovic et al., 1997). Our work in this paper is based on the *object-oriented database model*. Thus, a real-time database is a collection of objects which are used to model time-critical dynamic systems in the real world. Each object has some internal state which is protected by the object abstraction. The only way that objects can be accessed by transactions is to invoke the methods defined by objects.

UML is a general language for modeling object-oriented applications across a wide range of domains. Developing a truly adequate uniform modeling language in the face of these diverse domains seems an unsolvable quest and contrasts domain specific software engineering activities. Recently, many adaptations to the UML have been made to reflect a real-time domain's world. Thereby, several UML approaches were proposed to take into account the real-time system requirements such as *RT-UML* (Douglass, 2004), *UML-SDL* (ITU-T, 1999), and *ACCORD/UML* (Lanusse et al., 1999). In the UML standard, The basic concepts which are integrated are those of *RT-UML*, through the UML profile for

[*]This work is supported by the Haute-Normandie region **CPER** project "*Logistic Transport Network and Information Technics*".

Schedulability, **P**erformance, and **T**ime (denoted SPT profile) (OMG, 2005). However, UML constructs used by these approaches do not support real-time database requirements. Indeed, the design of real-time database must consider both temporal aspects of data and timing constraints of transactions (Stankovic et al., 1999). To the best of our knowledge, there is only one based UML proposal for real-time databases modeling (DiPippo and Ma, 2000). In their work, the authors have defined an UML package for specifying RTSORAC object, called *RT-Object*. However, the RT-Object package is based on the Extension Mechanisms package of UML1.3 which is a past standard.

The organisation of this paper is as follows. Section 2 describes our real-time object model. Section 3 details the UML-RTDB profile, which is a specialized variant of the UML2.0 in real-time database applications. This profile contains specialized versions of the metamodel elements, i.e. *Stereotypes*, defined in the UML2.0 metamodel that allow the design of class diagrams for real-time databases. In section 4, we conclude the paper and give some perspectives to our work.

2 REAL-TIME OBJECT MODEL

A real-time database is by definition a database system. It has queries, schemas, transactions, commit protocols, concurrency control support, and storage management (Stankovic et al., 1999). A real-time database models an external environment that changes continuously. It is designed to be kept in shared main memory for fast and predictable access (Ramamritham, 1993). The design of a real-time database has to take into account the management of all these components. That's why, in our work, we define a Real-time object in order to declare the time-constrained data, the time-constrained operations, the parallelism, and the concurrency property inherent to real-time databases. Real-time objects are real-time database entities. They represent dynamic entities of time-critical dynamic systems in the real world. As shown in the figure 1, each real-time object is made of four components: (i) a set of real-time attributes, (ii) a set of real-time methods, (iii) a mailbox, and (iv) a local controller.

Because of the dynamic nature of the real world, more than one transaction may send requests to the same real-time object. Concurrent execution of these transactions allows several methods to run concurrently within the same object. To handle this essential property of real-time database systems, we associate to each real-time object a local concurrency control

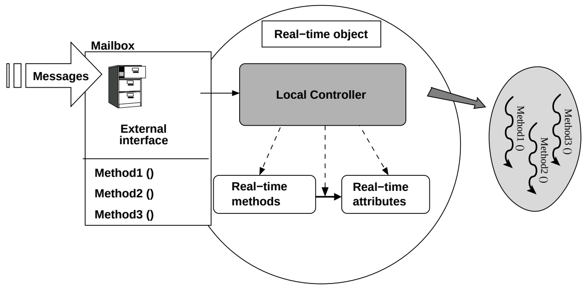

Figure 1: Schema of the Real-time object.

mechanism, named local controller, that manages the concurrent execution of its methods. Thus, the real-time object receives messages in its mailbox awaking its local controller that checks the timing constraint attached to messages and selects one message following a special scheduling algorithm. The local controller verifies the concurrency constraints with the already running methods of the real-time object. Then, it allocates a new thread to handle the message when possible. When a method terminates its execution, the corresponding thread is released and concurrency constraints are relaxed. If the service is periodic, the thread is not released and supports all periodic execution of the requested service. A real-time object is an extension of active object as usually defined in concurrent programming languages (Terrier et al., 1997), that encapsulates time-constrained data, time-constrained methods and concurrency control mechanisms.

2.1 Sensor and Derived attributes

Real-time data is divided into two types: *sensor data* and *derived data* (Ramamritham et al., 2004). Sensor data are the data issued from sensors. Derived data are the data calculated from sensor data. Our real-time data model is based on the model introduced in (Ramamritham, 1993) and we associate to this model the notion of *maximum data error* (MDE) introduced in (Amirijoo et al., 2006). Thus, a real-time data is modeled by d = (d_{value}, $d_{timestamp}$, d_{avi}, d_{mde}), where d_{value} represents the real world data value, $d_{timestamp}$ is the time at which the attribute's value was last updated, d_{avi} is the absolute validity interval and d_{mde} is the maximum amount of imprecision associated with the attribute's value.

We characterize the real-time object by a *real-time attribute model*, which is devided into two types of attributes: *sensor attributes* and *derived attributes*. Senor attributes are used to store a sensor data which must be periodically updated in order to closely reflect the real world state of the application environment. Derived attributes are used to store a derived data that has to be calculated from sensor attributes.

Name			
Current Value (CV)	TimeStamp (TS)	Validity Duration (VD)	Maximum Data Error (MDE)

Figure 2: Structure of the Real-time attribute.

As shown in the figure 2, each real-time attribute is characterized by <*N, CV, TS, VD, MDE*>. *N* is the name of the attribute. The second field, *CV*, is used to store the final attribute value captured by the last update correspondent method. This field is used by the system to determine logical integrity constraints of the attribute value. The third field, *TS*, is used to store the time at which the attribute's value was last updated (Ramamritham, 1993). Access to the timestamp of an attribute is necessary for determining temporal consistency of the attribute. The next field, *VD*, is used to store the *absolute validity interval* (denoted by *avi*) of the attribute value (Ramamritham, 1993). It represents the amount of time during which the attribute value is considered valid. This element is numeric and permits to determine, in association with *TS*, the *absolute consistency* of the attribute (Ramamritham, 1993). The last field, *MDE*, of a real-time attribute is used to memorize the *absolute maximum data error* tolerated on the attribute value (Amirijoo et al., 2006). This field determines the upper bound of deviation between the current data value and the updated value.

3 AN UML-RTDB PROFILE

There is a need to define an UML profile supporting real-time database requirements. In this section, we present an UML profile, entitled UML-RTDB, which is based on UML2.0 Profiles package and provides various stereotypes for *sensor attributes*, *derived attributes* and *real-time objects*.

3.1 Real-Time Attribute stereotype

The main goal of the UML-RTDB is to provide to the designers of real-time databases an UML extension (stereotypes) that supports both sensor and derived attributes features. A stereotype defines how an existing metaclass may be extended, and enables the use of platform or domain specific terminology or notation, in addition to the ones used for the extended metaclass (OMG, 2007). However, the standard stereotype can not express features of sensor and derived attributes because it does not allow the definition of structural and behavioral features (Debnath et al., 2003). Besides, the fields *timestamp*, *validity duration* and *maximum data error* represent structural features and should be expressed as ***Attributes***. Each atribute is also characterized by an update operation that ensures its freshness and must be modeled as an ***Operation***. For this reason, we base our work on the *Evolutionary Stereotype* extension mechanism of UML (Debnath et al., 2003). This extension mechanism allows the definition of new stereotypes with *structural* and *behavioral* characteristics.

Moreover, we base our proposal on the *Extension relationship* proposed in UML2.0 Profiles package (OMG, 2007). The *extension* is used to indicate that the properties of a metaclass are extended through a stereotype, and gives the ability to flexibly add (and later remove) stereotypes to (resp. from) classes. Thus, the *Stereotype::baseClass* attribute is replaced by an extension relationship to the *Class* metaclass. This former generalizes the *Stereotype* metaclass and contains a *name* attribute. In addition, the *Stereotype::icon* attribute is replaced by the *icon* role linking the *Stereotype* metaclass to *Image* metaclass.

Since the sensor attributes and derived attributes have the same structural characteristics (*Timestamp*, *Validity Duration*, *Maximum Data Error*) and behavioral characteristics (*Update operation*), we propose an abstract stereotype, called ≪*RealTimeAttribute*≫, in order to factorize these characteristics. So, instead of defining the structural and behavioral features for each stereotype aside, we define these features in a general manner within the ≪*RealTimeAttribute*≫ stereotype. This former is a realization of the Evolutionary stereotype that allows the statement of structural and behavioral features. Thereby, as shown in figure 3, the ≪*RealTimeAttribute*≫ stereotype is characterized by three structural and one behavioral characteristics. The first features are defined by ***TimeStamp***, ***Validity Duration***, and ***Maximum Data Error*** metaclasses, which specialize the *StructuralFeature* metaclass of UML metamodel. The last feature is defined by the ***Update*** metaclass, which specializes the *BehavioralFeature* metaclass.

The ***TimeStamp*** metaclass declares the timestamp of the attribute. It is characterized by two properties: *Type* and *Granularity*, where *Type* indicates the type of the Timestamp value, which is the *Time* type expressed in UML metamodel. The *Granularity* defines the granule of the Timestamp, which may be *"Minute"*, *"Second"*, etc.

The ***Validity Duration*** metaclass defines the valid time of the attribute value. It is characterized by two properties: *Type* and *Granularity*. The *Type* indi-

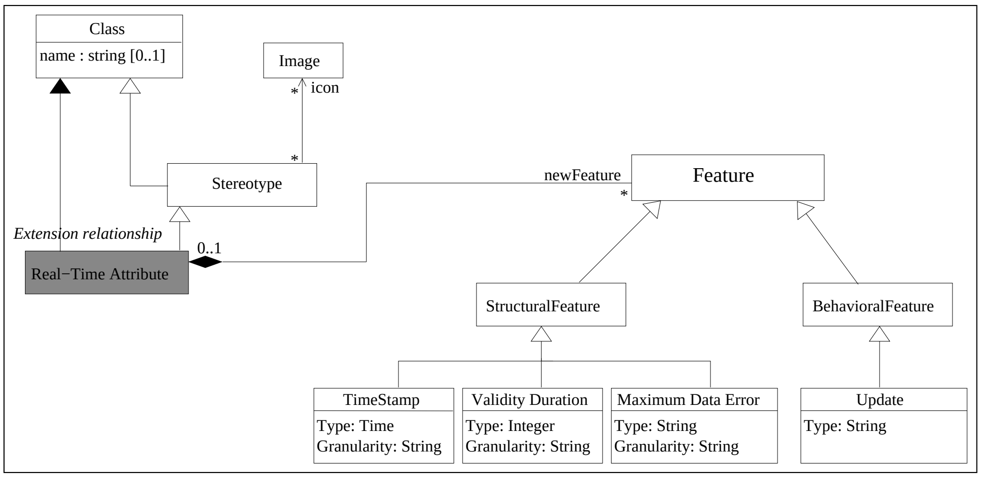

Figure 3: Abstract syntax of the ≪*RealTimeAttribute*≫ stereotype.

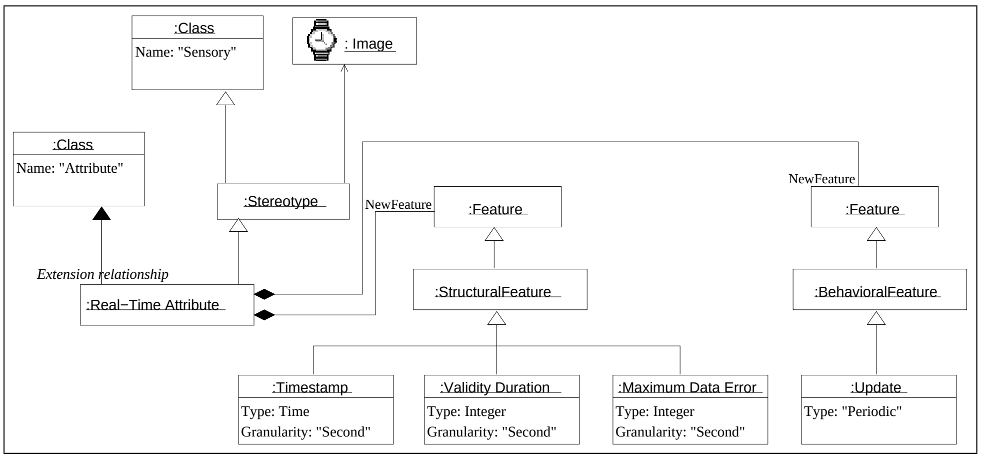

Figure 4: Abstract syntax of the ≪Sensory≫ stereotype.

cates the type of the Validity Duration value, which is typed *Integer*. The *Granularity* defines the granule of the Validity Duration, which may be *"Minute"*, *"Second"*, etc.

The ***Maximum Data Error*** metaclass defines the amount of error tolerated on an attribute value. It contains a *Type* property, which indicates the type of Maximum Data Error value. It has the same type as attribute value. A *Granularity* property defines the granule of the Maximum Data Error, which may be *"Minute"*, *"Second"*, etc.

The ***Update*** metaclass declares the operation which updates the *CurrentValue* and *TimeStamp* fields of the attribute. It contains a *Type* property that indicates the type of the operation periodicity. In our case, the operation periodicity depends on the type of the attribute. It is *"Periodic"* for a sensor attribute, and it is *"Sporadic"* for a derived attribute.

3.2 Sensory and Derived stereotypes

We define ≪*Sensory*≫ stereotype and ≪*Derived*≫ stereotype to declare respectively sensor attributes and derived attributes, in the structural model. Each stereotype presents a realization of the ≪*RealTimeAttribute*≫ stereotype according to the appropriate values of its properties, and an extension of the ***Attribute*** metaclass of UML metamodel. Figure 4 shows the abstract syntax of ≪*Sensory*≫ stereotype. Its structural features are declared by the metaclasses: *TimeStamp*, *Validity Duration*, and *Maximum Data Error*. In this work, we consider that time granularity is the *"Second"*. The designers of real-time databases can easily modify the values of stereotype properties according to the requirements of real-time applications. The only behavioral feature of the ≪*Sensory*≫ stereotype is defined by the

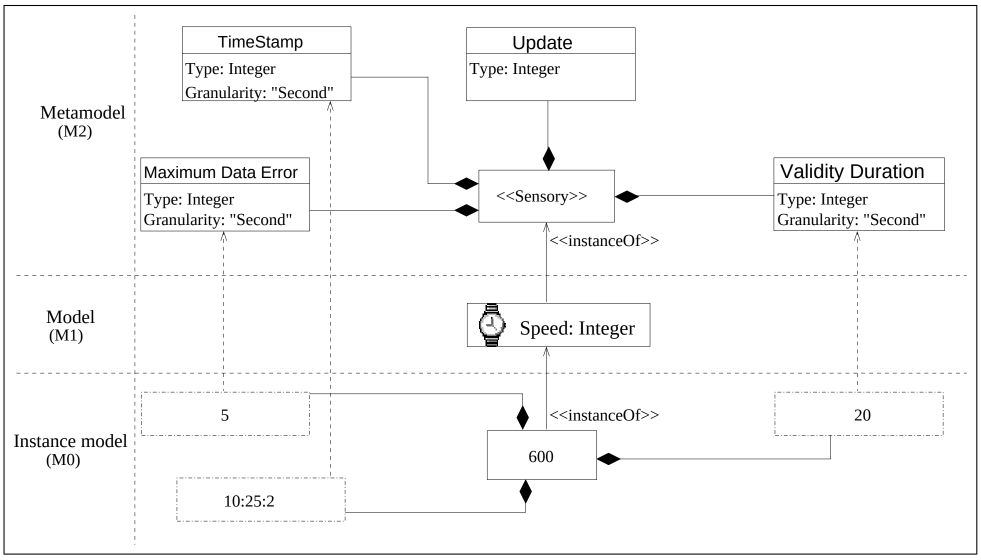

Figure 5: Instance Model of the ≪Sensory≫ stereotype.

Update metaclass. The ≪*Derived*≫ stereotype has the same abstract syntax as the ≪*Sensory*≫ stereotype. They have the same structural and behavioral characteristics. Besides, they extend the same meta-instance of the *Class* metaclass, i.e. *"Attribute"*, through the *Extension relationship*. However, these two stereotypes differ in their specific notations, their meta-instances of the *Class* metaclass which generalizes the *Stereotype* metaclass and specifies the *name* of the stereotype, and their operations periodicity. So, for the ≪*Sensory*≫ stereotype, the meta-instance name of the *Class* metaclass is *"Sensory"* (cf. figure 4). For the ≪*Derived*≫ stereotype, the meta-instance name is *"Derived"*. Besides, as shown in the figure 6, we have chosen a *"watch"* as an icon declaring a sensor attribute and a *"Calculator"* declaring a derived attribute. The operation periodicity of a sensor attribute is *"Periodic"* and it is *"Sporadic"* for a derived attribute.

Figure 5 shows an instance model of the sensor attribute *Speed*, according to three layers: metamodel, users model, and user objects (user data). It is characterized as follows:

≪*Name*: Speed, *CurrentValue*: 600, *TimeStamp*: 10:25:2, *ValidityDuration*: 20, *MaximumDataError*: 5≫.

3.3 Real-Time Object stereotype

A real-time database is a collection of real-time objects which are used to model a time-critical dynamic system in the real world. The design of a real-time database has to take into account the management of real-time object features. In our work, we define a ≪*RealTimeObject*≫ stereotype in order to declare the time-constrained data, the time-constrained operations, the parallelism, and the concurrency property inherent to real-time objects. The ≪*RealTimeObject*≫ stereotype is added to classes in order to specify that their instances will encapsulate real-time data, real-time operations, and a local concurrency control mechanism.

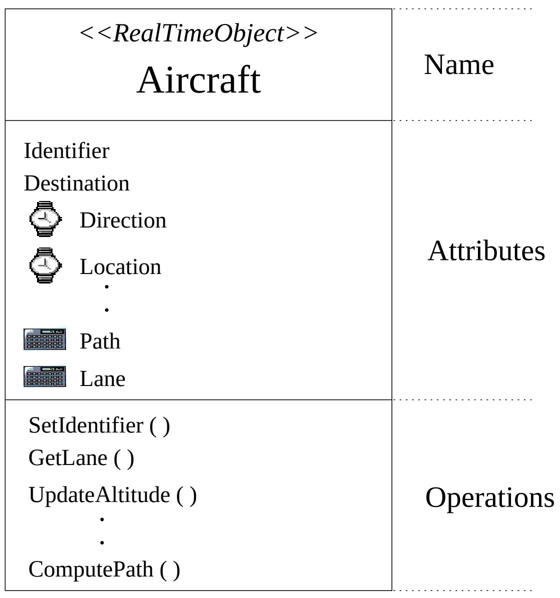

Figure 6: Aircraft real-time object class.

As an example, we consider an air-traffic control system (Locke, 2001), in which each aircraft in the airspace can be modeled as a real-time object. The designers of the system can include the UML-RTDB package when building their design in order to cope with the specific need of designing real-time databases. Figure 6 illustrates an *Aircraft* real-time object class. It encapsulates classical and real-time attributes (sensor and derived), and real-time operations.

4 CONCLUSION AND FUTURE WORK

In this work, we have presented how to represent real-time related properties in the object-oriented worlds, such as object-oriented databases and object-oriented CASE tools, such as UML2.0. For this purpose, we have proposed a real-time object model to include real-time aspects within databases from the viewpoint of object-oriented data model instead of traditional relational data model. We have also proposed an UML profile, called UML-RTDB, which is based on UML2.0 Profiles package and which provides various stereotypes for sensor attributes, derived attributes, and real-time objects.

In our future work, we will illustrate our proposal on an air traffic control system, as proposed in (Locke, 2001). We will also extend UML-RTDB with other stereotypes in order to express time-constrained operations, time-constrained associations, and time-constrained multiplicities.

REFERENCES

Amirijoo, M., Hansson, J., and Son, S. H. (2006). Specification and management of QoS in real-time databases supporting imprecise computations. *IEEE Transactions on Computers*, 55(3):304–319.

Debnath, N., Riesco, D., Maccio, A., Montejano, G., and Martellotto, P. (2003). Definition of a new kind of UML stereotype based on OMG meta-model. In *Proceedings of the ACS/IEEE Arab International Conference on Computer Systems and Applications: AICCSA'03*, Tunis, Tunisia.

DiPippo, L. C. and Ma, L. (2000). A UML package for specifying real-time objects. *Computer Standards and Interfaces*, 22(5):307–321.

Douglass, B. (2004). *Real Time UML, Third Edition : Advances in The UML for Real-Time Systems*. Pearson Education, Inc, 0-321-16076-2.

ITU-T (November 1999). Recommendation Z.109: languages for telecommunications applications - SDL combined with UML. International Telecommunication Union.

Kim, W. (1995). *Object-Oriented Database Systems: Promises, Reality, and Future. Modern Database Systems*, pages 255–280. Addison Wesly.

Lanusse, A., Gérard, S., and Terrier, F. (1999). Real-time modeling with UML: The ACCORD approach. In Bézivin, J. and Muller, P.-A., editors, *The Unified Modeling Language, UML'98 - Beyond the Notation. First International Workshop, Mulhouse, France, June 1998, Selected Papers*, volume 1618 of *LNCS*, pages 319–335. Springer.

Locke, D. (2001). Applications and system characteristics. In *Real-Time Database Systems: Architecture and Techniques*, pages 17–26. Kluwer Academic Publishers.

OMG (January 2005). "UML Profile for Schedulability, Performance and Time, v1.1", formal/2005-01-02.

OMG (November 2007). "Unified Modeling Language (UML), Infrastructure, v2.1.2", formal/2007-11-04.

Ramamritham, K. (1993). Real-Time Databases. *Journal of Distributed and Parallel Databases*, 1(2):199–226.

Ramamritham, K., Son, S., and DiPippo, L. (2004). Real-Time Databases and Data Services. *Real-Time Systems*, 28:179–215.

Stankovic, J., Son, S., and Hansson, J. (1999). Misconceptions about real-time databases. *IEEE Computer*, 32(6):29–36.

Stankovic, J., Son, S., and Liebeherr, J. (1997). Bee-Hive: Global Multimedia Database Support for Dependable, Real-Time Applications. In *Real-Time Database and Information Systems*, pages 409–422. Kluwer Academic Publishers.

Terrier, F., Lanusse, A., Bras, D., Roux, P., and Vanuxeem, P. (1997). Concurrent object for multitasking. In *L'objet*, volume 3, pages 179–196, Paris, France.

Wolfe, V. F., Prichard, J. J., Dipippo, L. C., and Black, J. (1997). *Real-Time Database System: Issues and Applications*, chapter The RTSORAC Real-Time-Object-Oriented DataBase Prototype, pages 279–301. Kluwer Academic Publishers.