Object–Oriented Database Systems: In Transition

François Bancilhon

Altaïr
B.P. 105
78153 Le Chesnay Cedex, France

Won Kim

UniSQL, Inc.
9390 Research Blvd.
Austin, Texas 78759

1. Introduction

An object–oriented database management system (OODB) is a database system which supports an object–oriented data model. Just as any traditional database system, it must provide disk management, data sharing, data integrity, security, and a query language. Further, in support of an object–oriented data model, it must manage complex objects with object identity, support objects that encapsulate data and behavior, structure objects in classes, and organize classes in a hierarchy.

Starting around 1983, the field of object–oriented databases has rapidly turned into a major area of research and become reasonably mature. It has attracted significant attention from the research community, the business community, and the user community. During the past 7 years much has been accomplished:

- *Existing database and programming language technologies have been reused and adapted*: Concurrency control, recovery techniques, storage techniques, distributed data management, query optimization and processing, type theory, and compilation techniques are examples of such technologies.

- *New technologies have been developed for OODBs*: Data models, query languages, indexing techniques, query optimization, schema modification, user interfaces, authorization mechanisms, performance metrics, client/server architecture, and in–memory object managers are examples of such new developments.

- *Considerable experimentation has been performed*: Complete and partial prototypes have been built, with IRIS [Fishman *et al.* 87], ORION [Kim *et al.* 90], and O2 [Deux *et al.* 90] being the major prototypes. These prototypes have been used as vehicles of experimentation with most of the design and implementation issues for OODBs; they have also been used to evaluate the performance and functionalities of the OODB technology.

- *A first consensus on the definition and core concepts concerning OODBs has been established* [Dittrich 86], [Kim 90] and [Atkinson *et al.* 89]; and standards committees are currently attempting to forge standards for various aspects of the OODB technology.

- *Some products have been introduced in the market*: Gemstone [Maier *et al.* 86], Vbase and its successor Ontos [Andrews 87], and the commercial version of ORION are examples of currently available products. Further, a few additional products are being readied for release in 1990 and 1991.

This high level of productivity in research and development is mainly due to the fact that OODB researchers and designers have reused and adapted the technology and experiences accumulated from the development of relational systems in the 1970s. We believe that the exploratory and experimentation phase for the OODB technology is rapidly winding down, and the technology is poised to shift gear into the next phase to take root in production environments to fulfill the needs that traditional database systems have failed to satisfy. Further, we believe that the transition from the current phase to the next will take at least a few years, requiring intensive further research and consolidation. The research must be focused on bridging and negotiating the gap between the OODB technology and traditional database technology; and the best results of the past and current research and development efforts, not only in the field of OODBs but also extensible databases, distributed databases, and user interfaces, must be consolidated in building the next generation of OODBs for production environments. On the basis of these observations, we outline in this paper five topics we regard as the most important and promising, in terms of relevance to OODB developers and/or technical challenges to OODB researchers.

- query model and query optimization
- user interfaces
- design methodologies and tools
- view mechanism
- performance benchmarks

We note that there are other, albeit less pressing in our opinion, topics of relevant research, including the following.

- type theory and its integration into database languages
- architectural issues in implementing object managers for both client/server and distributed machine configurations
- specification and enforcement of integrity constraints
- design, usage and maintenance of libraries of reusable objects

2. Query Model and Query Optimization

2.1 query model

Although some promising research has been done on queries (query model, query languages, and query optimization), various problems remain. The major one is the definition of a query model that would account for encapsulated objects:

- Unlike traditional query languages which only access data, query languages for OODBs must access data encapsulated in objects by invoking methods. A method may be system–defined or user–supplied; in either case, a method defines a far richer set of operators on data than those supported in traditional database systems.

- It may be desirable to extend the *closure property* of the relational query languages to OODBs. A query language with the closure property takes as input a schema (and a database) and generates as output another schema (and another database). An OODB schema usually consists of a set of classes, classes consist of objects which have a structure and a set of methods. The query languages proposed for OODBs thus far are in general able to map structured objects into other structured objects. However, the objects returned do not necessarily belong to any of the existing classes; that is, to save the result of a query seems to require the creation of some artificial new classes somewhere on the hierarchy of classes. Further, query languages may take methods as "input" but do not generate objects with new methods.

If we are to define a formal foundation for OODBs as an extension of the relational framework, it is highly desirable to define a query model with the closure property. The notion of a complete query language defined for relational query languages will be helpful to formalize the power of the query languages for OODBs. A complete query language with the closure property will also serve as the basis for a view mechanism for OODBs, as we will see in Section 5.

2.2 query optimization

Query optimization for OODBs, just as for relational databases, involves enumeration of logically correct scan orderings for the classes, generation of query–execution plans, cost estimation for each of the plans, and application of algorithms and heuristics to reduce the search space for query–execution plans. However, it is more complex than in relational systems because

1. data has a more complex structure,

2. queries involve method invocation and thus optimizing a query can mean optimizing an entire program, and

3. there is no object–oriented equivalent of the simple and concise relational algebra (and it may be impossible to define such an algebra).

The complex structure of data in OODBs simply adds to the computational overhead in generating query–execution plans, but does not appear to introduce fundamentally new and significant problems that designers of relational systems have not addressed. However, queries involving methods have no relational equivalent, and pose some serious problems. If a method is used as an operator in a query, conventional access methods, such as a B–tree index, are no longer useful in optimizing the query; as such, new access methods are necessary to expedite the evaluation of a query involving such an operator. Further, the selectivity of such a general operator must be defined to augment the selectivities defined for conventional operators in the query optimizer of conventional relational database systems.

3. User Interfaces

Despite the longstanding consensus on the importance of user interfaces for non–programming users of database systems, the area of user interfaces for database systems remains largely barren. Most of the work has been focused on the design of a specific user interface for a specific system, rather than on the development of a *generic user–interface technology*.

It is tempting to leave the problem to researchers in the user interface discipline: after all, the database community does not develop operating systems to support database systems, so why should it develop user–interface management systems (UIMS) specifically for database systems and users? We believe that the problems that OODBs pose are particular and challenging, and they will be best addressed by the database community.

First, a specific data model, a specific schema structure, and a specific query language all require a specific user interface. At the very least, one has to design such an interface using available user–interface technology and tools. User interfaces for OODBs may very well become *applications* for a UIMS.

Second, even if one is to design a specific user interface for a specific OODB, the nature of the OODB introduces some new challenges, beyond those addressed for traditional databases. In particular, user interfaces for OODBs must deal with

- complex objects and their representations on the screen, either by graphs, by embedded graphical structures or by indented text structures (object sharing also requires an adequate visual representation on the screen),
- multimedia objects,
- display and manipulation of very large objects on the screen, and
- active objects.

Interfaces to traditional databases only had to deal with "passive data" which simply needs to be displayed and edited. In an OODB, objects have associated methods which may be activated directly on the screen, objects may reference other objects, and objects may be cut and pasted on other objects.

Thus, we believe that one must use existing user interface technology and eventually develop a new one to build systems which display complex interactive objects and connect them to an application. These systems should display interactively very large objects on the screen, display multimedia objects, provide a clean separation between the application and the interface and allow users to design interactively the display they want for objects.

4. Design Methodology and Tools

There is currently a need for a design methodology for OODB applications. It is one of the consistent feedbacks users have been giving to OODB experimenters. The standard questions are "where and how do we start" and "where do we go next?"

We must also address some cultural problems. There is a set of available object–oriented software design methodologies; however, they take little or no account of databases. Further, database application program-

mers have used existing methodologies for the past decade, based on some formal theories, and sometimes with associated tools. It is not reasonable to simply expect that the programmers will discard all this technology and culture. We must therefore devise new design methodologies for OODB application by either generalizing the existing ones or by providing a migration path from the old to the new methodologies.

5. View Mechanism

Views, as provided by relational systems, form a simple and powerful mechanism for data representation and structuring. The three–level ANSI/SPARC architecture which uses the view mechanism is a useful architecture to design complex applications involving several designers. Current OODBs do not offer a general view mechanism. A few of them offer some degraded form of views, either through exports of schemas or through encapsulation, but no complete and simple mechanism is yet available.

One of the reasons was mentioned in Section 2: current query languages are neither closed nor complete.

Another reason is the added complexity the view–derivation relationship introduces to the schema; that is, the view–derivation relationship results in a directed graph of classes and views along which some aspects of the class or view definition may be inherited. This "view hierarchy" is in addition to the class hierarchy and nested attributes of classes.

Object identity also makes view support in OODBs more difficult than in traditional systems. Typically, view definitions are stored in databases; but this seems to be difficult to do in OODBs because of the need for the generation of new identifiers. For example, let us consider a database of people, with some of them being married. Assume the schema describes this by objects of the class PERSON with an attribute spouse of class PERSON. Suppose now we need to extract a view of this database with objects of the class COUPLE with attributes HUSBAND and WIFE of the class PERSON. Then, this view must contain new object identifiers for the COUPLE objects. One interesting question is whether these identifiers should be persistent or transient/virtual.

6. Performance Benchmarks

The question of the performance of systems is raised every time a new system or a new generation of systems is proposed. One of the important contributions the research community should make is to answer the question "what is a fast system?". This is usually done through benchmarks. The two traditional database benchmarks are the Wisconsin Benchmark which measures relational systems; and the ET1/TP1 benchmark which measures transaction systems.

A few benchmarks have already been proposed for OODBs: the SUN benchmark [Cattell 90], the Hypermodel benchmark [Anderson *et al.* 90], and the ACOB benchmark [DeWitt *et al.* 90]. All these benchmarks have been designed for engineering applications. Further, they only exercise complex object manipulation in OODBs.

We believe that a more general benchmark is necessary for OODBs. Such a benchmark should be designed for a much broader spectrum of application areas than just engineering applications, since applicability of OODBs is not confined to engineering applications. Further, the benchmark should be designed to exercise the large set of new database functions that OODB designers have been incorporating into the systems; examples are message passing, schema modification, version management, multimedia data management, etc. Finally, the benchmark should subsume relational benchmarks to allow a comparison across different OODB systems as well as a comparison between OODB systems and relational systems; this is an objective since there is a large degree of overlap in functions provided by OODB and relational systems, and an OODB data model can subsume the relational model of data.

7. Concluding Remarks

OODB research has been very active and productive during the past seven years, and the first batch of systems, incomplete as they may be, have reached the market in response to a growing demand for better solutions to data management problems than what the traditional systems offer. We believe that, although the OODB technology has significantly matured, there is much room for further research and experimentation

which should result in significantly more satisfactory next generation of systems. In this paper, we focused on five topics of research in OODBs which we regard as the most important and relevant in responding to the demands of the applications and which hold promise for exciting technical challenges.

References

[Andrews 87] T. Andrews and C. Harris. "Combining Language and Database Advances in an Object–Oriented Development Environment," " *Proceedings of the ACM Conf. on Object–Oriented Programming Systems, Languages, and Applications*, Orlando, Florida, Oct. 1987.

[Atkinson *et al.* 89] M. Atkinson, *et al.* "The Object–Oriented Database System Manifesto", *Proceedings of the 1st Intl. Conf. on Deductive and Object–Oriented Databases*, Kyoto, Japan, December 1989.

[Anderson *et al.* 90] L. Anderson, *et al.* "The HyperModel Benchmark," Proceedings of Conf. on Extending Data Base Technology, Venice, Italy, March 1990.

[Cattell 90] R. Cattell and J. Skeen. "Engineering Database Benchmark," SUN Microsystems Technical Report, Mountain View, Calif., April 1990.

[O. Deux *et al.* 87] O. Deux, *et al.* "The Story of O_2," *IEEE Trans. on Knowledge and Data Engineering,* March 1990.

[Carey *et al.* 88] M. Carey, D. DeWitt , and S. Vandenberg. "A Data Model and Query Language for EXODUS", *Proceedings of the 1988 ACM SIGMOD Conference,* Chicago, June 1988.

[DeWitt *et al.* 90] D. DeWitt, Ph. Futtersack, D. Maier, and F. Velez. "A Study of Three Alternative Workstation Server Architectures for Object–Oriented Database Systems," *Proceedings of the 16th Intl. Conf. on Very Large Data Bases*, Brisbane, Australia, August 1990.

[Dittrich 1986] K. R. Dittrich. "Object–Oriented Database System : The Notions and the Issues", in *Proceedings of the 1986 International Workshop on Object–Oriented Database Systems*, K. Dittrich and U. Dayal (eds.), IEEE Computer Society Press, 1986.

[Fishman *et al.* 87] D. Fishman *et al.* "Iris: An Object–Oriented Database Management System", *ACM Trans. on Office Information Systems,* vol. 5, no.1 January 87.

[Kim 90] W. Kim. "Object–Oriented Databases: Definition and Research Directions," *IEEE Trans. on Knowledge and Data Engineering,* March 1990.

[Kim *et al.* 90] W. Kim, J. Garza, N. Ballou, and D. Woelk. "Architecture of the ORION Next–Generation Database System," *IEEE Transaction on Knowledge and Data Engineering,* March 1990.

[Maier *et al.* 86] D. Maier, J. Stein, A. Otis, and A. Purdy. "Development of an Object–Oriented DBMS," *Proceedings of the ACM Conf. on Object–Oriented Programming Systems, Languages, and Applications*, Portland, Oregon, Oct. 1986.