W13. The First ECOOP Workshop on Object-Oriented Databases

Giovanna Guerrini[1], Isabella Merlo[1], Elena Ferrari[2], Gerti Kappel[3], and Adoracion de Miguel[4]

[1] DISI - Università di Genova, Italy
{guerrini,merloisa}@disi.unige.it
[2] DSI - Università di Milano, Italy
ferrarie@dsi.unimi.it
[3] Johannes Kepler University of Linz, Austria
gerti@ifs.uni-linz.ac.at
[4] Universidad Carlos III de Madrid, Spain
admiguel@inf.uc3m.es

Abstract. The goal of the *First ECOOP Workshop on Object-Oriented Databases* was to bring together researchers working in the field of object-oriented databases, to discuss the work which is going on. The aim of the workshop was twofold: to discuss the current status of research in the field and to critically evaluate object-oriented database systems in terms of their current usage, of their successes and limitations, and their potential for new applications. The workshop thus consisted of a number of presentations of reviewed papers and of discussions on the topics mentioned above.

1. Introduction

Three different directions can be devised aiming at integrating object-orientation and databases: the attempt of adding object-oriented features to the relational database standard SQL, which has lead to object-relational database systems [15, 16]; the extension of object-oriented programming languages to support persistent objects (e.g. PJama) [11] ; the attempt of definig a standard for "pure" object-oriented databases, namely, ODMG [3]. At the same time, some efforts have been made to extend the core object data model with richer modeling constructs, to better supporting integrity constraints, reactive capabilities, temporal data, schema evolution capabilities and to design efficient storage structures and implementation techniques for object databases. Object-oriented database systems have claimed of being more adequate than relational ones for handling non-traditional data, such as multimedia and semi-structured data, and for non-traditional applications, such as CADE/CASE, medical information systems, and for integrating heterogeneous database systems.

If you look at the programs of the last object-oriented and database conferences, however, there are very few papers on object-oriented databases. Thus, a first aim of the *First ECOOP Workshop on Object-Oriented Databases* has been to test whether there is still someone researching in that field. Fortunately, the answer has been positive. Together with these researchers, at the workshop, we aimed at discussing whether there is still anything to research and at critically evaluating the current state of the field. Are we satisfied about the reached results? Have all the problems of object-oriented databases been solved, or simply isn't there anyone interested in solving them, since relational DBMSs would never be overcome? Is the future in object-relational DBMSs and in persistent object-oriented languages? Does the web and web-related data open new perspective for object-oriented databases?

Thus, the aim of the workshop was to discuss the research going on in the object-oriented database field, critically evaluating object-oriented database systems in terms of their current usage, of their successes and limitations, and their potential for new applications. The discussion have covered all the aspects of object-oriented databases, ranging from data modeling issues, to implementation issues and practical evaluations of OODBMSs.

Submission of papers describing mature results or on-going work were invited for all aspects of object-oriented databases, including, but non limited to:

- data modeling concepts,
- performance, storage structures, query processing and optimization,
- experiences in using OODBMSs,
- object-oriented database standards,
- object-relational DBMSs,
- persistent object-oriented languages,
- interoperability and heterogeneity,
- active object-oriented databases,
- spatial and temporal aspects,
- applications in handling multimedia and semi-structured data.

A program committee of 14 people active in the field of object-oriented databases was assembled during the workshop preparation. The program committee members are experts in several aspects concerning object-oriented databases, ranging from theoretical foundations of object-oriented databases to object-oriented system implementations. The program committee members and their affiliations are listed below.

Suad Alagiċ	Wichita State University, USA
Elisa Bertino	University of Milano, Italy
Mary Fernandez	AT&T Labs Research, USA
Giorgio Ghelli	University of Pisa, Italy
Guido Moerkotte	University of Mannheim, Germany
Tamer Ozsu	University of Alberta, Canada
Isidro Ramos	Polytechnic University of Valencia, Spain
Awais Rashid	Lancaster University, UK
Elke Rundensteiner	Worcester Polytechnic Institute, USA
Gunter Saake	Magdeburg University, Germany
Markku Sakkinen	University of Jyväskylä, Finland
Marc Scholl	University of Konstanz, Germany
Roel Wieringa	University of Twente, The Netherlands
Roberto Zicari	Johann Wolfgang Goethe University, Germany

The workshop has been advertised inside both the object-oriented and the database communities, and many research groups active in object-oriented databases were directly contacted to encourage them to participate at the forum.

A total of 28 papers were submitted for presentation at the workshop. Submissions were attracted from 15 different countries giving a truly international flavour to the workshop. All the submitted papers were thoroughly reviewed by the programme committee and finally the organizing committee carefully selected the 10 papers to be presented at the workshop. Most of the submitted papers were of very good quality and the number of accepted papers has been limited by time restrictions.

The accepted papers were grouped into five main categories: evaluation of commercial systems, persistent object-oriented programming languages, query languages and heterogenous databases, extension of object-oriented databases with temporal features, and, finally, consistency and verification. Each accepted paper has been presented by one of the authors. Each presentation has been devoted 30 minutes: about 20 minutes for the paper presentation and ten minutes for questions and discussion. We have decided to devote at least ten minutes for questions and discussion to encourage the interaction among who was presenting the paper and other workshop participants, and we believe we have achieved good results: some discussions have been continued during lunch time! Informal proceedings were produced for the workshop and they have been given to the participant at the beginning of the workshop.

23 people participated at the workshop and most of them actively participated at the final discussion, giving a significant contribution to the workshop. Most of the participants came from the academy, though some of them came from the industry. The list of participants can be found at the end of the report.

The report is organized as follows. Section 2 describes the session dealing with the evaluation of commercial object-oriented database systems. Section 3 is devoted to the session on persistent object languages and systems. Section 4 describes the session devoted to query languages and to the management of heterogeneous information. Section 5 reports on the session discussing work aiming at extending object-oriented databases with temporal features. Section 6 is devoted to the session dealing with consistency and verification in object-oriented databases. For all those sessions, we present the abstracts of the works presented in the session, and briefly report on the discussion on all of them. Section 7 reports on the last session of the workshop, which was entirely dedicated to discussion. Finally, Section 8 concludes the workshop report.

2. Evaluation of Object-oriented Database Systems

In this session two very interesting papers giving an evaluation of object-oriented database systems from a "user" point of view were presented. The first one focuses on the requirements of organizations that have to deal with really very large databases, whereas the second one is centered around the capabilities offered by various systems for data and schema evolution.

2.1. Object Databases and Petabyte Storage - Dreams or Reality? (Dirk Düllmann, Jamie Shiers)

The European Laboratory for Particle Physics (CERN) is located on the border between France and Switzerland, just outside Geneva. Much of the on-going activities of the laboratory are focused on a new accelerator, the Large Hadron Collider (LHC), that is currently under construction. Scheduled to enter operation in 2005, experiments at this facility will generate enormous amounts of data. Over an estimated 20 year running period, some 100PB - 10^{17} bytes - of data will be acquired at rates ranging from 100MB/s to 1.5GB/s. A number of research and development projects have been initiated to find solutions to the many challenges that the LHC will pose. Among these, the RD45 project has focused on the problems of providing persistent storage to these vast quantities of data. Starting in 1995, this project has focused exclusively on object-oriented solutions and object database management systems in particular. In the paper [6], presented by Jamie Shiers, the authors describe the criteria by which the object-oriented technology have been choosen, issues related to product selection, and their experience in using OODBMSs in production. Finally, the risks involved with the current strategy and outline future directions for the project are discussed.

The discussion following the paper presentation was very animated. Both the system selected for the project (Objectivity/DB) and those examined as candidates were discussed, with strong emphasis on the architectural issues arosen in the presentation. The presentation was very interesting since it really evaluated object-oriented database systems in terms of their potential pratical usage in a very relevant application. Another important contribution of the presentation was indeed to recall to all the participants from the database community what *very large databases* mean.

2.2. Evaluation for Evolution: How Well Commercial Systems Do (Awais Rashid, Peter Sawyer)

The conceptual structure of an object-oriented database application may not remain constant and may vary to a large extent. The need for these variations (evolution) arises due to a variety of reasons, e.g. to correct mistakes in the database design or to reflect changes in the structure of the real world artefacts modeled in the database. Therefore, like any other database application object database applications are subject to evolution. However, in object-oriented databases, support for evolution is a critical requirement since it is a characteristic of complex applications for which they provide inherent support. These applications require dynamic modifications to both the data residing within the database and the way the data has been modelled i.e. both the objects residing within the database and the schema of the database are subject to change. Furthermore, there is a requirement to keep track of the change in case it needs to be reverted. The paper [13], presented by Awais Rashid, discusses the evolution facilities offered by some of the existing systems. The authors first provide an overview of the ODMG 2.0 standard from an evolution viewpoint. Then, they describe the extension to the database evolution features specified by the ODMG 2.0 standard. Using this extension as a basis, they form evaluation criteria, which are employed to assess the various evolution facilities offered by four commercially available object database management systems: POET, Versant, O2 and Jasmine.

Also in this case the presentation of the paper has been followed by an extensive discussion, and also in this case the choice of the systems considered in the comparison has been debated and motivated. The choice of the evolution primitives considered was also discussed, and the feeling that they should be driven by experiments on real projects rather than from the literature emerged. The other main issues arisen in the discussion concern typing problems caused by schema evolution, that is, whether the notions of schema evolution and of type safety can be conciled, and the potential offered by reflection to achieve type safety in an evolving environment.

3. Persistent Object-oriented Languages

In this session two papers dealing with persistent object systems were presented. The first one deals with the problem of schema evolution in the context of the PJama persistent object language, whereas the second one describes how persistence has been added to the BETA object-oriented programming language in the Mjølner system.

3.1. Evolutionary Data Conversion in the PJama Persistent Language (Misha Dmitriev, Malcom P. Atkinson)

PJama is an experimental persistent programming system for the Java programming language. It has much in common with modern object-oriented database systems used together with Java. Therefore, the problem of schema and data evolution is really important for PJama and will be even more important if it becomes a product.

The approach to and the implementation of schema (class) evolution in PJama was described in [5]. This paper [4], presented by Misha Dmitriev, deals with persistent object conversion (data evolution) facilities which have been recently developed. These facilities include default (automatic) and custom conversion, the latter meaning that the programmer can explicitly encode in Java the transformations for the objects that make these objects match new declarations of the respective classes. If custom conversion is needed, the programmer has a choice between *bulk* and *fully* controlled conversion. In the first case, a conversion method (function) should be written for each evolved class. The system then scans the persistent store, applying a suitable method for every evolving object that it finds. In case of fully controlled conversion, a complete program should be written, that can transform individual objects, as well as any larger data structures in arbitrary order. During any kind of conversion the “old” object graph remains unchanged, that is, substitute objects are not directly reachable from it, making their own, separate world. The paper shows that this is crucial for comprehensible semantics of evolution code. The author believes that the present set of facilities is complete, i.e. it is enough to convert any persistent store instead of rebuilding it from scratch. However, the question of whether this can be generally proven is raised.

The discussion following the presentation covered three main issues. First, the comparison with lazy and eager data conversion mechanisms offered by some OODBMSs, in particular by Objectivity/DB. Then, the fact that while the inspiring principle of PJama was to extend the Java programming language with persistence without modifying it, schema evolution capabilities lead to the violation of the Java type system rules. Again, the issue of schema evolution versus type safety has raised. Finally, the more apropriateness of lazy data conversion mechanisms to a transactional context where "stopping" the computation is unaccettable was pointed out.

3.2. Transparent, Scalable, Efficient OO-Persistence (Stephan Korsholm)

Doing garbage collection and handling persistent objects have much in common. Especially the marking phase of a mark-sweep garbage collector, during which the transitive closure of some set of garbage collection roots is traversed and marked. This phase of the garbage collector does in many respects perform the same kind of work as when the transitive closure of some set of persistent roots is serialized and exported from memory onto secondary storage. This work [10], developed within the Desartee project (Esprit LTR project no 31870), describes how efficient, scalable persistence has been transparently added to the Mjølner System (MS) by changing the generational garbage collector to identify and remove persistent objects from memory onto persistent storage, while hardware support has been utilized to load persistent objects on demand. Transparency has been achieved both at the user and at the machine level, while scalability is intended both with respect to time and to space. MS supports development of programs written in BETA, which are statically typed, garbage collected and compiled into binary machine code. The changes introduced to support persistence have not affected the size, layout or handling of non-persistent objects.

The discussion following the presentation pointed out that, according to this approach, objects are loaded when they must be used and are saved when garbage collecting, thus the transactional notions of commit and rollback are missing. The possibility of integrating transactional principles in the approach by forcing garbage collection upon commmit has been discussed. Moreover, the applicability of the presented techniques to Java without modifications to the Java Virtual Machine was also debated. Finally, the possibility of extending the approach with distribution and concurrency features was discussed.

4. Query Languages and Hetereogeneous Databases

This session has been devoted to query languages and interoperability. The first paper describes a multi-database system exploiting the object-oriented technology, whereas the second one describes a visual query language based on OQL.

4.1. Distributed Mediation using a Light-Weight OODBMS (Vanja Josifovski, Tore Risch)

Tore Risch presented this work [9] in which an overview is given of a light-weight, object-oriented, multi-database system, named Amos II. Object-oriented multi-database queries and views can be defined where external data sources of different kinds are translated through Amos II and integrated through its object-oriented mediation primitives. Through its multi-database facilities many distributed Amos II systems can interoperate. Since most data reside in the data sources, and to achieve good performance, the system is designed as a main-memory DBMS having a storage manager, query optimizer, transactions, client-server interface, etc. The Amos II data manager is optimized for main-memory and is extensible so that new data types and query operators can be added or implemented in some external programming language. Such extensibility is essential for data integration.

The presentation of the paper provided several hints for discussion. A first issue concerned the various levels and granularities of optimization that can be exploited in such a system. Then the applicability of that approach to applications requiring the storage of huge amounts of data was discussed, pointing out that a possible mean to reduce the amount of stored data to a reasonable size is to distribute them into different databases. Finally, problems related to method dispatch and its efficiency in this context were debated.

4.2. VOODOO: A Visual Object-Oriented Database Language for ODMG OQL (Leonidas Fegaras)

Query and data visualization are key components of many commercial database systems, such as Microsoft Access and Paradox. Object-oriented databases offer excellent opportunities for visual data browsing because they provide a direct and natural mapping of real-word objects and relationships into equivalent constructs. Even though there is already a sizable number of graphical user interfaces for object browsing, which include commercial products, such as O2Look and O2Tools of O2, there is a little work on visual query formulation in OODBs. Some researchers believe that, since OODB queries can be arbitrarily complex, this complexity must be reflected in the query formulation tool itself. In this work [8], the author shows that this is not necessarely true. According to the author point of view, if a query language is well-designed and uniform, as it is the case of ODMG OQL [3], it would require very few visual constructs for query formulation, since it would map similar features into similar visual constructs.

This paper presents VOODOO (which stands for a Visual Object-Oriented Database language for ODMG OQL) which is a simple and effective visual language to express ODMG OQL queries. The language is expressive enough to allow most types of query nesting, aggregation, universal and existential quantifications, group-by, and sorting, and at the same time is uniform and very simple to learn and use. The visual language is strongly typed in the sense that the queries constructed are always type-correct. In addition, there is sufficient type information displayed by the system that guides every stage of the query construction. The main difference between this language and other related visual query languages is that only one generic visual construct, called a *template*, is used instead of inventing a new one for each OQL syntactic feature.

The main issue arisen in the discussion on the paper concerned the extensibility of the visual query language to queries containing method invocations.

5. Temporal Object-oriented Databases

The extension of object-oriented databases with temporal features is addressed in the papers presented in this session. In particular, the first one proposes a flexible versioning mechanism allowing one to keep different portions of the object data over time. The second one, by contrast, deals with the problem of querying temporal document databases.

5.1. Outdating Outdated Objects (Holger Riedel)

Nowadays a lot of information is collected by many applications. Although storing mass data causes no problems, the processing of interesting queries poses many restrictions. Especially collecting temporal data leads to increasing costs for updates and queries, while the worth of such information shrinks quite rapidly. Working with such outdated data is insufficiently supported in current research and products. Within temporal databases, the concept of vacuuming was proposed to destroy a certain amount of historical data of a temporal relation. An even simpler solution is used in data warehouses, where the information which might be useful in the future is selected when transferring data from the operational database to the data warehouse.

In this paper [14] the author proposes a flexible temporal versioning schema for object-oriented databases which supports controlled restructuring of the history of objects. Therefore, arbitrary parts can be marked for deletion according to a temporal condition. These parts are put into different versions organized in a linear order. Thus, each object of such a versioned class walks through these versions in a fixed way. Although this leads to versioning on the object-level, arbitrary OQL queries including additional temporal constructs can be used, either ignoring or explicitly using the versions of a specific class. In order to get type-correct query results, specific null values are provided in a promising way. The author shows how this approach can be implemented by the use of lazy and eager evaluation schemes for version migration, flexible object-oriented storage structures, and enhanced query transformers and optimizers.

The relationships between versioning and the inheritance hierarchy, that is, the impact of versioning a class on its subclasses and superclasses were discussed after the presentation.

5.2. Retrieval of Information from Temporal Document Databases (María José Aramburu Cabo, Rafael Berlanga Llavori)

Current technology allows us to create large databases of documents from where they can be retrieved in several ways. However, the resulting systems do not offer proper tools for retrieving historical information from the stored documents by considering their temporal properties. Starting from the TOODOR object-oriented database model for representing up-to-date documents, the main proposal of this paper [1], presented by María José Aramburu Cabo, is a family of new document retrieval operators called *chronicles*. The temporal object-oriented data model of TOODOR supports two different temporal features of documents: publication and event times. The publication time represents the evolution of document classes and the time instants at which documents are published. The event time of a document is defined as the temporal projection of its contents, purpose or structure. By means of event times, chronicles can be used to group retrieved documents according to several predefined temporal patterns. These object operators are able to find out documents that refer to the same reported event, which is described by the user through an information retrieval condition, some time granularity and a temporal pattern. Among other things, chronicles are useful to determine the evolution of events, to search for certain temporal patterns, and to find out cause-effect relationships among documents.

During the presentation of the paper, a number of interesting issues have arisen and have been discussed. In particular, the differences between the event and publication times supported by the model and the classical notions of valid and transaction times of temporal database systems have been discussed. The motivations for providing an ad-hoc model and language for temporal document databases and the reasons of inadequacy of available models were also debated.

6. Consistency and Verification

This session deals with consistency and verification in the context of object-oriented database schemas. Two papers were presented in this session, the first one dealing with integrity constraint specification and consistency management, and the second one devoted to verification of schema requirements for an advanced transaction model.

6.1. Consistency Management in Object-Oriented Databases (Hussien Oakasha, Stefan Conrad, Gunter Saake)

The paper [12], presented by Gunter Saake, illustrates concepts and ideas underlying an approach for consistency management in object-oriented databases. In this approach constraints are structured as first class citizens and stored in a meta-database called constraint catalog. The structure of a constraint is an aggregation of two parts. The first part is called the *kernel* which is an object that is sharable among interrelated objects that are subject to the same constraint. The second part is called the *shell* which is an object being sharable among objects of the same class. When an object is created, constraints of this object are retrieved from the constraint catalog and relationships between these constraints and the object are established. The structure of constraints has several features that enhance consistency management in OODBMSs which do not exist in conventional approaches in a satisfactory way. These features are:

- monitoring object consistency at different levels of update granularity,
- integrity independence,
- efficiency of constraint maintenance,
- controlling inconsistent objects,
- enabling and disabling of constraints globally to all objects of database as well as locally to individual objects, and
- the possibility of declaring constraints on individual objects.

All these features are provided by means of basic notions of object-oriented data models.

The presentation of the paper stimulated discussion on several issues. Among the discussed topics, we mention criteria to establish how and when constraints should be checked in a context where data are accessed through programs whose execution may be very long. In addition, the possibility of exploiting reflection techniques to gain the chance of performing type checks over constraints was debated (such checks are impossible if constraints are codified as strings).

6.2. Compensation methods to support generic graph editing: A case study in automated verification of schema requirements for an advanced transaction model (Susan Even, David Spelt)

Compensation plays an important role in advanced transaction models, cooperative work, and workflow systems. However, compensation operations are often simply written as a^{-1} in transaction model literature. This notation ignores any operation parameters, results, and side effects. A schema designer intending to use an advanced transaction model is expected (required) to write correct method code. However, in the days of cut-and-paste, this is much easier said than done. The authors demonstrate in this paper [7], presented by Susan Even, the feasibility of using an off-the-shelf theorem prover (also called a proof assistant) to perform *automated* verification of compensation requirements for an OODB schema. In addition, they discuss the results of a case study in verification for a particular advanced transaction model that supports cooperative applications. The case study is based on an OODB schema that provides generic graph editing functionality for the creation, insertion, and manipulation of nodes and links.

Most of the discussion following the presentation focused on performance result of the developed tool, on the applicability of the approach to verify other properties about methods, and to different method definition languages.

7. Panel Discussion

In order not to start from scratch, we started the discussion with some comments on the *Asilomar Report* [2]. The Asilomar Report is a very interesting document published in 1998 on SIGMOD Record. On August 1998, a group of 16 among the most famous and important database system researchers from academy, industry, and government met at Asilomar (California), to assess the database system research agenda for the next decade. The goal of this meeting was to discuss the current database system research agenda and, if appropriate, to report their recomendations. The Asilomar Report summarizes the results of that meeting. Several interesting results are presented in the document which makes one thinking about the status of the research in computer science, not only in databases, nowadays. What is surprising is that object-oriented databases are not even mentioned in [2].

This fact, together with the small number of papers on object-oriented databases presented in international database conferences, makes one thinking that the research in object-oriented databases is "almost" finished. We came up with the conclusion that this is completely false. To testify that there is the success of the workshop, that, at its first edition, received a number of papers comparable to the one of a small conference. On this conclusion there has been a total agreement of the audience, perhaps even because that was the right audience.

It is, however, undoubted that the object-oriented database community is going through a frustration phase, mainly because of market reasons. That is, object-oriented DBMSs represent somehow a mature technology (most systems came into the market at the beginning of the 90s) but they are still not obtaining much commercial success. A proof of that can also be found in the fact that, though the workshop solicited the participation of people from the industry experiencing object-oriented DBMSs in real applications, the industrial representation at the workshop was really low (the only non-academic users reporting experiences of using OODBMSs were the CERN group). Thus, the general impression could be that of no industrial interest in OODBMSs, that is, no market for them.

This is however surprising, since if we take a look of the industrial market in computer science the object-oriented paradigm has proved itself successful in many respects. First, we would like to mention the success of the Java object-oriented programming language, which is the leading programming language on the Web. Moreover, we can mention UML, the new unified modeling language in software engeeniring, which is object-oriented, the success of the CORBA (Common Object Request Broker Architecture) architecture in distributed systems, and so on. By contrast, if we take a look of the database industrial market we notice that the leading systems are still totally based on the relational model. Recently, some efforts have been made by most successful relational DBMS producers to extend their systems with object-oriented features. The last systems are thus *object-relational*, that is, they support the possibility to define abstract data types as values of attributes and to refer to tuples via object identifiers. The relation, however, still remains the main construct for organizing data. The extent to which the object-oriented paradigm has been incorporated in object-relational (also known as *universal*) database systems is thus not fully satisfactory. The people at the workshop were all in agreement on the fact that there has not been a strong financial support by the industrial world for researching and improving object-oriented database systems. A reason for that can perhaps be that it is very hard to change ones mind when this is used to something for a long time, as relational databases, even though there are several advantages. Another reason can be found in the presence of the huge number of legacy systems based on relational databases which are still in use. We were not claiming, of course, that object-oriented DBMSs would be better than relational ones in an absolute way, rather than they can offer several advantages at least with respect to certain kinds of applications. Thus, there was the consensum of the workshop participants on the fact that the object-relational and the object-oriented database markets could coexist, simply because they fulfill different applicative requirements.

An important negative point of object-oriented DBMSs in their competion with object-relational DBMSs is surely represented by their weakness with respect to traditional database features. ObjectStore, for instance, is one of the most successful object-oriented DBMSs from the market point of view, and it misses many DBMS features: it has no declarative query language, no constraints, no complex transaction support, no security mechanism. Thus, a cause of the problems of object-oriented DBMSs can be to call them database management systems when they are (still) not full-fledged database management systems. ObjectStore itself can perhaps be better characterized as an *object repository* rather than a DBMS. That is the reason why persistent object languages could be more successful, offering some solution to the "static" market of object-oriented databases. If object-oriented programming languages are the programming languages of the future, the database community has to deal with managing data "inside" this languages. It is well known that embedded SQL suffers of several problems that could be avoided extending the used language with some facilities. Thus, the first step in this direction is the addition of persistent capabilities to object-oriented programming language. The model of persistence that thay should offer should be orthogonal, that is objects of any type may be persistent or transient. A starting point in the development of the Mjølner System [10], for instance, was that the border between the application and the

DBMS, which was sharp in the relational context, should not be stressed in the object-oriented context. Thus, their aim was to provide some functionalities, not to build a complete DBMS. The issue then arises of whether extending object-oriented programming languages with persistence features is enough. Database systems offer several features that completely lack in object-oriented programming languages, such as efficient set-oriented data manipulation, query languages, constraint specification, active and temporal features, index management, and so on.

In addition to architectural aspects such as indexes, clustering, transaction support, access control, a key DBMS feature missing in persistent object languages is related to declarativity: declarative query languages working on collections of objects, and integrity constraints. For instance, the PJama system, which extends Java, lacks all these features, since Java has not been conceived as a database programming language. Similarly, most OODBMSs, also among those which adhere to the ODMG standard, are actually not providing any declarative query language; only few of them implemented a subset of OQL. Thus, the issue was debated of whether switching to the object-oriented context would necessarily mean to impose procedural programming to database users. The general feeling was against this conclusion.

Finally, taking advantage of the presence of Jamie Shiers, reviewer member of ODMG, and of Suad Alagic, who is also very close to the ODMG group, and who have participated to many ODMG meetings, also the current status of the work of the group and of the standard has been discussed. In spite of the fact that ODMG 3.0 is expected to be released at the end of this year, the work of the group seems somehow frozen, and in particular the C++ group (that is, the group working on the C++ binding) has almost stopped its activity, and this can be seen as a witness of the Java “hegemony”. The name of the group is to be changed from Object Database Management Group to Object Data Management Group, and this could be interpreted as a bad signal for OODBMSs. Some participants pointed out that the ODMG standard is de facto not promoted by the companies participating in it, and that it seems to have gained more attention from the academy rather than from the industry.

Thus, the conclusion of the discussion was that there is still much to work for the object-oriented database community in the direction of making the OODBMS technology competitive with the relational one. Anyway, a necessary condition to the success of that is the market financial support.

8. Conclusion

The results of the First ECOOP Workshop on Object-Oriented Databases were, in our opinion, very positive, both in terms of number and quality of submissions, of number of participants, and of discussion at the workshop. We would like to remark that the workshop would not have been possible without the help and support of many people. The program committee and additional reviewers have input their time, skills and knowledge in reviewing the submissions of the

workshop. The final program is due to their hard work. We would also like to thank the Workshop Chair of ECOOP'99, Ana Moreira, for the scheduling and the local arrangements.

List of Participants

In the following we list the participants at the workshop, their affiliations and their e-mail addresses.

Name	Affliation	E-mail address
Alagiȁ Suad	Wichita State University, USA	alagic@cs.twsu.edu
Àlvarez-Gutiérrez Darío	University of Oviedo, Spain	darioa@lsi.uniovi.es
Aramburu Cabo María José	Universitat Jaume I, Spain	aramburu@inf.uji.es
Belen Martinez Prieto	University of Oviedo, Spain	belen@lsi.uniovi.es
Dmitriev Misha	University of Glasgow, UK	misha@dcs.gla.ac.uk
Düllmann Dirk	CERN Geneva, Switzerland	Dirk.Duellman@cern.ch
Even Susan	University of Twente, NL	seven@cs.utwente.nl
Fegaras Leonidas	University of Texas, USA	fegaras@lambda.uta.edu
Grouleff Morten	Mjølner Informatics, Denmark	mg@mjolner.dk
Guerrini Giovanna	Università di Genova, Italy	guerrini@disi.unige.it
Heide Damm Christian	University of Aarhus, Denmark	damm@daimi.au.dk
Korsholm Stephan	Mjølner Informatics, Denmark	sek@mjolner.dk
Merlo Isabella	Università di Genova, Italy	merloisa@disi.unige.it
Nylorg Mads	Technical University of Denmark	mn@iac.dtu.dk
Rashid Awais	Lancaster University, UK	marash@comp.lancs.ac.uk
Riedel Holger	University of Konstanz, Germany	Holger.Riedel@uni-konstanz.de
Risch Tore	University of Linköping, Sweden	torri@ida.liu.se
Rodrìguez M. Elena	Universtity of Catalonia, Spain	malena@lsi.upc.es
Saake Gunter	Magdeburg University, Germany	saake@iti.cs.uni-magdeburg.de
Sakkinen Markku	University of Jyväskylä, Finland	sakkinen@cs.jyu.fi
Shiers Jamie	CERN Geneva, Switzerland	Jamie.Shiers@cern.ch
Spelt David	University of Twente, NL	spelt@cs.utwente.nl
Torvill Bjorvand Anders	University of Oslo, Norway	torvill@trolldata.no

References

[1] M. J. Aramburu-Cabo and R. Berlanga-Llavori. Retrieval of Information from Temporal Document Databases. Available at ftp://www.disi.unige.it/person/GuerriniG/ecoopws99/final12.ps.gz.

[2] P.A. Bernstein, M.L. Brodie, S. Ceri, M.J. Franklin D. J. DeWitt, H. Garcia-Molina, J. Gray, G. Held, J. M. Hellerstein, H. V. Jagadish, M. Lesk, D. Maier, J. F. Naughton, H. Pirahesh, M. Stonebraker, and J. D. Ullman. The Asilomar Report on Database Research. *SIGMOD Record*, 27(4):74–80, 1998.

[3] R. Cattel, D. Barry, D. Bartels, M. Berler, J. Eastman, S. Gamerman, D. Jordan, A. Springer, H. Strickland, and D. Wade. *The Object Database Standard: ODMG 2.0.* Morgan Kaufmann, 1997.

[4] M. Dimitriev and M. Atkinson. Evolutionary Data conversion in the PJama Persistent Language. Available at ftp://www.disi.unige.it/person/ GuerriniG/ecoopws99/final14.ps.gz.

[5] M. Dmitriev. The First Experience of Class Evolution Support in PJama. In *Advances in Persistent Object System. Proc. of the 8th International Workshop on Persistent Object Systems (POS-8) and the 3rd International Workshop on Persistence and Java (PJAVA-3)*, pages 279–296, 1999.

[6] D. Dullman and J. Shiers. Object Database and Petabyte Storage - Dreams or Reality? Available at ftp://www.disi.unige.it/person/GuerriniG/ ecoopws99/final1.ps.gz.

[7] S. Even and D. Spelt. Compensation Methods to Support Generic Graph Editing: A Case Study in Automated Verification of Schema Requirements for an Advanced Transaction Model. Available at ftp://www.disi.unige.it/person/ GuerriniG/ecoopws99/final16.ps.gz.

[8] L. Fegaras. VOODOO: A Visual Object-Oriented Database Language For ODMG OQL. Available at ftp://www.disi.unige.it/person/GuerriniG/ ecoopws99/final13.ps.gz.

[9] V. Josifovski and T. Risch. Distributed Mediation using a Light-Weight OODBMS. Available at ftp://www.disi.unige.it/person/GuerriniG/ ecoopws99/final25.ps.gz.

[10] S. Korsholm. Transparent, Scalable, Efficient OO-Persistence. Available at ftp://www.disi.unige.it/person/GuerriniG/ecoopws99/final10.ps.gz.

[11] Ronald Morrison, Mick J. Jordan, and Malcolm P. Atkinson, editors. *Advances in Persistent Object System. Proc. of the 8th International Workshop on Persistent Object Systems (POS-8) and the 3rd International Workshop on Persistence and Java (PJAVA-3)*. Morgan Kaufmann, 1999.

[12] H. Oakasha, S. Conrad, and G. Saake. Consistency Management in Object-Oriented Databases. Available at ftp://www.disi.unige.it/person/GuerriniG/ ecoopws99/final20.ps.gz.

[13] A. Rashid and P. Sawyer. Evaluation for Evolution: How Well Commercial Systems Do. Available at ftp://www.disi.unige.it/person/GuerriniG/ ecoopws99/final5.ps.gz.

[14] H. Riedel. Outdating Outdated Objects. Available at ftp://www.disi.unige.it/ person/GuerriniG/ecoopws99/final11.ps.gz.

[15] C.M. Saracco. *Universal database management*. Morgan Kaufmann, 1998.

[16] M. Stonebraker. *Object-relational DBMSs*. Morgan Kaufmann, 1999.