# INCORPORATING AN OBJECT-ORIENTED DBMS INTO AN INTEGRAL OBJECT-ORIENTED SYSTEM

**Ana Belén Martínez Prieto, Darío Álvarez Gutiérrez, Juan Manuel Cueva Lovelle, Francisco Ortín Soler and Jesús Arturo Pérez Díaz**
**Department of Computer Science, University of Oviedo**
**Oviedo, Calvo Sotelo 33007, SPAIN**

## ABSTRACT

The aim of this paper is to show an object-oriented database management system, named BDOviedo3, which is highly integrated with a persistent object-oriented abstract machine and with an object-oriented operating system. It is being developed within a research project named Oviedo3[1], which tries to build an experimental integral object-oriented system where all the components share the same object-oriented paradigm.

This database management system has been structured as a set of modules that are briefly described. Specifically, indexing in object-oriented database management system is analyzed and the major features to create a possible indexing mechanism for the Oviedo3 OODBMS are proposed.

**Keywords**
Object-Orientation, OODBMS, Abstract-Machine, Operating System, Persistence, Indexing.

## 1. INTRODUCTION

The object-oriented technologies are well established in the software industry. Nevertheless, the adoption of the object-oriented paradigm is not done in an integral way in all the system components, producing:

- *Impedance mismatch*. It is basically due to two reasons. Firstly, the no suitable abstractions of the operating systems, so when a client object has to invoke a method that is offered by a server object, and both objects are not into the same address space, it is necessary to use the mechanisms that are offered by the operating system, and these mechanisms do not became proper to the object-oriented paradigm since they are oriented to communicate processes. In order to solve this problem an intermediate software is included as for example COM (Component Object Model) [1] and CORBA (Common Object Request Broker Architecture) [2]. In the second place, an impedance mismatch is too caused every time that the object-oriented applications need to use the operating system services. The interface services are close to the procedural paradigm, generally in the shape of system calls. A solution can be the use of adaptation layers which encapsulate the operating system interface by means of a class library (i.e. MFC library [3]).
- *Interoperability problem between object models*. Although different system elements use the object-oriented paradigm, an interoperability problem can exist between them. So, an application implemented using the C++ language, with the C++ object model, can easily interact with its objects, but when it wants to use objects that have been created with another programming language or another object-oriented database an interoperability problem appears. It is due to the fact that an object model for an element is not necessarily compatible with the object model of other elements. This problem can be solved adding software layers, again. So, CORBA defines a proper object model with basic data types, and a mapping is specified between the object model of each language and its interface into the CORBA model.

### Object-Oriented Technologies in Databases

OODBMSs have been developed to support new kinds of applications for which semantic and content are represented more efficiently with the object model [4]. Therefore, the OODBMSs present the two problems previously mentioned. Besides, object-orientation is reduced to wrappers in quite a number of commercial DBMSs (Persistence, UniSQL, Illustra, Omniscience,etc.). These systems, named *object-relational* [5], are based on traditional relational models. In these cases the impedance mismatch is again reduced adding a software layer that is the responsible for the assembling/disassembling of objects into tables.

### Object-Oriented Integral System

As mentioned, the interoperability problem and the impedance mismatch can be solved generating a proliferation of additional software layers that reduce system portability and flexibility. In fact, extra complexity is introduced in the system, further reducing global system performance.

Another approach in order to solve these problems is based on the construction of a homogeneous system in which all elements share the same object-oriented paradigm: an Integral Object-Oriented System. Oviedo3 [2][6] is a research project that tries to build an experimental integral object-oriented system based on that foundation [7]. All components: user interfaces, applications, languages, compilers, databases, … and the operating system itself share the same object-oriented paradigm.

[1] Project PBP-TIC97-01 "Object Oriented Integral System: Oviedo3". Partially supported by the second regional plan of research of the Principality of Asturias (FICYT).

[2] Oviedo Orientado a Objetos (Object-Oriented Oviedo). Oviedo is the name of the University.

The object-oriented operating (OOOS) provides only one abstraction: objects. Objects can only create new objects from a class or send messages to others. One technique to structure an OOOS aimed to support an integral object-oriented system which offers many advantages is to use an object-oriented abstract machine as the substrate of the OOOS. This machine offers the basic object model and support to all objects of the rest of the system.

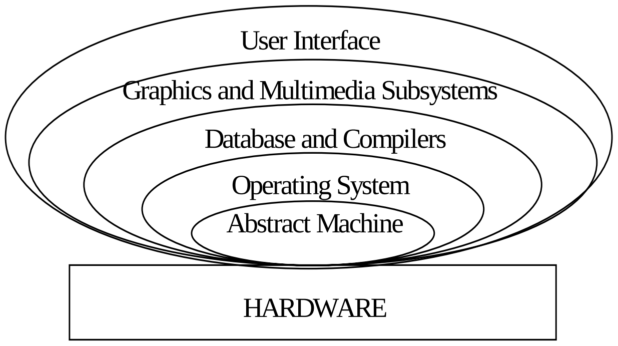


Figure 1 Components of the Oviedo3 System, shown in logical order of development

The object-oriented database management system (OODBMS) of the Oviedo3 system is highly integrated with the abstract machine and the operating system. It shares with them the same object-oriented paradigm. Furthermore, the OODBMS will take advantage of some important features that the operating system transparently achieves like: persistence, concurrency, distribution and security. BDOviedo3 will provide a query processing based on an indexing mechanism that is being currently constructed.

The master guidelines of the object-oriented abstract machine in which the Oviedo3 system is based are described in the next section. After this some preliminary advantages due to the machine are listed. Section 3 briefly describes the object-oriented operating system.

The major features and the architecture of the OODBMS designed for the Oviedo3 system are introduced in the next section. This section shows some benefits of using the object-oriented abstract machine and operating system for the construction of the OODBMS too.

Finally, the following sections show the functionality of every module in which this OODBMS has been structured. Specifically, in the section 5, different OODBMS indexes are listed and the major features for the indexing mechanism of the BDOviedo3 OODBMS are introduced.

## 2.THE OBJECT-ORIENTED ABSTRACT MACHINE

The abstract machine, called Carbayonia[3], supports an object model with the following features [8]: object identity and abstraction, encapsulation, polymorphism or message passing, inheritance and also generic and aggregation relationships between objects. This model includes object-orientation concepts widely accepted and allows, therefore, to represent the most used programming languages models.

[3] "Carbayón" means "big oak" in Asturian, the local language. It is a symbol for the city of Oviedo. "Carbayonia" means land of oaks.

Figure 2 shows the architecture of the abstract machine consists of five areas: class area, references area, instance area, threads area and system references area. Each area can be considered as an object in charge of the management of its data. The main characteristic of the machine is that every action upon an object is made using a reference to it.

- *Class area*. Maintains the description of each class. There is a set of primitive classes defined permanently in this area.
- *References area*. Stores the references. Every reference has a type (relates to the class area) and points to a object of this type (relates to the instance area).
- *Instance area*. Stores objects created. At run time, the information of its class can be accessed in the class area.
- *Threads area*. A main thread is created for every running application. This area stores the set of existing threads at run time. Every thread has a contexts heap.
- *System references area*. Contains references with specific functions in the machine.

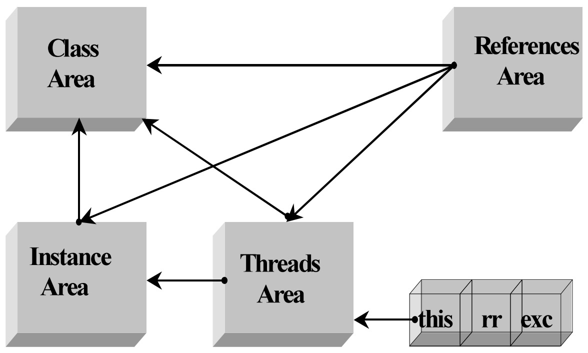


Figure 2 Abstract Machine Architecture

The machine language is a pure object-oriented low level language, named Carbayón. It allows class declaration, method definition and exception handling. It will be the intermediate language used by compilers of programming languages like Eiffel, C++, etc. Once compiled into this intermediate language, the source language of object creation is irrelevant. There are a number of classes built into the class area of the machine, with "Object" as the root class.

Some of the preliminar advantages derived from using an abstract machine in general for this system are:

- *Portability.* All system elements, from the operating system, databases, etc. to applications developed in it will work without modifications in every platform where the abstract machine is present.
- *Impedance mismatch is removed*. There is a common object model for the system, supplied by the machine for every object. Thus impedance mismatch between user objects and the operating system is removed.

- *Programming language independence*. Once an object is created in the machine, it is independent from the programming language used to define it. Objects can interact, migrate, etc. with no need for additional mechanisms by the use of different languages.

- *Implementation of operating system features.* The abstract machine facilitates the implementation of operating system features such as: persistence, security, distribution, and concurrency. So, the persistence is a native property of the abstract machine. That is, the abstract machine functionality has been extended to offer persistent objects. This will also benefit the OODBMS. For more information about the abstract machine see [7,9].

## 3. THE OBJECT-ORIENTED OPERATING SYSTEM

The operating system, named SO4, offers the abstraction of a single object space where objects exist indefinitely, virtually infinite, and where objects placed in different machines cooperate transparently using messages. Besides, the operating system transparently achieves a set of important features: capability-based protection mechanism, complete persistence, object distribution and concurrency.

Security ensures only authorized objects are allowed to send messages to other objects. Persistence makes objects stay in the system until they are no longer needed. Distribution permits object invocation regardless its location. Concurrency gives the object model the ability to execute tasks in parallel. For more information about the operating system see [10].

## 4. THE OBJECT-ORIENTED DATABASE MANAGEMENT SYSTEM

BDOviedo3[4] is the OODBMS for the Oviedo3 system. As it is an OODBMS provides the features of an object model and all capabilities of a DBMS. The BDOviedo3 object model is the abstract machine object model, and some DBMS capabilities (such as concurrency, persistence, etc.) are supplied by the operating system.

BDOviedo3 is a pure OODBMS. Applications development uses object-oriented analysis and design tools. Then, modeled objects can be directly translated into object-oriented programming languages objects, and finally objects can be directly stored in the database using the abstract machine persistence system.

Furthermore, BDOviedo3 system can be customized. Selecting the indexing mechanism, the cost model functions for queries, etc. is allowed. The main idea is to make comparisons between different strategies for index management, cost models, etc.

Nowadays there are a number of OODBMS (Thor, Poet, ObjectStore, Ode, etc.) and persistent storage managers (Shore, Texas, PSE, etc.). Some of them are commercial as well as research projects of different universities. Shore [11] is a persistent object system, expanding on the basic capabilities of Exodus [12], that represents a merger of object-oriented database and file system technologies. Texas [13] is an object-oriented persistent storage implemented as a C++ library. Thor [14] is an OODBMS that is intended to run on a heterogeneous collection of machines connected by a communications network. It provides a universe of objects that can be shared by programs written in different programming languages. Poet [15] and ObjectStore [16] are commercial OODBMS that work on C++ and include version-control mechanism, transaction management, etc. Poet works on Java too. Nevertheless, we think that the structure and features mentioned before (abstract machine and operating system) are very suited for an OODBMS and worth to research.

### Advantages Taken by Building on the Operating System and the Abstract Machine

The existence of an object-oriented operating system and ultimately of an object-oriented abstract machine provides a set of benefits for the construction of the OODBMS:

- *Facilitates its construction.* So, the database engine can take advantage of some operating system capabilities, such as: security, persistence, distribution and concurrency, building on them incrementally by means of the object-orientation present in the system.

- *Seamless integration with the rest of the system*. The OODBMS is not an individual element inside this integral system, like in conventional operating systems. The OODBMS is an integral part of the computation environment. Database objects are objects just like other objets in the system (operating system or user ones). The database only offers convenient access to them. The system is composed of a set of homogeneous objects. The OODBMS can be seen as playing the part of the file system in conventional operating systems, but with database capabilities. The database would not be used in isolation, but as a management system encompassing all objects in the system. For example, a user would find any object uniformly by querying the database.

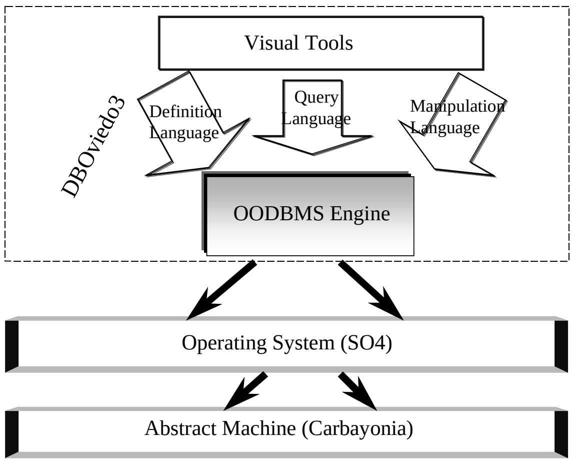
- *Performance increase*. It is not necessary to add new software layers to conventional operating systems to fill the semantic gap between operating system abstractions and objects. The integral system is already object-oriented.

- *Productivity increase*. Building database applications on this system is more productive, since it is not necessary for the programmer to change paradigms. Database and operating system share the same object-oriented paradigm, which is used uniformly in the system. For example, the same query language would be applied to database programming and user interface search tasks.

### BDOviedo3 Architecture

The BDOviedo3 OODBMS has been structured into three major modules (Figure 3). At present, a first prototype is being developed. Some of its features are briefly described in the next sections. More advanced features of the OODBMS, such as schema evolution, object versioning, etc. will be considered in future prototypes.

[4] Base de Datos de Oviedo3 (Oviedo3 Database)



## THE ENGINE

This module has two basic functions, query processing and storage management, that will be described in the next paragraphs.

### Query Processing

The majority of techniques for processing relational queries need to be extended, and some new techniques need to be developed for processing and optimizing object-oriented database queries. As, OODBMS queries can involve different data types (it makes the design of an object algebra harder), path expressions (an object may have references to other objects) and methods (which make difficult to estimate the cost of executing a query). The existence of class hierarchy represents another problem for the query processor.

Despite the above differences between relational and object-oriented query processing, methodologies for processing relational queries can be adopted for processing object-oriented queries. There are basically two such methodologies [17], algebra-based optimization and cost estimation-based optimization.

Algebra-based optimization consists of primarily two steps. In the first step, the input query is expressed in an algebraic expression, which is then transformed into a semantically equivalent but more efficient expression using equivalence transformation rules. In the second step, physical characteristics of the database such as the existence of fast access paths and database statistics are taken into consideration to generate a concrete and efficient execution plan for the algebraic expression obtained in the first step. The advantages of this methodology include extensibility and a relative easiness for implementation. The main drawback is that the search space for optimization in the second step is limited by the algebraic expression generated in the first step. So, there is a good possibility that the resulting execution plan is not close to an optimal plan.

The cost estimation-based optimization methodology tries to combine the above two steps as follows. The equivalence transformation rules are used to systematically transform the initial algebraic expression to all reasonable and equivalent expressions, and for each such expression, the cost of its best execution plan is estimated using the physical characteristics of the database. Eventually, the execution plan with the lowest estimated cost is selected for actual execution. The cost estimation-based approach is usually more effective than the algebra-based approach. This is the approach [18] that is being analyzed for BDOviedo3 OODBMS query processing.

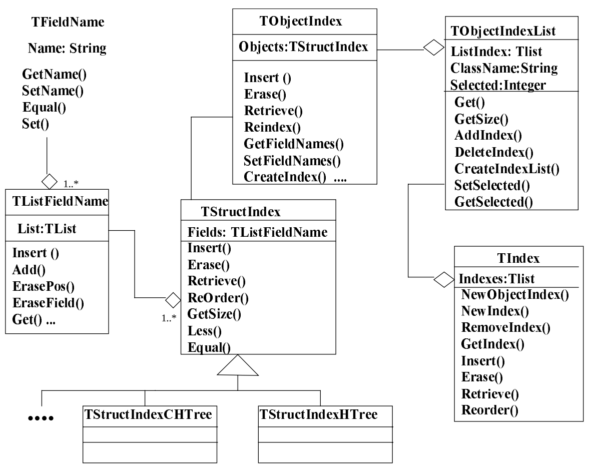
### Indexing Techniques

An important issue related to query languages concerns optimization techniques and access structures able to reduce query processing costs. Indexes contribute significantly to the efficient processing of database queries. Indexing techniques for OODBMSs can be classified [19] as structural and behavioral.

*Structural indexing* is based on object attributes. It is very important because most object-oriented query languages allow query predicates to be issued against object attributes. Structural indexing techniques proposed so far can be classified into three categories: the first category providing support for nested predicates or aggregation hierarchy (such as Nested index, Path index, Multiindex, etc.). The second, supporting queries issued against an inheritance hierarchy (such as CH_Tree, H_tree, etc.). The last category supporting for both class hierarchies and composition hierarchies (i.e. Nested Inherited index).

*Behavioral indexing* aims at providing efficient execution for queries containing method invocations. It is based on pre-computing or caching method results and storing them into an index. The major problem of this approach is how to detect changes to objects that invalidate the results of a method.

**Indexing Techniques Supporting Queries Issued Against an Inheritance Hierarchy.** Next, some indexing techniques in order to speed up the associative search for supporting queries against inheritance hierarchy are described. These techniques are based on the fact that an instance of a subclass is also an instance of its superclass. As a result, the access scope of a query against a class generally includes not only its instances but also those of all its subclasses. A query may also be formulated explicitly against a class and some of its subclasses. In order to support the superclass-subclass relationship efficiently, the index must achieve two objectives. First, the index must support efficient retrieval of instances from a single class. Second, it must also support efficient retrieval of instances from classes in a hierarchy of classes. These


TFieldName
Name: String
GetName()
SetName()
Equal()
Set()
1..*
TListFieldName
List:TList
Insert ()
Add()
ErasePos()
EraseField()
Get() ...
TObjectIndex
Objects:TStructIndex
Insert ()
Erase()
Retrieve()
Reindex()
GetFieldNames()
SetFieldNames()
CreateIndex() ....
TStructIndex
Fields: TListFieldName
Insert()
Erase()
Retrieve()
ReOrder()
GetSize()
Less()
Equal()
1..*
TObjectIndexList
ListIndex: Tlist
ClassName:String
Selected:Integer
Get()
GetSize()
AddIndex()
DeleteIndex()
CreateIndexList()
SetSelected()
GetSelected()
TIndex
Indexes:Tlist
NewObjectIndex()
NewIndex()
RemoveIndex()
GetIndex()
Insert()
Erase()
Retrieve()
Reorder()
....
TStructIndexCHTree
TStructIndexHTree

The query language proposed by ODMG (OQL) is not computationally complete. It provides declarative access to objects with a like-SQL syntax and provides high-level primitives to deal with sets of objects (although it is not restricted to this collection construct). OQL can be invoked from within programming languages for which an ODMG binding is defined. Conversely, OQL can invoke operations programmed in these languages.

### BDOviedo3 Applies the Standard

Some OODBMSs propose an object database language. For example, Thor [14] proposes the Theta language, Ode [29] proposes the O++ language (a C++ extension), etc. However, the languages for BDOviedo3 will follow as much as possible the ODMG 2.0 specification. Therefore, it is convenient to keep in mind the following points:

- The base object-oriented programming language for the database languages (ODL, OML and OQL) must be selected. This language can be C++, Java, etc. But, it is possible to use an object-oriented language (C++ Oviedo3) designed for Oviedo3 because when it is compiled Carbayón object code for the abstract machine is already generated. The C++ Oviedo3 language takes advantage of some abstract machine features.

- The BDOviedo3 OML will be created by extending the C++ Oviedo3 language with the database capabilities (i.e. queries, transactions…). Any database manipulation program implemented with this language will generate Carbayon object code when compiled.

```
Class Professor
{
attribute string FirstName;
attribute string LastName;
attribute short Age;
short GetAge();
relationship  set<Section>  Teaches  inverse
Section::Is_Taught_By;
};


Class Section
{
attribute short Number;
short GetNumber();
relationship  Professor  Is_Taught_By  inverse
Professor::Teaches;
};
```

Figure 5 ODL specification in the BDOViedo3 System

- The ODL and OQL specifications can be directly translated to the abstract machine language. A translation to C++ Oviedo3 intermediate language can be made too. The first option is used in the current prototype (Figure 5 and Figure 6) .

- The object code will be linked with the BDOviedo3 engine that will supply all DBMS capabilities.

```
Class Professor

AGGREGATION
     /* class attributes */
      LastName:String;
      FirstName:String;
      Age:Integer;

      /* relationship attributes */
      Teaches:Tset;

METHODS
      FieldbyName(ParamName:String):Object;

REFS
       Bres:Bool;

INSTANCES
      Param1str:String('LastName');
      Param2str:String('FirstName');
      Param3str:String('Age');
CODE

     ParamName.Equal(Param1str):bRes;
     JFD bRes, Label1;
     Assign rr, LastName;
     Jmp Fend;

Label1:

      ParamName.Equal(Param2str):bRes;
      JFD bRes, Label2;
      Assign rr,FirstName;
      Jmp FEnd;

Label2:
      ParamName.Equal(Param3str):bRes;
      JFD bRes, Label3;
      Assign rr, Age;
      Jmp FEnd;

Label3:

FEnd:
      Exit;
ENDCODE

/* class methods */

GetAge():Integer
CODE
ENDCODE

ENDCLASS
```

Figure 6 Carbayón code generated from the ODL specification of the Professor Class

## 7. CONCLUSIONS

The combination of an object-oriented abstract machine offering object support with an object-oriented operating system implemented as a set of objects is a promising way of structuring object-oriented integral systems. Furthermore, it facilitates the construction of an integrated OODBMS. Advantages such as portability, language independence and impedance mismatch removal are complemented with the improvement and ease of implementation of the funcionality of the OODBMS.

BDOviedo3 is a pure OODBMS, the objects defined with an object-oriented programming language are allowed to be directly stored into the database (using the abstract machine persistence system). Besides, BDOviedo3 system can be customized. The indexing mechanism, the cost functions for queries, etc. can be selected, in order to comparisons between different strategies can be made.

We expect to find many difficulties when we integrate the different mechanisms which compose the system. Nevertheless we believe this mixture of properties is very promising and it will be worth building a system like this.

Keep in mind, our main aim is to build a working prototype in which the primary elements of the system work. Subsequently, other aspects, such as transactions management, schema evolution, versioning, etc. will be considered.

## 8. REFERENCES

[1] D. Rogerdson, "Inside COM", Microsoft Press,1996.

[2] "Object Manaement Group. Common Object Request Broker Architecture and Specification (CORBA)", URL http://www.omg.org, March 1998.

[3] "Microsoft Corporation. Microsoft Visual C++ MFC Library Reference," Microsoft Press, 1997.

[4] E. Bertino and L. Martino, "Object-Oriented Database Systems: Concepts and Architectures," Addison-Wesley, 1993.

[5] M. Stonebraker and D. Moore, "Object-Relational DBMSs," Morgan Kaufmann, 1996.

[6] J.M. Cueva Lovelle et. al, "El Sistema Integral Orientado a Objetos: Oviedo3," (The Integral Object-Oriented System : Oviedo3), II Jornadas sobre Tecnologías Orientadas a Objetos, Oviedo, Spain, 1996, (in spanish).

[7] D. Álvarez, "Persistencia Completa para un Sistema Operativo Orientado a Objetos usando una Máquina Abstracta con Arquitectura Reflectiva," (Complete Persistence for an Object-Oriented Operating System using an Abstract Machine with Reflective Architecture), Doctoral Thesis, University of Oviedo, Spain, March 1998.

[8] G. Booch, "Object Oriented Analysis and Design with Applications," Addison-Wesley, 1994.

[9] F. Ortín, D. Álvarez, R. Izquierdo, A.B. Martínez and J.M. Cueva, "El Sistema de Persistencia en Oviedo3," (The Persistence System for Oviedo3), III Jornadas de Tecnologías de Objetos, Sevilla, Spain, 1997, (in spanish).

[10] D. Álvarez, L. Tajes et. al., "An Object-Oriented Abstract Machine as the Substrate for an Object-Oriented Operating System," 11th European Conference on Object-Oriented Programming (Workshop). Jyväskylä 1997.

[11] "Shore- A High-Performance, Scalable, Persistent Object Repository," URL http://www.cs.wisc.edu/shore/, March 1998.

[12] "Exodus-An Extensible Object-Oriented Database System Toolkit," URL http://www.cs.wisc.edu/exodus, March 1998.

[13] V. Singhal, S.V. Kakkad and P.R. Wilson, "Texas: An Efficient, Portable Persistent Store," Proc. of the Fifth International Workshop on Persistent Object Systems, Italia, September-92.

[14] "Programming Methodology Group," URL http:// www.thor.lcs.mit.edu/index.html, March 1998.

[15] "Poet Software," URL http://www.poet.com, March 1998.

[16] "The ObjectStore ODBMS Resource Center," URL http://www.odi.com/products/produts.html, March 1998.

[17] C. Yu and W. Meng, "Principles of Database Query Processing for Advanced Applications," Morgan Kaufmann, 1998.

[18] E. Bertino and P. Foscoli, "A Model of Cost Functions for Object-Oriented Queries," Proc. of 5th International Workshop on Persistent Object Systems. Italia, 1992.

[19] E. Bertino and P. Foscoli, "Index Organizations for Object-Oriented Database Systems," IEEE Transactions on Knowledge and Data Engineering. Vol.7, 1995.

[20] W. Kim, K.C. Kim and A.Dale. "Indexing Techniques for Object-Oriented Databases," in Object-Oriented Concepts, Databases, and Applications, Addison-Wesley, 1989.

[21] S. Ramaswamy and C. Kanellakis. "OODB Indexing by Class Division," ACM SIGMOD, 1995.

[22] C. Chin, B. Chin and H. Lu, "H-trees: A Dinamic Associative Search Index for OODB," ACM SIGMOD, 1992.

[23] E. Bertino, "Index Configuration in Object Oriented Databases," VLDB Journal,3, 1994.

[24] M. Ozsu, "Transaction Models and Transaction Management in Object Oriented Database Management Systems," In Object-Oriented Database Systems, Springer-Verlag, 1994.

[25] M. Loomis,"Object Databases. The Essentials," Addison-Wesley, 1995.

[26] "Object Database Management Group," URL http://www.odmg.org, March 1998.

[27] R. Cattell, T. Atwood, J. Duhl et. al, "The Object Database Standard:ODMG-93," Morgan Kaufmann, 1994.

[28] R. Cattell, D. Barry, D. Bartels et al, "The Object Database Standard: ODMG 2.0," Morgan Kaufmann, 1997.

[29] D. Moore, "An Ode to persistence. Journal Object Oriented Programming," November-December, 1996.