

BDOVIEDO3 AN OBJECT-ORIENTED DBMS INCORPORATED TO AN INTEGRAL OBJECT-ORIENTED SYSTEM

Ana Belén Martínez Prieto, Darío Álvarez Gutiérrez, Juan Manuel Cueva Lovelle and Francisco Ortín Soler

Languages and Computer Systems Area. Department of Computer Science.

University of Oviedo

Calvo Sotelo, 33007 OVIEDO - SPAIN

E-mail: {belen,darioa,cueva,ortin}@pinon.ccu.uniovi.es

Keywords: Object-oriented database management system, Integral object-oriented system.

Abstract

The aim of this paper is to show the work in the field of databases within the Oviedo3¹ research project. The scope is the construction of an object-oriented database management system (OODBMS), called BDOviedo3, that is highly integrated with a persistent object-oriented abstract machine and an object-oriented operating system. We describe the major modules in which the OODBMS has been structured: engine, languages and tools. Besides, the indexing in OODBMS is analyzed and some features for the indexing mechanism of BDOviedo3 system are presented.

Project partially supported by the second regional plan of research of the Principality of Asturias (FICYT).

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 caused every time that a system element (i.e. operating system) must interact with another (i.e. object-oriented application).
- *Interoperability problem between object models*. 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.

The result is a proliferation of additional software layers trying to alleviate these problems, but reducing the system portability and flexibility, and introducing also an extra complexity in the system that reduces the global system performance.

¹ Project PBP-TIC97-01 "Object Oriented Integral System: Oviedo3".

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 [Bertino, 1993]. Therefore, the OODBMSs present the two problems previously mentioned.

Besides, object-orientation is reduced to wrappers in quite a number of commercial DBMSs. These systems are based on traditional relational models and are labelled like *object-relational* systems [Stonebraker, 1996]. Its presence is considerable in the market. For example, Persistence [Persistence, 1998] is a product that provides an object-oriented layer (by means of C++ programming language) to relational databases like Oracle, Sybase, Informix, etc. Other products are Illustra, UniSQL [Kim, 1995], Omniscience, etc. 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.

A Solution: Integral Object-Oriented System

An approach in order to solve the previously mentioned problems is to move the object-oriented support for the rest of the system to a common place into the operating system. Oviedo3² [Cueva, 1996] is a research project that tries to build an experimental integral object-oriented system based on that foundation. All components: user interfaces, applications, languages, compilers, databases, ... and the operating system itself share the same object-oriented paradigm.

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.

The OODBMS of the Oviedo3 system, called BDOviedo3³, is a pure OODBMS. Applications development uses object-oriented analysis and design tools. Then, modelled 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. This OODBMS is being constructed on the abstract machine and the operating system, sharing with them the same object-oriented paradigm (impedance mismatch disappears). Besides, the operating system transparently achieves a set of important features: persistence, security, distribution and concurrency, which the OODBMS will take advantage. BDOviedo3 will provide an efficient query processing based on an indexing mechanism that is being currently constructed. It includes visual tools that facilitate the work with the database, too.

Section 2 introduces some related work. In the next section the BDOviedo3 OODBMS is presented: objectives, Oviedo3 elements closely related with it, and its architecture. The following sections show the functionality of every module in which this OODBMS has been structured: engine, languages and tools. Specifically, in the section 4 different OODBMS indexes are listed and the major features for the indexing mechanism of the BDOviedo3 OODBMS are introduced.

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

³ Base de Datos de Oviedo3 (Oviedo3 Database).

2. RELATED WORK

There are a number of OODBMS and persistent storage managers nowadays: commercial as well as research projects of different universities. However, an OODBMS integrated with a persistent object-oriented abstract machine and an object-oriented operating system is a novel approach, and we think that this structure is very suited for an OODBMS and worth to research. If the Java Virtual Machine [Java, 1998] is analyzed as one of the most used abstract machines in history, it appears that such machine is not persistent. Even though there are different persistence proposals for it, one has not been adopted yet. Next some features of relevant systems studied are shown.

Persistent Storage Systems

Shore [Shore, 1998] is a persistent object system under development at the University of Wisconsin that represents a merger of object-oriented database and file system technologies. It expands on the basic capabilities of the EXODUS [Exodus, 1998] storage manager. Every persistent datum in Shore is an object, and each object has an identity denoted by a unique object identifier. Shore is a collection of cooperating data servers, with each data server containing typed persistent objects. To organize this universe of persistent Shore objects, an Unix-like name space is provided. The Shore type system is embodied by the Shore Data Language (SDL), which is quite similar in nature to the ODL proposal from the ODMG consortium. Besides, Shore provides support for concurrency control and crash recovery and optimized object queries over bulk types.

Texas [Singhal, 1992] is an object-oriented persistent storage system implemented as a C++ library at the University of Texas (Austin). An application linked with the Texas library can create and manipulate two kinds of objects: transient and persistent. In Texas, object persistence is orthogonal to the type. A persistent object is simply one that is allocated in the persistent heap, as opposed to the conventional heap, or in the activation stack or static area. A key component of the design is the use of pointer swizzling at page fault time, which exploits existing virtual memory features to implement large address spaces efficiently on stock hardware. Long pointers are used to implement an enormous address space, but are transparently converted to the hardware-supported pointer format when pages are loaded into virtual memory. Like any useful persistent store, Texas supports checkpointing and recovery.

Mneme [Cattell, 1994a] is an object manager developed at the University of Massachusetts. It provides object identifiers, persistence, concurrency control and recovery. The client interface to Mneme is a set of procedures. The Mneme data store is simply a heap of untyped objects. Objects are grouped into pools that can have different storage-management policies. The policy (i.e. for caching and clustering) is defined by the database administrator writing a policy module. One or more pools are stored in one operating system file.

PSE [Pse, 1998] is a persistent storage engine that allows the transparent retrieval and storage of objects in their native C++ format (Java format too). It is part of the ObjectStore family of products provided by Object Design, Inc. When PSE stores a C++ object, not only does it store it in its native format, but also it leaves all its pointers intact. When the object is transferred back to program memory, the pointer values are also left unchanged, it means, the objects are stored at the same virtual memory address that their originally occupied. PSE supports database-level locking and recoverable save points. The *PSE Pro* product supports also multi-user access, distributed processing, query processing, integrity control, access control, concurrency control, etc.

Object-Oriented Database Management Systems

Thor [Thor, 1998] is an OODBMS being developed at MIT. It 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. The Thor universe has a persistent root object, from which all other persistent objects can be reached. An object becomes persistent automatically if it is made reachable from the root. Its architecture is organized with an object repository and front-end/client pairs. The object repository resides at the server machine and is a collection of processes that manage the persistent storage of objects. The front-end process serves as an intermediary between the client program and the repository. For every client language supported by Thor a small code library that implements the interface between the client and the front-end is provided. All objects in Thor are defined and implemented using a new object-oriented language called Theta.

Ode [Ode, 1998] is an active OODBMS based on the EOS storage manager. It is commercially available from AT&T software. In Ode the persistence is a property of the object instance itself rather than the data type, there are not significant differences between accessing and manipulating an instance of an object persistent or transient, and it is transparent to move instances of objects from persistent store to volatile store. It allows allocate objects of any built-in or user-defined type or class in either transient or persistent store. Ode uses a simple language C++ extension, called O++ [Moore, 1996]. Besides, Ode supports versioning, triggers, C++ templates fully, and two working modes: client-server and single-user.

Poet [Poet, 1998] is an OODBMS commercially available from Poet Software. The database objects are created or instanced from classes that can be persistent. It supports class extents (automatically maintained), OQL as the query language and navigational access within a hierarchy. Poet includes a complete transaction model, a concurrency control mechanism, and an authorization mechanism for supervising the rights given to users groups and single users. Other more advanced features are cross-database references and schema versioning.

ObjectStore [ObjectStore, 1998] is a commercial product of Object Design. It uses a technique for storing objects in which persistence is not part of the type of an object. A virtual memory mapping architecture (VMMA) is designed to make the speed of dereferencing pointers to persistent objects the same as that for transient objects. The database manipulation language interface uses extensions to the C++ language allowing any C++ data, not only objects, to be persistent. Besides, it supports conventional transactions (short-term concurrency), version-control mechanism (long-term concurrency), multiple inheritance and referential integrity between two objects.

3. THE DATABASE MANAGEMENT SYSTEM BDOVIEDO3

BDOviedo3 is an OODBMS and therefore provides the features of an object model and all capabilities of a DBMS. The object model is the object model of the abstract machine adopted for the rest of elements of the system. Some capabilities of a DBMS such as persistence, concurrency, distribution and security are transparently supplied by the operating system.

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. Although the first prototype is

being implemented with a specific indexing mechanism and cost model, new ones could be easily added.

3.1 Objectives

The main purpose for constructing the BDOviedo3 OODBMS is to verify the following:

- *Using an integral object-oriented system as the basis facilitates the construction of an OODBMS.* 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 objects 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.

3.2 Oviedo3 Elements Closely Related with BDOviedo3

As has been previously mentioned, BDOviedo3 is highly integrated with the abstract machine and the operating system. Next sections describe briefly both elements.

3.2.1 Abstract Machine

The abstract machine, called Carbayonia⁴, supports an object model with the following features [Booch, 1994]: 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.

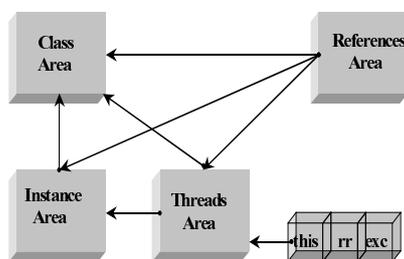


Figure 1 Abstract Machine Architecture

The architecture of the abstract machine consists of five areas (Figure 1): class area,

⁴ “Carbayón” means “big oak” in Asturian, the local language. It is a symbol for the city of Oviedo. “Carbayonia” means land of oaks.

references area, instance area, threads area and system references area. Each area can be considered as an object 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. The machine language, called Carbayón, is a pure object-oriented low level language. It allows class declaration, method definition and exception handling. It will be the intermediate language used by compilers of programming languages. 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.

Persistence [Álvarez, 1996] is a native property of the abstract machine. That is, the abstract machine functionality has been extended to offer persistent objects. For more information about the abstract machine see [Álvarez, 1998] and [Ortín, 1997].

3.2.2 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, it achieves a set of important features: security, persistence, 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 [Álvarez, 1997].

3.3 BDOviedo3 Architecture

The BDOviedo3 OODBMS has been structured into three major modules (Figure 2). 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.

- **Engine.** This module has two main objectives:
 - *Query processing.* The basic and more complex purpose of this module is the construction of the query-evaluation engine.
 - *Storage Management.* It is primarily focused on transaction management.
- **Languages.** The database languages (definition, manipulation and query) for the OODBMS are specified in this module.
- **Visual Tools.** This module deals with the implementation of a visual development environment that allows to the user to interact with the database.

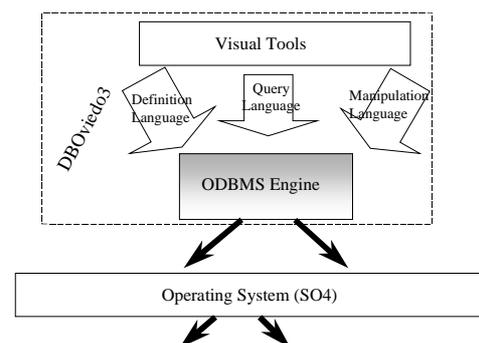


Figure 2 BDOviedo3 Architecture

4. THE ENGINE

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

4.1 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. This is necessary because there are many fundamental differences between the relational and object-oriented data models, and between the relational and object-oriented query languages. Thus, 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, *algebra-based optimization* and *cost estimation-based optimization* [Yu, 1998].

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 [Bertino, 1992] that is being analyzed for BDOviedo3 OODBMS query processing.

4.1.1 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 [Bertino, 1995] 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.

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 techniques are briefly stated because are being used in the first prototype of BDOviedo3.

CH- Tree: one index tree for all the classes of a class hierarchy

Kim [Kim, 1989] proposed a scheme, called CH-Tree (*Class Hierarchy Tree*), based on B^+ -trees. A CH-Tree maintains only one index tree for all the classes of a class hierarchy. A performance study shows that the indexing scheme of one index for all classes in a class hierarchy performs better than the indexing scheme that supports one index for each class. However, a major drawback of the CH-Tree is that it does not support the superclass-subclass relationship naturally. Searching for values in a single class is treated in the same way as searching for values in a hierarchy of classes. This scheme is now implemented for BDOviedo3 as it is the most used scheme in OODBMSs.

Ramaswamy [Ramaswamy, 1995] proposed an alternative to CH-Tree called CD (*Class Division*). It is viewed as an extension of CH-Tree. It consists of the index built for the root of the hierarchy (this is CH) plus some other indexes that allow speeding-up range queries.

H- Tree: an index tree for each class of a class hierarchy

In [Chin, 1992] an index called H-Tree (*Hierarchy Tree*) is proposed. An H-Tree structure is maintained for each class of a class hierarchy and these trees are nested according to their superclass-subclass relationships. When indexing an attribute, the H-Tree of the root class of a class hierarchy is nested with the H-Trees of all its immediate subclasses, and the H-Trees of the subclasses are nested with H-Trees of their respective subclasses and so forth. The nesting supports efficient traversal of the nested H-Trees by enabling traversal of a nested H-Tree to start at appropriate subtrees via the links maintained in its superclass's H-Tree. In addition, a nested H-Tree can also be accessed independent of its superclass's H-Tree. The difficult for dinamizing this scheme is the major drawback.

In [Bertino, 1994] and [Bertino, 1995] other structural indexing schemes, such as Multiindex, Nested Inherited Index, etc. are analyzed. Even the cost for retrieval, delete and insert operations is evaluated. These schemes are being considered for future versions of BDOviedo3.

The Indexing Mechanism of Oviedo3

Thus, BDOviedo3 provides an indexing mechanism with the following basic features:

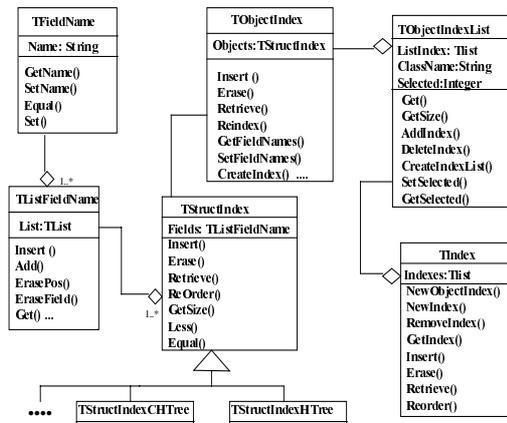


Figure 3 Indexing mechanism classes diagram

- The system allows different indexing schemes (CH-Tree, H-Tree, etc.). Initially, CH-Trees are considered in the first prototype that is being developed. However, the class hierarchy (Figure 3) designed for the implementation of the BDOviedo3 indexing mechanism allows easily adding new schemes.

- The mechanism is generic. It implies that the indexing can be performed over any data type (not simple types only). In order to get it the mechanism allows to define comparison operators users-defined.

4.2 Storage Management

Disk space management and main memory management are storage manager functions of a DBMS. In BDOviedo3 OODBMS this functionality is basically supplied by the persistence system of the abstract machine [Ortín, 1997]. The persistence system itself stores in disk and brings back to memory the persistent instances transparently. A pagination plus segmentation technique with a configurable page size is used for speeding-up the swapping between volatile and persistent memory. This storage system can be improved, for example, with some clustering techniques. These possibilities will be considered in next BDOviedo3 prototypes.

4.2.1 Transaction Management

The basic function of the transaction manager is to guarantee the appropriate processing of concurrent transactions. OODBMSs need more powerful techniques [Ozsu, 1994] than the traditional techniques (i.e. locking).

The first prototype that is being developed does not consider in detail the transactions management. It will be dealt in a more advanced stage of the system.

5. THE LANGUAGES

OODBMSs try to integrate the database languages with the programming languages (different traditionally) in order to provide a single syntax with a unified model and type system [Loomis, 1995]. It makes database objects appear as programming language objects. This is the scope of the standard suggested by ODMG (Object-Oriented Database Management Group) [ODMG, 1998].

ODMG 2.0 Standard

ODMG [Cattell, 1994b, Cattell, 1997] proposes to transparently extend an object-oriented programming language with persistent data, concurrency control, data recovery, associative queries, and other database capabilities. It specifies the syntax for an object definition language (ODL) and an object query language (OQL). However, it doesn't believe exclusively in a universal database manipulation language (DML) syntax. Instead, the DML

defines standard bindings of OODBMSs to object-oriented programming languages such as C++, Java or Smalltalk.

ODL is not intended to be a full programming language. It defines the characteristics of types, including their properties and operations. ODL defines only the signatures of operations and does not address the definition of the methods that implement those operations. Its major objective is to support the portability of database schemes across conforming OODBMSs. The C++, Smalltalk, and Java ODL bindings are designed to fit smoothly into the declarative syntax of their host programming language.

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

The languages for BDOviedo3 will follow as much as possible the ODMG 2.0 specification. Therefore, is convenient to bear 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.
- 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.
- The object code will be linked with the BDOviedo3 engine that will supply all DBMS capabilities.

6. VISUAL TOOLS

The main scope of this module is the construction of a visual development environment that allows interacting easy and in an efficient way with the database for a large number of users as possible. Such environment will have a set of visual tools grouped [Martínez, 1996] into:

- *Development Tools.* These tools allow as defining the database scheme as to manipulate and to query stored data in a visual way. The objective of these tools is to minimize the code that the user must write.
- *Administration Tools.* These tools facilitate the database (backup, regenerating the indexes, etc.) and users administration. This module demands a *control language* for protecting the database, restricting accesses to objects and methods, etc.

7. CONCLUSIONS

BDOviedo3 is an OODBMS built upon an integral object-oriented system. The main purpose for constructing of this OODBMS is verify the following: whether the integral object-oriented system facilitates the OODBMS construction, whether seamless integration with the rest of the system is allowed and whether a performance and productivity increase is obtained. Another advantage for BDOviedo3 is that all applications developed on it will work without modifications in every platform where the abstract machine (Carbayonia) is present.

BDOviedo3 is a pure OODBMS allowing objects defined with an object-oriented programming language to be directly stored into the database (using the abstract machine persistence system).

BDOviedo3 system can be customized. The indexing mechanism, the cost functions for queries, etc. can be selected, so comparisons between different strategies can be made.

A first prototype is being developed and it has been structured into three modules: engine, languages and visual tools. At present, we are working with the indexing mechanism and query processing. Translators from the database languages (ODL, OQL and OML C++Oviedo3) to Carbayón, abstract machine language, are also being developed.

8. REFERENCES

- [Álvarez, 1996] D. Álvarez, A.B. Martínez, J.M. Cueva, Persistencia en Sistemas Operativos Orientados a Objetos. Ventajas para los Sistemas de Gestión de Bases de Datos. (Persistence in Object-Oriented Operating Systems. Advantages for DBMSs), I Jornadas de Investigación y Docencia en Bases de Datos, Spain, 1996. (in spanish).
- [Álvarez, 1997] D. Álvarez, L. Tajés 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.
- [Álvarez, 1998] 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.
- [Bertino, 1992] E. Bertino, P. Foscoli, A Model of Cost Functions for Object-Oriented Queries, (Proc. of 5th International Workshop on Persistent Object Systems), Italia, 1992.
- [Bertino, 1993] E. Bertino, L. Martino, Object-Oriented Database Systems: Concepts and Architectures, Addison-Wesley, 1993.
- [Bertino, 1994] E. Bertino, Index Configuration in Object Oriented Databases, VLDB Journal , 1994.
- [Bertino, 1995] E. Bertino, P. Foscoli, Index Organizations for Object-Oriented Database Systems, IEEE Transactions on Knowledge and Data Engineering, Vol.7, 1995.
- [Booch, 1994] G. Booch, Object Oriented Analysis and Design with Applications, Addison-Wesley, 1994.
- [Cattell, 1994a] R. Cattell, Object Data Management. Object Oriented and Extended Relational Database Systems(Revised Edition), Addison Wesley, 1994.
- [Cattell, 1994b] R. Cattell, T. Atwood, J. Duhl . et. al, The Object Database Standard:ODMG-93, Morgan Kaufmann, 1994.
- [Cattell, 1997] R. Cattell , D. Barry, D. Bartels, et. al, The Object Database Standard: ODMG 2.0, Morgan Kaufmann, 1997.
- [Chin, 1992] C. Chin, B. Chin, H. Lu, H-trees: A Dinamic Associative Search Index for OODB, ACM SIGMOD, 1992.
- [Cueva, 1996] J.M. Cueva et. al, El Sistema Integral Orientado a Objetos: Oviedo3. (The Integral Object-Oriented System: Oviedo 3), II Jornadas sobre Tecnologías Orientadas a Objetos, Spain, 1996. (in spanish).
- [Kim, 1989] W. Kim, K.C. Kim, A. Dale, Indexing Techniques for Object-Oriented Databases. In W. Kim y F.H. Lochovsky (ed) : Object-Oriented Concepts, Databases, and Applications, Addison-Wesley, 1989.

- [Kim, 1995] W. Kim, Modern Database Systems. The Object Model, Interoperability, and Beyond, Addison-Wesley, 1995.
- [Loomis, 1995] M. Loomis, Object Databases. The Essentials, Addison-Wesley, 1995.
- [Martínez, 1996] A.B. Martínez, J.M. Cueva, D. Álvarez, Desarrollo de SGBDOO en Oviedo3. (The Construction of an OODBMS for Oviedo3), I Jornadas de Investigación y Docencia en Bases de Datos, Spain, 1996. (in spanish).
- [Moore, 1996] D. Moore, An Ode to persistence. Journal Object Oriented Programming, November-December, 1996.
- [Ortín, 1997] F. Ortín, D. Álvarez, R. Izquierdo, A.B. Martínez, J.M. Cueva , El Sistema de Persistencia en Oviedo3. (The Persistence System for Oviedo3),III Jornadas de Tecnologías de Objetos, Spain, 1997. (in spanish).
- [Ozsu, 1994] M. Ozsu, Transaction Models and Transaction Management in Object Oriented Database Management Systems, In A. Dogac, M. Ozsu et. al : Object-Oriented Database Systems, Springer-Verlag, 1994.
- [Ramaswamy, 1995] S. Ramaswamy, C. Kanellakis, OODB Indexing by Class Division, ACM SIGMOD, 1995.
- [Singhal, 1992] V. Singhal, S.V. Kakkad, P.R. Wilson, Texas: An Efficient, Portable Persistent Store, (Proceedings of the Fifth International Workshop on Persistent Object Systems), Italia, 1992.
- [Stonebraker, 1996] M. Stonebraker. and D. Moore, Object-Relational DBMSs, Morgan Kaufmann, 1996.
- [Yu, 1998] C. Yu and W. Meng, Principles of Database Query Processing for Advanced Applications, Morgan Kaufmann, 1998.
- [Exodus, 1998] Exodus-An Extensible Object-Oriented Database System Toolkit, URL <http://www.cs.wisc.edu/exodus>, March 1998.
- [Java, 1998] Java Virtual Machine Specification, URL <http://www.javasoft.com/docs/books/vmspec/index.html>, March 1998.
- [ObjectStore, 1998] The ObjectStore ODBMS Resource Center, URL <http://www.odi.com/products/produts.html>, March 1998.
- [Ode, 1998] Database Systems Research Department. Lucent Technologies, URL <http://www-db.research.bell-labs.com/project/ode/ode-announce/index.html>, March 1998.
- [ODMG, 1998] Object Database Management Group. URL <http://www.odmg.org>, March 1998.
- [Persistence, 1998] Persistence, URL <http://www.persistence.com>, March 1998.
- [Poet, 1998] Poet Software, URL <http://www.poet.com>, March 1998.
- [Pse, 1998] ObjectStore PSE Resource Center, URL <http://www.odi.com/products/pse.html>, March 1998.
- [Shore, 1998] Shore- A High-Performance, Scalable, Persistent Object Repository, URL <http://www.cs.wisc.edu/shore/>, March 1998.
- [Thor, 1998] Programming Methodology Group, URL <http://www.thor.lcs.mit.edu/index.html>, March 1998.