Object Storage in DOE

A White Paper

SunSoft
A Sun Microsystems Company

Table of Contents

Object Storage in Project DOE

Overview

SunSoft™ wants to provide customers a range of alternatives for object data persistence and a range of alternatives for accessing persistent data in order to best suit customer requirements. As object-oriented technology becomes the dominant paradigm for application development in the 1990's, it is becoming increasingly important to provide object storage capabilities to developers and users. Because users have differing requirements for functionality, performance, and size, SunSoft will insure that a range of capabilities will be available that allows developers to choose the approach that best fits their needs.

While conventional UNIX files or relational databases are satisfactory for some purposes, storage mechanisms tailored to the granularity of objects are often more convenient. Table 1 shows the choices available to a user for storing objects persistently.

Table 1 Choices for Persistent Object Storage

Storage Mechanism	**Data Types Permitted**	**Functionality Provided**	**Example Application**
File Storage	stream of bytes	simple file operations	Simple text editor
Relational Database Management System	tables	SQL standard, heterogeneity	Accounts receivable entry
Persistent Storage Manager (PSM)	data structure described in Interface Definition Language (IDL)	semi-transparent field access	Nested documents
Object Database Management System (ODBMS)	any PL data structure, plus extensions	PL integration, concurrency, query, versions	CAD editor

This paper will focus on the last two categories in the table: Persistent Storage Manager (PSM) and Object Database Management Systems (ODBMS).

Benefits of Object Storage

We expect object storage to become increasingly important because a programming environment enabled to provide object storage allows users to more easily gain the benefits object-oriented development: reusability, modularity, and extensibility. These benefits translate to important achievements in a world where software is crucial to the way companies do business and compete:

- shorter time to market
- lower cost of development
- ability to manage complexity for next-generation business systems

ODBMS

Object-oriented database management systems (ODBMS) are typically used to build and deploy multi-user, distributed applications. Database management systems were developed to simplify software development by taking responsibility for:

- sharing of data among users
- ensuring data integrity and recovery
- distributing data in a network
- managing the query process for large sets of data

An ODBMS provides the traditional services of a DBMS and raises the level of abstraction from records to objects. This level of abstraction facilitates development of reusable code that is easier to maintain and enhance by closely supporting object-oriented languages.

Most ODBMS also augment traditional DBMS functionality with support for version and configuration management, and concurrency control for groupware applications. These systems provide the services necessary to manage millions of finer-grained objects in a distributed network application. ODBMS are ideal for the development of applications that rely on complex, non-record-oriented data types with inheritance (such as applications required for multi-media presentation or engineering design).

PSM

The Project DOE Persistent Storage Manager (PSM) can be thought of as a simple ODBMS that provides object storage and retrieval mechanisms without the more advanced ODBMS features such as versions, complex queries, and object-level, multi-user concurrency control. The PSM is designed for applications that do not require these more advanced features and that should not be burdened with the associated cost. The PSM is integrated with programming languages in a fashion similar to ODBMS, which simplifies upgrade to a full ODBMS if and when additional functionality is required.

ORBs

Object Request Broker (ORBs) are used to communicate between objects in distributed environments. An ORB is designed to support transparent communication and distribution of objects among applications from different vendors installed on distributed, heterogeneous platforms. The ORB provides services for a client to transparently:

- locate an object in a wide area network
- verify that the object is the expected type
- invoke a method on that object to generate a desired behavior
- transfer invocation data to and from the object in a type-safe and architecture-independent fashion

This technology makes it possible for applications or database systems written by different vendors to exchange information. The client is insulated from having to know what application or vendor created the object, and from having to worry about the hardware or operating system where the object is resident. You can write object "wrappers" around existing applications to allow them to coexist in an object-oriented world.

ORB Client to Server Object Interfaces

The ORB client/server architecture is described in the SunSoft *Project DOE: Future Solaris Technologies* white paper; it is recommended that you read this paper or have some understanding of the ORB, BOA, IDL, and the rest of the DOE architecture before proceeding with this document.

In the world of DOE, client objects call server objects through the ORB. The call interface for the server objects is described in IDL. The server objects must manage the data that they require for the functionality that they support; typically, they will keep their data in some kind of storage manager. Figure 1 pictorially represents the architecture.

Note – The same object may be a client object in one context and a server object in another context; server objects provide a service, and client objects invoke the server objects to use that service.

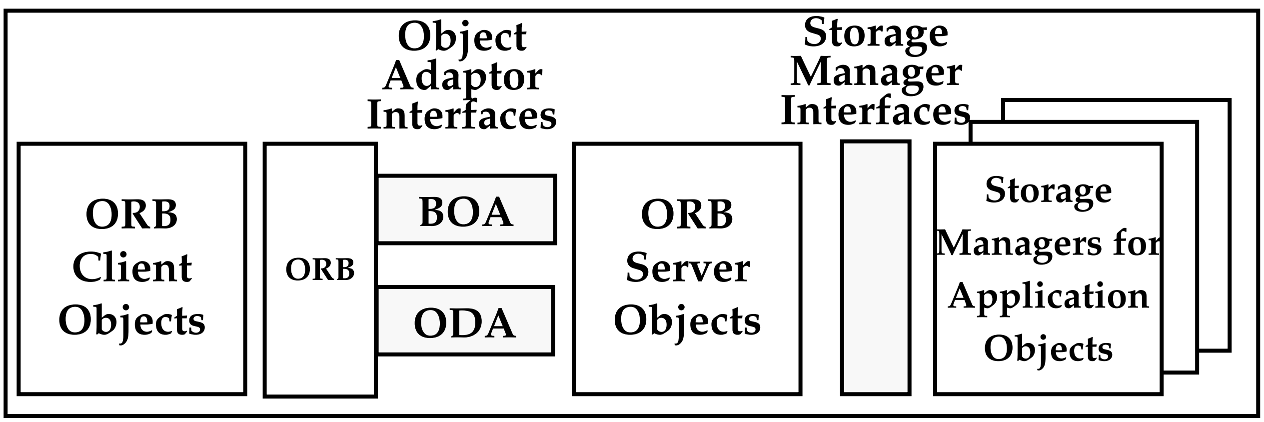

Figure 1 Object storage in the DOE environment

Figure 1 shows two interfaces in this architecture for which standards need to be established:

1. An *object adaptor interface* that connects the ORB and the server objects. An object adapter is the interface by which the ORB locates and invokes a server object when it is called by a client. An ORB *objref* (object reference) uniquely identifies a server object.

2. A *storage manager interface* that connects server objects to the storage manager used to store their persistent data.

A server object locates and accesses its persistent data through this interface. If the storage manager is the PSM or an ODBMS, this data is maintained as one or more data objects per server object accessible through unique identifiers.

In both cases, more than one alternative interface standard is needed because a wide range of application requirements within the architecture needs to be supported.

Object Adaptors

The CORBA specification defines a Basic Object Adapter (BOA) intended for typical medium-to-large server objects (such as documents or windows) implemented in a host programming language.

The CORBA specification also anticipated the need for other types of adaptors for server objects. SunSoft has proposed an additional adaptor, called an Object Database Adaptor (ODA).

The ODA allows clients to access application objects stored in an ODBMS via the ORB. Calls to objects that are implemented via an object database are routed through the ODA. The object database engine is then invoked to retrieve the state and set up the customer-supplied methods for the objects. See "Object Database Adaptor" for a discussion on SunSoft's approach to developing a prototype for an ODA.

Storage Manager Interfaces

A standard API needs to be defined for how server objects interface with the storage managers used to store objects in an application. Figure 2 illustrates some choices for storage manager interfaces.

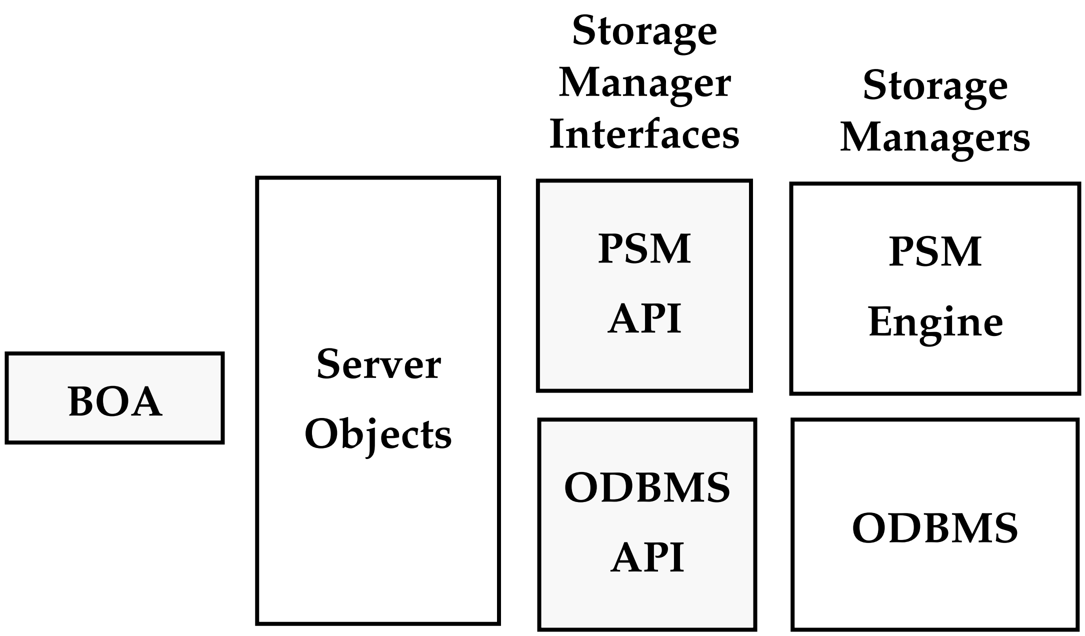

Figure 2 Choices for Storage Manager Interfaces

PSM

Users of the BOA and SunSoft's DOE will have the ability to store objects directly in the PSM that will be provided as part of the DOE environment. The interface to the PSM is via an API based upon C or C++. The PSM software supplies basic object storage capability for applications using SunSoft's DOE software. See "Persistent Storage Manager" for a discussion of the PSM.

The C++ interface is intended to be a subset of a full C++ ODBMS API standardized by the Object Database Management Group (ODMG). See "C++ Object Database Interface" for a discussion of the ODMG standard.

Although the PSM has less functionality than a full ODBMS, it has the advantage of using less system resources and being available as part of DOE. To help customers upgrade to a full object database, the PSM programming language binding is targeted to be a compatible subset of the ODMG standard.[1]

Note – A derivative of the PSM interface was submitted to the Object Management Group (OMG) in response to a request for proposal (RFP) for persistence and was accepted as OMG COSS1.

ODBMS

For applications that need more sophisticated database capabilities than the capabilities that are provided by the PSM (such as concurrency control for data access, relationships, and complex query support), an object database is the right choice for storing objects. An object database can provide persistence for any data structure that can be described in the host programming language. Most object databases also support a query language and shared access to the data from multiple users and distributed sites though locking, versioning, and transaction mechanisms.

Because both DOE and object databases provide mechanisms for building distributed applications, it is SunSoft's goal to provide, either directly or through partnerships, a way for users that start with PSM to upgrade to a full-fledged object database; and for existing applications to share data between object databases and applications based upon the PSM.

Persistent Storage Manager

The PSM is built on a storage management engine supplied by Object Design, Inc. (ODI). The PSM is designed for DOE applications with simple object persistence requirements.

SunSoft's implementation of the PSM is based on a PSM Engine (PSME) developed by ODI. The PSME, a subset of ODI's database product, uses much less memory and system resources than a full object database product, yet

1. It is not a precise subset at the time of this writing because ODMG and PSM are still moving targets, and because ODBMS do not yet support multi-threaded processes.

allows the PSM to provide basic storage management for objects. SunSoft has extended PSME with a layer of software that exports the PSM interface. Figure 3 illustrates the PSM.

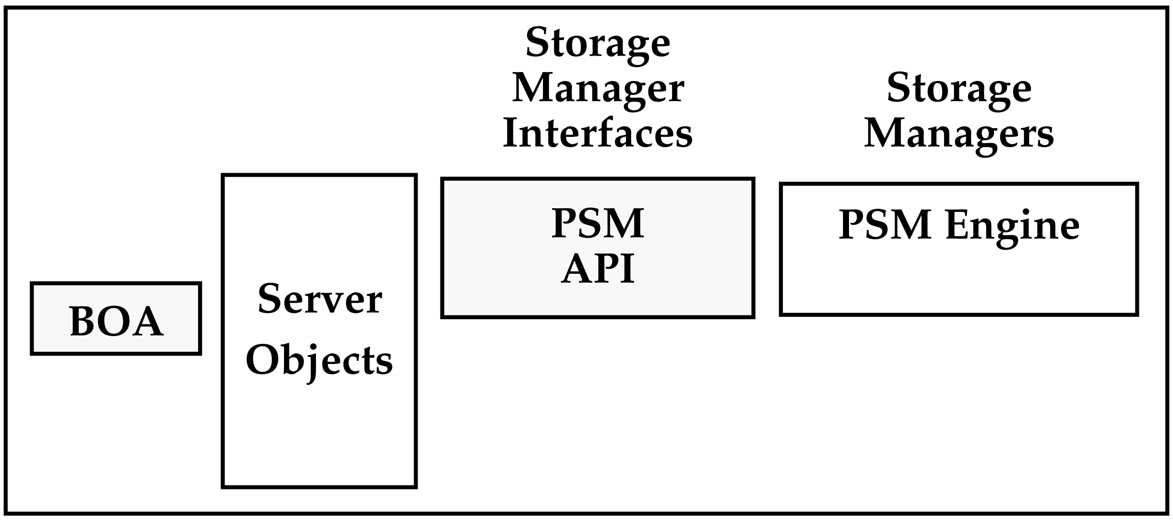

Figure 3 Persistent Storage Manager

The PSM is intended for use by new applications in the DOE environment that have basic storage management requirements but that do not need advanced database features such as sophisticated queries, data object versions, or multi-user database transactions. These applications implement concurrency control (if any) through ORB protocols rather than through a database; concurrent data access is not necessary since their data is private.

The PSM provides data storage at the granularity of individual data objects. An application's data objects are described in an extension of IDL rather than an extension of the host programming language declarations (as in ODBMS). The use of IDL provides a number of advantages:

- The schema description can be used for database access from multiple programming languages.
- DOE users are already familiar with IDL.

- Data objects can semi-transparently be replaced with ORB objects as an application schema evolves (for example, the application need not be rewritten if previously stored data attributes are instead computed).

The interface between the PSM data objects and the application is not through IDL and ORB calls, however; it is provided directly from C or C++ in order to provide performance and programming convenience similar to the best ODBMS.

SunSoft wants to make it easy for a PSM customer to upgrade to a full object database product. Some customers will find that as their application evolves, they require more functionality than the PSM provides. To rewrite all data management aspects of the application in order to switch to a full object database is not desirable.

We are keeping the PSM C++ interface as close as possible to the evolving C++ interface standard defined for object databases by ODMG. This will make the upgrade to an object database will be straightforward — the application program written for PSM should continue to work with the full object database product, and the customer can incrementally take advantage of the full object database functionality as is convenient.

C++ Object Database Interface

An object database provides an architecture that is significantly different than other DBMS as viewed by the customer and have significant added value in SunSoft's object environment. Rather than providing only a high-level query language such as SQL, an object database transparently integrates database capability with an application programming language. This transparency makes it unnecessary to learn a separate query language, obviates the need to copy data between database and programming language structures, and supports substantial performance advantages through data caching in applications. In addition, object databases provide the query language capabilities of relational systems, augmented to support the more comprehensive object-oriented data structures.

ODMG Standardization Effort

At SunSoft's invitation in 1991, the object database vendors formed a consortium, the Object Database Management Group (ODMG), to work on standards for programming language interfaces for their products. The ODMG is establishing shared, consistent protocols for managing objects supported by the object databases. The focus has been on the C++ interface for object databases.

At the time of this writing, ODMG has completed Version 1 of their standard, which has been published as “ODMG-93”.

The ODMG C++ standard benefits object database customers regardless of whether they use the ORB. The establishment of the standard facilitates building-blocks for customers, enhancing the overall utility of object-oriented software systems.

The ODMG standard also increases the popularity of object databases. Much of the early success of relational databases came from the standardization that they offered. For example, the SQL standard allows a high degree of portability and inter-operability between systems, simplifies learning new relational DBMS’s, and represents a wide endorsement of the relational approach. These advantages are especially important for object databases because the products in this area are offered by relatively new companies. The standard provides the investment protection of application portability across many vendors and also provides synergy for multi-vendor tools for application development.

The intense ODMG effort has given the object database industry a “jump start” towards standards that would otherwise have taken many years. The data schema, programming language binding, and data manipulation/query languages are portable. Eventually, the standards proposal should be helpful in allowing inter-operability between the object database products as well (for example, heterogeneous distributed databases communicating through the ORB).

Components of ODMG Standard

The following major components of the ODMG standard are described fully in the ODMG-93 Object Database Standards document:

- Object Model
  The ODMG members have agreed to support a common data model based upon the OMG Object Model. Since the OMG core model was designed to be a common denominator for object request brokers, object database systems, object programming languages, and applications, ODMG has defined extensions for object databases, in keeping with the OMG vision.

- Object Definition Language
  ODMG defines a data definition language for object databases. This language is called the object definition language, or ODL, to distinguish it from traditional database data definition languages, or DDL. OMG IDL is used as the basis for the ODMG ODL syntax.

- Object Query Language
  ODMG defines a declarative (non-procedural) object query language (OQL) for finding and updating database objects. Wherever possible, SQL syntax is used as the basis for OQL, although OQL must support more powerful capabilities (the results of a query need not be relations).

- C++ Language Binding
  The ODMG draft defines the standard binding of object databases to C++. It explains how to write portable C++ code that manipulates persistent objects. This binding is called the C++ OML, or object manipulation language. The C++ binding also includes a version of the ODL based on C++ syntax and a simplified version of OQL for queries on C++ object collections.

- Smalltalk Language Binding
  The ODMG draft defines a standard binding of object databases to Smalltalk. The Smalltalk binding is (symmetrically) called OML/ODL for Smalltalk.

- Object Database ORB Binding
  An appendix to the ODMG document describes the architecture for the object database adapter for the ORB.

The ODMG standard will be extended at a future date to incorporate other programming language bindings and an IDL binding.

Note – Unlike relational DBMS, object database data manipulation languages are tailored to specific application programming languages in order to provide a single, integrated environment for data manipulation. Therefore, a new chapter is needed in the standards document for each language binding.

Object Database Adaptor

The object database adaptor (ODA) makes object database objects into ORB objects; it provides an interface between the two object environments. Figure 5 illustrates this concept.

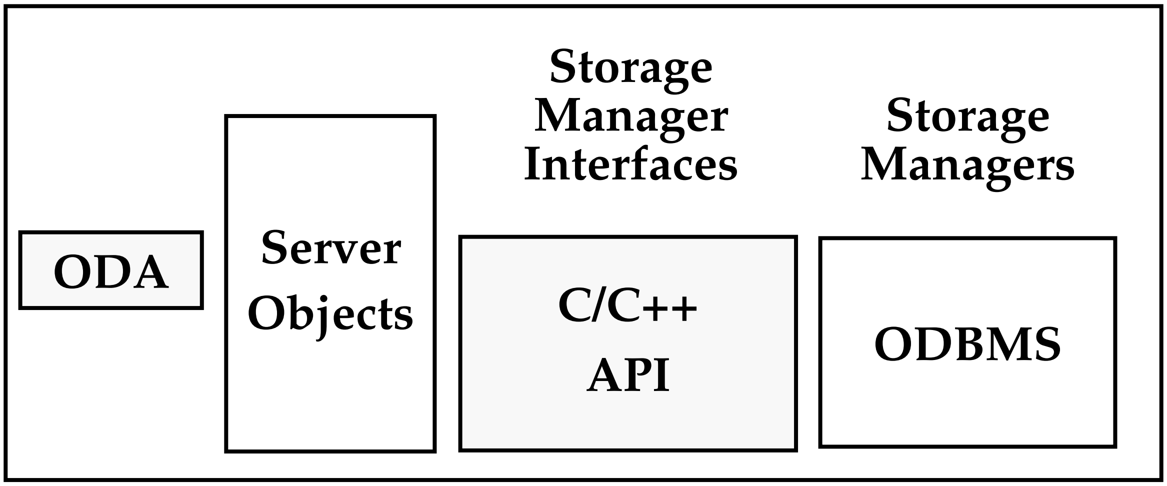

Figure 4 Object Database Adaptor

The ODA will allow customers that store objects in an object database to make their objects automatically accessible through the ORB.

Object databases overlap somewhat with the ORB in the functionality that they provide. Both systems provide remote access to objects; both systems have an object identifier mechanism and a mapping to make their objects look like C++ objects. However, an object database provides remote access to an object's data, which is then cached locally, while the ORB provides remote access to an object's methods, which are executed remotely. Thus, the systems are optimized for different application workloads and provide complimentary functionality.

The goal for an object database adaptor (ODA) is to allow object database products to integrate seamlessly with the ORB. Although we have not yet determined what design would give maximum value for our investment, preliminary work suggests that a simple ODA based upon the existing BOA could yield a number of benefits. In the longer term, a more sophisticated ODA could support some of the more advanced features of object databases and enhance the integration of these capabilities.

Status

Objectivity has agreed to design and build an ODA to interface the Objectivity/DB product with the ORB. Version 1 of the ODA will use the existing SunSoft BOA and all of the work will be done outside of SunSoft. SunSoft will publish the ODA API so that other vendors can adapt it to their products. At this time, no design documents are available from Objectivity.

The essence of ODA Version 1 would be a mechanism allowing object IDs used by the database to be “wrapped” in a ORB object reference. This wrapper allows an application to hand out references to its database objects for invocation (remotely or locally) through the ORB. It allows references to database objects to be stored in a non-database application or in another vendor's database system.

Procedures in our ORB libraries allow a customer to store a limited amount of application-defined data inside of *objrefs*; this capability can be used to hold the object IDs needed by the database system. When the object is later invoked, a “router” provided by the object database system can be invoked to find and initialize the object's state when it is invoked, given the object ID information.

For ODA Version 1, the object would always be invoked on a specific machine. The machine and routing process would be specified when an object database registers with the ODA.

Running in the client process is more complex to support due to security, data integrity, and concurrency control issues. ODA Version 2 will permit the server objects to be invoked in the client's address space, utilizing the object database's capability to cache data and run remotely. ODA Version 2 will probably be built using the Library Object Adaptor technology that SunSoft plans for a later release of the ORB.

When to Use

There are several potential benefits of an ODA, the most important of which is that ODBMS and the ORB can be used as complimentary technology long-term:

1. SunSoft customers will find it easier to use object database products in conjunction with the ORB (including customers that start with one of these products, then want to use the other product). This compatibility makes the ORB attractive relative to other ORB implementations, and makes it more likely that users of object databases will try using the ORB as well.
2. The effort leverages development investment in ORB technology and tools.
3. The ODA allows customers to use both an object database and the ORB, as appropriate. The customer may utilize the respective strengths of both these technologies.
4. The ODA and ORB lay the groundwork for a multi-vendor heterogeneous distributed database system.

Summary

There are a variety of ways to store and access object data in the DOE environment. SunSoft is working with partners to provide customers maximum flexibility for object-granularity data storage including access through the PSM, a C/C++ object database interface, an IDL object database interface, and an ODA.

To facilitate definition of a C/C++ object database interface, SunSoft is working in conjunction with the ODMG to standardize on interfaces to object databases; the query language, the object model, and the programming language bindings. The goal is to make it possible to write programs that can work with any of the object database products.

SunSoft, Inc.
2550 Garcia Avenue
Mountain View, CA 94043

For more information, call 1 800 227-9227.

Printed in USA 4/94 93382-002