

ODMG 2.0: A Standard for Object Storage

ODMG 2.0 builds on database, object and programming language standards to give developers portability and ease of use

by Doug Barry

Component Strategies - July 1998

The Internet explosion has fundamentally changed application development goals and strategies, requiring new applications that are dynamic and data-rich. To build them, developers are employing multi-tier architectures and object programming languages like Java. Consequently, object storage, especially Java language object storage, has become a crucial problem that many application developers face.

ODMG Simplifies Object Storage

ODMG 2.0 is the industry standard for persistent object storage. It builds upon existing database, object and programming language standards to simplify object storage and to ensure application portability. ODMG 2.0 extends Java, C++ and Smalltalk with complete database programming facilities, so that application developers are able to work entirely within the native language environment. Developers can store objects directly from Java, C++ or Smalltalk into ODMG 2.0-compliant relational, object/relational or object databases without having to manage the actual storage process. Tight language bindings integrate application and database programming into a single environment so that developers deal with a single data model. In addition, because it is actually built on top of other standards, ODMG 2.0 is very easy to learn and to use.

The two primary benefits of the ODMG standard are ease of use and portability. Using a common storage specification minimizes the need for training, because the ODMG bindings are native language extensions to existing object programming languages. That means, for example, that Java programmers can use the ODMG storage interface from within their existing Java language and type environment. Moreover, since objects are stored directly from within the programming language, it eliminates the need to write mapping code or to work in a specialized sub-language for data storage. Portability reduces software development risk by eliminating single-sourcing, in addition to facilitating reusability across different compliant platforms. By providing transparent object storage across a variety of platforms, ODMG allows users to choose the storage backend that fulfills their business and technical requirements.

Because the ODMG has broadened its charter to encompass any storage mechanism that uses the ODMG specification, the organization recently changed its name from the Object Database Management Group to the Object *Data* Management Group. This is especially relevant to the ODMG Java Binding, which is supported on object, object/relational and relational database management and object/relational mapping products.

The ODMG's members include product vendors and interested parties who collaborate to define data storage portability standards for object-oriented applications. Rick Cattell, the ODMG's chair and a distinguished engineer at JavaSoft started the organization in 1991 to create a set of standard language bindings for object storage. The first ODMG specification was released in 1993. The most current major release, ODMG 2.0, was published in 1997.

The ODMG Standard

The ODMG standard builds upon the existing OMG, SQL-92, INCITS (formerly ANSI) programming language standards, and JavaSoft's Java specification to define a framework for application portability between compliant data storage products. The standard's functional components include an Object Model, an Object Definition Language, an Object Query Language, and Language Bindings to Java, C++, and Smalltalk. Figure 1 shows the relationship between ODMG 2.0 and the standards upon which it is based.

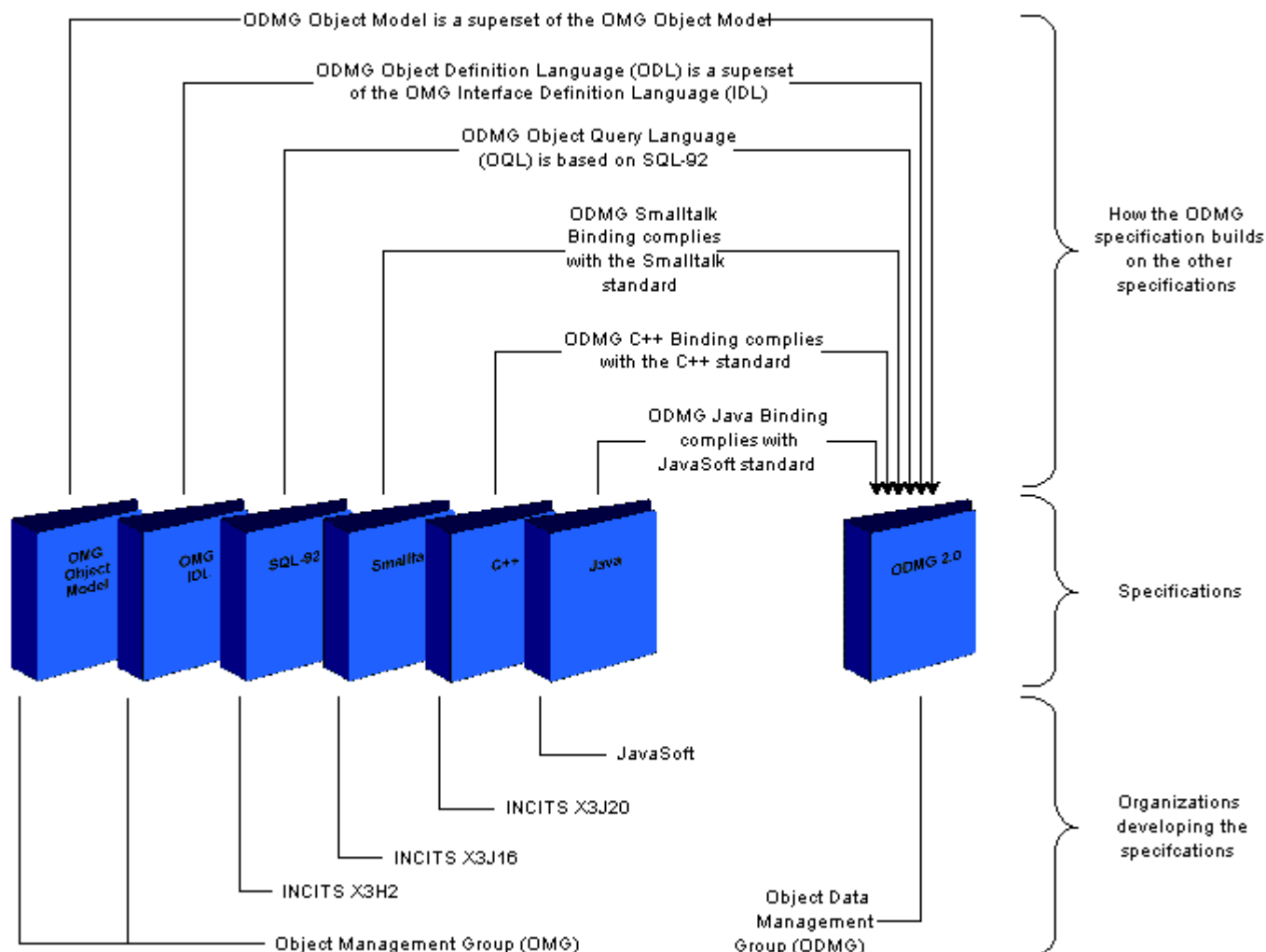


Figure 1. Standards on which ODMG 2.0 is built.

(adapted from Douglas K. Barry, *The Object Database Handbook*, John Wiley and Sons, 1996)

Object Model

The ODMG Object Model lies at the center of ODMG standard, and is based on the OMG Common Object Model. The Common Object Model was intended as a common denominator for object request brokers, object storage systems, object programming languages, and other applications. In accordance with the OMG architecture, the ODMG defined an ODBMS profile for the model, adding components like relationships and transactions to the OMG Common Object Model to support object storage needs. An instance is a first-class object that can be queried and manipulated directly.

The central features of the object model are:

- Attributes and relationships object properties
- Object operations (behavior) and exceptions
- Multiple inheritance
- Extents and keys
- Object naming, lifetime and identity
- Atomic, structured and collection literals
- List, set, bag and array collection classes
- Concurrency control and object locking
- Database operations

Object Definition Language (ODL)

The ODMG Object Definition Language (ODL) is a strict superset of the OMG Interface Definition Language (IDL) for database schema definition. Objects may be defined in terms of type, attributes, relationships and operations. ODL creates a layer of abstraction so that an ODL-generated schema is programming language- and database-independent to allow applications to be moved between compliant databases, different language implementations, or even translated into other Data Definition Languages (DDLs), such as those proposed by SQL3.

Object Query Language (OQL)

OQL is an SQL-like declarative language that provides a rich environment for efficient querying of database objects, including high-level primitives for object sets and structures. OQL is closely based on the query portion of SQL-92 and provides a superset of the SQL-92 SELECT syntax.

OQL also includes object extensions for object identity, complex objects, path expressions, operation invocation and inheritance. OQL's queries can invoke operations in ODMG language bindings, and OQL may be embedded in an ODMG language binding. OQL maintains object integrity by using an object's defined operations, rather its own update operators.

Below is an example of an OQL query that appears in the published ODMG standard (*ODMG 2.0*. ISBN 1-55860-463-4.). Notice its similarity to a SQL query, but with object extensions:

```
select c.address
from Persons p,
p.children c
where p.address.street="Main Street" and
count(p.children) >= 2 and
c.address.city != p.address.city
```

The "dot" notation used in the query traverses the data structure as shown in Figure 2. The query inspects all children of all "Persons" to find people who live on Main Street with at least two children. It returns only those addresses of children who do not live in the same city as their parents. It navigates from the Person class using the child reference to another instance of the Person class and then to the Address and City classes.

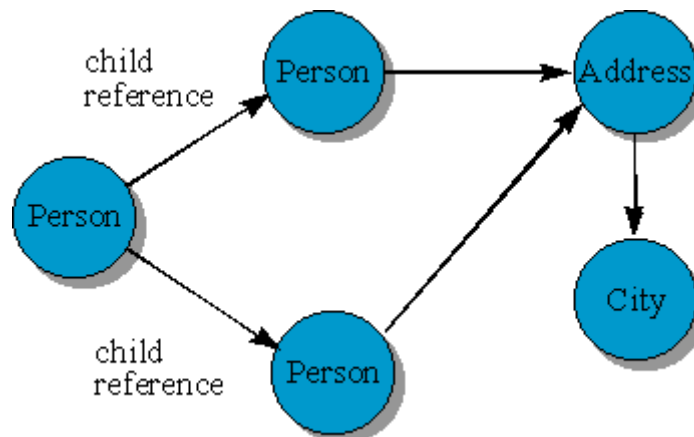


Figure 2. Navigating a data structure with OQL

Language Bindings

The ODMG standard's Java, C++ and Smalltalk bindings define Object Manipulation Languages (OMLs) that extend the programming languages to support persistent objects. The bindings also include support for OQL, navigation and transactions. The OML enables developers to work inside a single language environment without a separate database language.

Java Language Binding

The ODMG Java binding adheres to the same principles as the Smalltalk and C++ bindings. The binding uses established Java language practice and style to be natural to the Java environment and programmers. Instances of existing classes can be made persistent without changes to source code. Persistence is by reachability: once a transaction is committed, any objects that can be reached from root objects in the database are automatically made persistent in the database. The ODMG binding to Java adds classes and other constructs to the Java environment to support the ODMG object model, including collections, transactions and databases, without altering the semantics of the language.

The ODMG Java binding was recently updated to support the expanded collection classes used to model relationships between objects (one-to-one, one-to-many, and many-to-many) in Sun's Java Development Kit (JDK) Version 1.2. The updated Java binding chapter is available in the latest reprint of the standard.

C++ Language Binding

The ODMG C++ binding provides language-transparent extensions that provide facilities for object creation, naming, manipulation and deletion, and transactions and database operations. The C++ binding allows persistence-capable classes to be created by inheritance. OQL and OML can be interspersed, so that OQL queries may be built in-line to application code and conversely, OQL can call OML. The binding also includes recent INCITS C++ enhancements, like support for Standard Template Library (STL) algorithms (to provide sequential access to members of a collection) and exception handling.

Smalltalk Language Binding

The ODMG Smalltalk binding provides the ability to store, retrieve and modify persistent objects in Smalltalk. The binding includes a mechanism to invoke OQL and procedures for transactions and operations on databases. As in the Java binding, Smalltalk objects are made persistent by reachability. This means that an object becomes persistent when it is referenced by another persistent object in the database, and it is garbage-collected when it is no longer reachable. The Smalltalk binding directly maps all of the classes defined in the ODMG object model to Smalltalk classes. Object Model collection classes and operations are mapped wherever possible to standard collection classes and methods. Relationships, transactions and database operations are also mapped to Smalltalk constructs.

ODMG Object Storage in the OMG/CORBA Environment

ORBs and object storage play different roles. In the OMG environment, an ODMG object storage supports definition, creation, and manipulation with the services of persistence, transactions, recovery, and concurrent sharing for application objects ranging from the smallest units (words in a word processor, cells or formula terms in a spreadsheet) to the largest (documents, systems). Many applications require these services to provide transparent distribution in a heterogeneous mixture of platforms and other services such as versioning and security, all from a single product. ODMG object storage is best thought of in terms of the services that it provides, and not according to any particular implementation. Radically different implementations are possible, each offering different levels of services and trade-offs.

In contrast, The ORB provides a larger-scale set of services across heterogeneous vendors and products. For example, it allows clients to use multiple data storage products. It provides behavior invocation or method dispatch. In contrast, a given ODMG object storage provides a single-vendor capability and only a specific set of services rather than an arbitrarily-defined set. However, those services include more complete capabilities of high-performance, fine-grained persistence that are used directly within applications to support millions of primitive objects. The ORB, when it needs

persistence services, could choose to implement them with an ODMG object storage, whose services may be invoked via the ORB.

The ODMG Organization

The ODMG consortium goals are to continue to evolve the ODMG standard, to educate the developer community about the standard, to promote its use and to provide certification of compliance to the standard. ODMG is enhancing the existing Java, C++ and Smalltalk language bindings and plans to develop additional object storage bindings in new market areas, like application servers, which could benefit from standardization. ODMG is also working to create tighter integration to OMG's CORBA architecture. The organization is recruiting tool vendors, consultants and other companies that can gain technology leverage from an object storage standard. The ODMG is also working towards the creation of certification test suites, particularly for the Java binding.

ODMG Membership

The ODMG membership continues to grow. Since its inception in 1991, the ODMG has continued to grow and today includes database vendors, tool vendors, consulting firms, and corporate end users. The ODMG membership consists of voting, reviewer and associate members. Voting membership requires that a company commits 20 percent of the time of a senior technical representative to ODMG activities and uses the ODMG standard. Reviewer membership requires a company to commit 10 percent of the time of a senior technical representative, but does not confer voting privileges. Associate membership allows companies that cannot commit to a high level of standards development participation to track to the certification development process.

The voting member companies are Ardent Software, IBEX Computing SA, Lockheed Martin, Object Design, Objectivity, POET Software and Versant Object Technology. Reviewer member companies are Andersen Consulting, Baan, CERN, Computer Associates, GemStone Systems, Hitachi, JavaSoft, Microsoft, NEC Corporation, Sybase, and Telenor R&D.

For more information about the ODMG, visit its web site at www.odmg.org. ODMG 2.0 (ISBN 1-55860-463-4), edited by R. G. G. Cattell and Douglas K. Barry, is published by Morgan Kaufmann Publishers.

Douglas Barry has worked in database technology for more than 20 years, with an exclusive focus on the application of database technology for objects since 1987. He is the author of *The Object Database Handbook* (John Wiley & Sons, 1996), *The DBMS Needs Assessment for Objects* (Barry & Associates, 1998), and *The ODBMS Implementation Stories* (Barry & Associates, 1996). He also serves as the executive director of the ODMG. You can email him at dkbarry@odbmsfacts.com. His web site is www.odbmsfacts.com.