ODMG 93 - The Emerging Object Database Standard

Dirk Bartels, POET Software Corporation

The Object Database Management Group (ODMG) is a consortium of the leading Object Database (ODMBS) vendors. The consortium was formed in 1992 with the objective to define a standard for the emerging ODBMS industry. Within 18 months, the first release of the standard, so called ODMG-93 was published in October 1993. The following abstract gives a comprehensive overview of the standard and the extension that have been made since the initial publishing. The overview includes the ODMG Object Model, the ODMG Object Definition Language (ODL), the ODMG Object Query Language (OQL), the ODMG C++ binding and the ODMG Smalltalk binding.

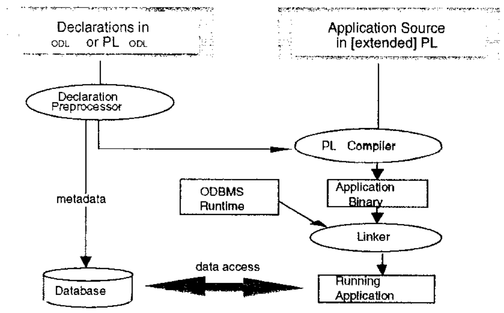
History

Back in 1991, ODBMS companies struggled to get market acceptance. One of the major concerns of corporate customers at that time was that unlike for relational databases (RDBMS), it didn't exist a standard for objects in databases. All attempts to create an ODBMS standard, e.g. based on a product of a dominating vendor or based on the work of other related groups such as the Object Management Group (OMG), had failed. Finally the initiative of Rick Cattell from Sun Microsystems brought all major ODBMS vendors to the table and the formation of the Object Database Management Group (ODMG) took place. The charter of the ODMG is to define and to establish an industry standard for object databases that would eventually do what SQL and the SQL Access Group did for the RDBMS world. Within only 18 months of intensive work, the first release of the standard was published, a major achievement given the facts that most of the participants represented at that time were fairly small companies. Since that date, the participating vendors have implemented the standard or parts of it. The feedback from the implementors as well as feedback from various groups, including databases researchers, reviewing members from the user community and technical staff members of the ODMG has been used to create subsequent releases. Release 1.1, published in 1994, had its focus on inconsistencies between the different sections of the book, that had been over looked in the first release. Release 1.2, published at the end of 1995 addresses a number of important issues, e.g. a much stronger Smalltalk binding, a higher level of compatibility to the SQL 92 query language and aspects related to the new ANSI C++ standard.

The ODMG is committed to continue to improve the standard. Many interesting technical specifications are well under way to be added to the standard. Today, ODMG is the only source for a reliable basis to develop open ODBMS applications. Its voting members represent probably more than 90 % of the existing commercial ODBMS market and all vendors are shipping standard compliant products and components as you read this abstract.

The ODMG Architecture

ODMG's approach to provide an interface to an ODBMS is based on the architecture of a tight binding to an existing object oriented programming language, such as C++ or Smalltalk. Unlike SQL, which provides a Data Definition Language (DDL) and a Data Manipulation Language (DML) for special purpose, the ODMG standard leverages existing computational complete OO host languages and adds value by providing database centric functionality such as an ad hoc query language. The following picture illustrates the architecture of the ODMG standard. A "Declaration Pre-Processor" parses an ODMG ODL (Object Definition Language) script or a programming language specific ODL. This pre-processor builds and maintains the database



schema (meta data). The programs that manipulates and queries the database is written in a standard OO programming language, with C++ and Smalltalk

1063-6382/96 $5.00 © 1996 IEEE

bindings already described in the standard. The ODBMS runtime is conceptually linked into the application binary. Real implementation might consider slightly different architectures, e.g. client/server systems.

The ODMG Object Model

The ODMG standard is based on a common object model that will not surprise anyone. Actually, the Object Model is based in many aspects on OMG's Object Model. It does support types (interface) and classes (implementation), encapsulation, inheritance and polymorphism.

However, a few database specific features have been added to fit the requirements:

A *Database* stores objects, enabling them to be shared by multiple users and applications. A database is based on a schema that is defined in ODL and contains instances of the types defined by its schema.

A *Relationship* is a property of an object. Relationships can be defined to one or more objects.

An *Extent* of a type represents all instances of this type in a particular database.

A *Key* uniquely identifies an instance of a type. Simple and compound keys are supported.

Object Identifiers (OID) are unique within a storage domain. The value of an OID never change over the lifetime of an object and is never re-used.

An *Object Name* can optionally be applied to an object. Unlike keys, an object name is not a property of a type.

The *Object Lifetime* differentiates between "transient" and "persistent" objects.

A number of predefined *Collection Types* are available to build complex objects. This includes Set<t>, Bag<t>, List<t> and Array<t> collections.

The *Atomic Literals* Long, Short, Unsigned Long, Unsigned Short, Float, Double, Boolean, Octet, Char, String and Enum are supported.

The *Structured Literals* Date, Time, Timestamp and Interval are also supported.

The *Transaction Model* guarantees the basic ACID model (atomicity, consistency, isolation, durability). Transactions can be nested.

The Object Definition Language (ODL)

The Object Definition Language is a pure specification language, independent of any programming language. ODL is a strict superset of OMG's Interface Definition Language (IDL). It adds the notion of relationships, collections, extents and keys to IDL to map all existing features of the ODMG Object Model. Since ODL is a specification language only, it is not designed to express any implementation detail. The following is an example of an ODL declaration:

```
interface Professor: Person
(
        extent professors
        keys faculty_id, soc_sec_no[10]): persistent
{
// attributes
        attribute String name;
        attribute Unsigned Short faculty_id[6];
        attribute Long soc_sec_no[10];
        attribute Address address;
        attribute Set<String> degrees;
// relationships
        relationship Set<Student> advises
                        inverse Student::advisor;
        relationship Department department
                        inverse Department::faculty;
// operations
        Short grant_tenure ()
                        raises (ineligible_for_tenure);
};
```

The Object Query Language (OQL)

The Object Query Language (OQL) has been designed as an ad hoc, non procedural language. Unlike SQL that tries to cover data definition (DDL) and data manipulation (DML) as well, OQL really serves only the query purpose and is intentionally computational incomplete.

Over the course of the ODMG standardization process, OQL has been modified to meet as much SQL compatibility as possible. As a result, OQL today is a superset of the standard SQL that deals with database queries. Any select SQL sentence which runs on relational tables, works with the same semantics on collections of ODMG objects. The OQL extensions concern object oriented notions, like complex objects, object identity, path expression, polymorphism, operation invocation etc. This makes OQL far superior than SQL but yet simple to use.

Typically, OQL retrieves objects instead of plain and flat data. Objects keep their identity and can be used for subsequent queries and navigation.

The ODMG C++ Binding

The C++ binding consists of a language specific C++ ODL and a C++ class library that provides the database functionality to the C++ programmer. The concept of the ODMG C++ ODL is based on the idea of a single type system between the programming language and the database language. The C++ type system is used as the specification language for the database. All features of the ODMG Object Model, especially relationships, collections, atomic literals and structured literals are available as C++ classes. Mapping the ODL example into a C++ ODL results into the following code:

```
// define symbols for relationship declarations
extern const char _faculty[];
extern const char _advisor [];

class Professor: public d_Object
// extents and keys are not directly supported by the C++
// ODL and must be maintained by the programmer
// (
//          extent professors
//          keys faculty_id, soc_sec_no[10]): persistent
// {
// attributes
private:
          d_String name;
          d_UShort faculty_id[6];
          d_Long soc_sec_no[10];
          Address address;
          d_Set<String> degrees;
// relationships
          d_Rel_Set<Student, _advisor> advises;
          d_Rel_Ref Department, _faculty> dept;
// operations
          d_Short grant_tenure ();
};
```

The interesting issues in the C++ ODL are probably the mapping of the relationships into the d_Ref<T> template class and the collection classes. The binding comes with a couple of collection classes as described in the Object Model. The d_Iterator class has been enhanced with STL compliant functionality to provide a higher integration with STL template functions.

The database manipulation from C++ is based on a class library. Persistent capable classes are derived from the base class d_Object. Persistence is achieved at construction time via a special new() operator. Changes to persistent objects are automatically written to disk at transaction commit time. Objects are loaded into memory via navigation or by performing a database query. The C++ binding offers an interface to OQL. The C++ binding proposes a pessimistic concurrency control with automatic read locks.

The ODMG Smalltalk Binding

The ODMG Smalltalk binding is based upon two principles: it should bind to Smalltalk in a natural way which is consistent with the principles of the language, and it should support language interoperability consistent with the ODL specifications and semantics.

There are a number of aspects that are different between ODL and Smalltalk that are based on the different approaches. For the database declaration, the Smalltalk binding is based on ODL. However, multiple inheritance is of course not supported. The collection classes are mapped to existing Smalltalk collection classes with a number of additions to the protocol to achieve compatibility with the ODMG Object Model.

The notion of persistence is determined by reachability. The Smalltalk programmer can define database root objects. Other persistent objects need to be related to these root objects to become persistent. This also implies that explicit deletion of objects is not supported. Persistent objects that are no longer reachable will be deleted via garbage collection.

Relationships are defined as a set of accessor operations for adding or removing associations between objects. The Smalltalk binding for relationships results in public methods to from and drop members from the relationship. The following code example illustrates this:

```
// ODL declaration fragment
     relationship Department department
                         inverse Department::faculty;
// Smalltalk mapping
     formDepartment: aDepartment
     dropDepartment: aDepartment
     Department
```

The Smalltalk binding offers also an interface to OQL. OQL queries can be invoked via a query class. The result is typically a collection that can be iterated.

Other database functionality, e.g. transactions and database operations, is presented in a Session class.

Conclusion

This abstract gave a very brief overview about the different chapters of the Object Database Standard ODMG-93 rel 1.2. This standard is probably the most consistent document available and very well received from the ODBMS vendor community as well as the user community.

The Object Database Standard: ODMG-93, Release 1.2, Edited by R.G.G. Cattell, Morgan Kaufmann Publishers, San Francisco, California