

Team:

## Position Paper

Yevgeniy Boyko

Simon Galperin (team lead)

Alexander Stepanov

### Should we use Object Database Management Systems?

Object-oriented database-management systems have place in development of applications with specific requirements. In particular, these applications might store and manipulate data in classes defining multiple cross-references among themselves, classes which organize data as deep tree structures or applications which constantly evolve and data structures they supports must evolve as well.

Many industry professionals, however, written off the object oriented database technology as a failure. They usually state several reasons why ODBMSs didn't rise to the industry expectations:

- There is no standard for object oriented databases. "Despite all these activities, currently there is no standard for object-oriented databases; that is, there is no standard object-oriented database language in which to program applications" (Won Kim, Page 1)

- Object databases lack a formal mathematical foundation which leads to weaknesses in their query support "There is no universally agreed data model for an OODBMS, and most models lack a theoretical foundation." (Thomas Connolly, Carolyn Begg, Page 883)

- ODBMSs usage is still relatively limited in the industry and it is hard to find qualified professionals to work on such systems, as well as it is hard to find third party tools for these environments "In comparison to RDBMSs, the use of OODBMSs is still relatively limited. This means that we do not yet have the level of experience that we have with traditional systems" (Thomas Connolly, Carolyn Begg, Page 884).

- ODBMS systems don't support arbitrary queries well.

- ODBMSs yield in performance and scalability to relational database systems. "The main drawback of OODBMSs has been poor performance. Unlike RDBMSs, query optimization for OODBMSs is highly complex. OODBMSs also suffer from problems of scalability, and are unable to support large-scale systems." [4]

We will try to rebut above-mentioned arguments and show that when satisfying certain business requirements, ODBMS will outperform relational database management systems and outweigh the disadvantages this solution is associated with.

As with any technology, ODBMS has its advantages and disadvantages. It is not a big secret that ODBMSs didn't become widespread and that this technology is experiencing rough times at the moment. However, we believe that ODBMS "winter season" is eventually going to pass and its market continue to grow because the answer to the question why object database products find themselves in such a state lies not in the area related to ODBMS technology itself, but due to the fact that vast amounts of data are still stored in relational databases and companies are forced to use object-relational mapping solutions in their effort of eliminating impedance mismatch.

#### ODBMS standards

First of all, some of the criticisms towards object databases are simply based on out-dated information about ODBMSs. The Object Data Management Group has proposed a standard known as ODMG-93, revised into ODMG 2.0 and later to ODMG 3.0. The standard consists of:

- object model
- object specification languages (Object Definition Language (ODL) and Object Interchange Format (OIF))
- object query language (OQL)

- and bindings to OO programming languages

The OMG core model was designed to be a common denominator for object request brokers, object database systems, object programming languages, and other applications.

ODL is a specification language used to define the object types that conform to the ODMG Object Model and is based on the OMG IDL.

OIF is a specification language used to dump and load from a file or set of files.

OQL is a declarative (nonprocedural) language for querying and updating objects. The primary interface in an OODBMS for creating and modifying objects is directly via the object language (C++, Java, etc.) using the native language syntax. [3]

Argument could be made that products available today implement very little of ODMG 3.0 standard. We would like to mention that this criticism can be leveled at both RDBMS and ODBMS products. Moving an application from one Relational DBMS (RDBMS) to another is a significant effort because of uneven implementation of SQL and proprietary SQL extensions among RDBMS products. It is true however that vendors don't generally have big interest in ODL or OQL because, in first case ODBMS products provide processing of class definitions for defining a database by using Java or C++ class definitions, and in second case because most vendors already had a SQL-like query language.

There are several organization that are working on the standardization of the object oriented databases.

### **Lack of a mathematical foundation**

The mathematical foundation of relational theory has been quite useful academically for proving theorems, but in practice it is only useful for applications with simple, tabular data structures and simple operations. For these applications, the SQL query mechanism applies directly and the optimizers directly apply to the user's problem. However, in most cases the application's data structures are not simple tables... Complexity, whatever it is, is inherent to the application. With an RDBMS, all that complexity is left to the application programmer [5].

Also, the weaknesses in query support could be offset by the fact that some ODBMSs fully support SQL in addition to navigational access. Example of such database is Objectivity/SQL++.

### **ODBMSs are relatively new technology**

It is true that object database model is relatively new and there are not as many people who know how to manage ODBMSs compared to relational databases. It is possible to argue though, that it is hard to find highly qualified professionals in any area of expertise, be it Matisse or Oracle database development and administration. As far as application programmers are concerned, the primary training that they need is the object programming language they are using, simply because of transparent persistence ODBMSs offer. Transparent persistence refers to the ability to directly manipulate data stored in a database using an object programming language except for a few additional commands. These commands are used for opening databases, starting transactions, issuing queries, ending transactions, and closing databases. At the same time, ODMG Binding shields programmers from internal models and provides an interface to the data they manage that is much like the interface provided by SQL.

There are solutions which allow for transparent persistence as well. One of them is to use a resolver that maps the transient objects onto the physical database [7]. This approach has its benefits accommodating future technologies and allowing companies to use legacy systems and protecting huge investments. Arguably, this additional layer adds complexity and deteriorated performance. Studies show that up to 80% of the lifetime cost of applications is in maintenance, during which time features are added, bugs are fixed, capabilities changed, all of which require changes to the data model underneath. When the application changes its data model, this tends to amplify, through the

mapping layer, into many changes, both in that mapping layer and in the flat tabular data underneath [5]. Christof Wittig, the founding CEO of db4objects called it “band aid for a problem” [8].

The lack of tools for ODBMS environment is in fact a negative aspect of using ODBMS products. Tools such data aggregation, report writers, OLAP tools and so forth are not readily available and have to be custom written which consumes developer resources that could be used more productively elsewhere. However, due to the almost total integration between “memory object space” and “persistence object space”, developing these tools is quick and easy [6].

### **Arbitrary query support**

RDBMSs are well designed to deal with traditional data such as numbers and character strings, and all SQL products are able to formulate complex queries and apply them to a database after primary implementation. On other hand, some ODBMSs don't support these sorts of queries. If it is a case, it involves some prescience to resolve this issue by introducing indexed collections of objects when an application is first designed. These collections reference objects of interest for this particular collection according to needs of a business. In other words, these collections help in navigating to a root object that has the desired target in its object tree [6]. Also, businesses have to articulate these needs in advance to accomplish this task, which admittedly might be very difficult to do.

Dealing with ad hoc queries is much simpler in RDBMSs or ODBMSs which support SQL. The problem gets solved by building a new query with appropriate series of joins.

### **Performance and scalability**

There is an opinion among many application developers that ODBMSs suffer from poor performance. However, experiences of some businesses, which use object database systems, contradict with this common opinion. In fact, significant performance gains over other persistence mechanisms are reported [6]. For example, Cimplex Corporation reported that Objectivity/DB performed 1000x faster than the leading RDBMS [5]. This superiority of ODBMS products when dealing with complex data could be explained quite easily. The more complex the data structure, the more time it takes to disassemble and reassemble it from its primitive components. Any nested structures or varying-sized structures require multiple RDBMS tables, and joins between them. Similarly, any relationships require joins. The join, although it's quite flexible, is the weak point of relational technology because it's slow. It requires a search-and-compare operation, which takes time, and gets slower as the database tables get larger [5]. In contrast, ODBMSs do not need to do the slow search-and-compare of the joins. They understand such structures and are optimized to efficiently manage them.

Numerous examples of using ODBMSs prove that these systems are very scalable as well.

CERN is the leading worldwide consortium, headquartered in Geneva, Switzerland, for performing scientific experiments at the sub-particle level. For their new generation accelerator, they need to build the largest database ever built, which is expected to grow to 100 PB = 100,000 TB = 100,000,000 GB =  $10^{17}$  Bytes. After evaluating all other available DBMS technology, both RDBMS and ODBMS, they concluded that the only system scalable to this level is Objectivity/DB [5].

Stanford Linear Accelerator Center (SLAC) conducts its flagship research project named B Factory. As of November, 2003, the SLAC stored over 800 terabytes of production data using Objectivity/DB. The production data is distributed across several hundred processing nodes and over 100 on-line servers [9].

BBN, the creators of the Internet, have deployed a Crisis Management and Resource Logistics Management system built on Objectivity/DB. The top three requirements that led to the choice of the ODBMS are scalability, flexibility, and survivability. The first is due to the need to support massive operations, without the worry that a central server will overload. Instead, they can simple add more servers as needed [5].

## **Other considerations**

There is no distinction between an “in-memory” objects and “on-disk” objects in an environment that uses an ODBMS. One can say that an object that refers to other object “contains” the object it references. In systems that use RDBMS, object often contains a key to the referenced object, i.e. the referenced object is not stored as an object. Thus, working with ODBMSs is more natural for a developer since objects are always objects in ODBMS model.

Some inadvertent operations on objects result to dangling references in a database. Garbage Collections eliminate this problem in ODBMS. Persistent objects are “deleted” once they are no longer referenced by any other instances reachable from a well-defined root object [6].

Businesses employing ODBMS products report that they gain significant advantage over their competitors in that they are able to deploy their product more quickly. For example, Nouseft – a company, which specializes in Library Document Management, built its new system on Objectivity/DB. The same implementers built the new system in 1/4 the time of the old. Although the implementers had the benefit of experience, they also implemented significantly more functionality, and credited the 1/4 savings to object technology and the ODBMS [5].

## **Conclusion**

We must point out that ODBMSs are not a panacea and must be used in certain situation. For example, it is unlikely that an ODBMS should be utilized for such application as billing. The key is to use the right tool for each job. For example, if the business problem can be solved with the relational database without a lot of effort and the performance requirements could be met then there is no reasons to use object oriented database. However, if the database will have many to many or tree structure relationship, large scalability (very large number of clients or very large data transactions), complex data like multi dimensional arrays and binary streams (nested or complex data structures), distributed deployment architecture (which involves the use of many databases) then object oriented database would be the right choice.

We strongly support ODBMS and promote future research and development of object oriented databases. The object oriented databases are much easier to learn than the relational databases and also much cheaper to own. Therefore, the object oriented approach is more natural alternative for new growing companies that do not have resources to purchase relational databases and to provide intensive training to their staff. ODBMS are also the perfect choice for the scientific communities and university research. In fact, over the past decade several well known organizations researched and worked on the new standards for object oriented databases. There are many organizations, including Open Software Foundation that support standard for object oriented database, so developers could better understand and manipulate the data output.

## References

- 1) Won Kim, "Research Directions in Object-Oriented Database Systems", April 1990, Publisher: ACM Press
- 2) Thomas Connolly, Carolyn Begg, "Database Systems: A Practical Approach to Design, Implementation, and Management", 2005
- 3) ODMG 3.0 [http://www.service-architecture.com/database/articles/odmg\\_3\\_0.html](http://www.service-architecture.com/database/articles/odmg_3_0.html)
- 4) Ramakanth S. Devarakonda, "Object-Relational Database Systems - The Road Ahead" <http://www.acm.org.ezproxy.library.drexel.edu/crossroads/xrds7-3/ordbms.html>
- 5) Objectivity, "Object Oriented Database vs Relational Database" <http://www.objectivity.com/pages/object-oriented-database-vs-relational-database/performance.html>
- 6) Vincent Coetzee, "Experiences using an ODBMS for a high-volume internet banking system", Pages: 334 - 338 2003, Publisher: ACM Press <http://portal.acm.org.ezproxy.library.drexel.edu/citation.cfm?id=949439&coll=ACM&dl=ACM&CFID=2978956&CFOKEN=15455306&ret=1#Fulltext>
- 7) Jen-Yao Chung, "Object and relational databases", Pages: 164 – 169, 1995, Publisher: ACM Press <http://portal.acm.org.ezproxy.library.drexel.edu/citation.cfm?id=260273&coll=ACM&dl=ACM&CFID=2978956&CFOKEN=15455306>
- 8) International Conference on Object Programming Systems, Languages, and Applications 2006. Panel "Objects and Databases, State of the Union 2006" <http://www.odbms.org/download/010.04%20Cook%20OOPSLA%20Panel%20Objects%20and%20Databases%20October%202006.PDF>
- 9) Objectivity Customer Stanford Linear Accelerator Center receives grand prize in winter corporation top ten program [http://www.objectivity.com/pdf/040112\\_SLAC\\_Winter.pdf](http://www.objectivity.com/pdf/040112_SLAC_Winter.pdf)