A New Renaissance for ODBMSs?

Rick Cattell, Consultant.
Robert Greene, Vice President, Versant.
Leon Guzenda, CTO, Objectivity.
Luis Ramos, Principal Systems Engineer/Consultant, Progress Software.
Carl Rosenberger, Co-founder, db4objects.

July 27, 2009 -- *This is a follow up of the panel discussion, which took place at the ICOODB 09 conference on July 2, 2009 in Zurich. I have asked each of the panelists to give written reply to some of the questions that were asked at the panel. I also added here a few new questions.*
Roberto V. Zicari, Editor ODBMS.ORG

Q1. Could you estimate the current market for ODBMS, how is it changing (size, geography, licenses, dollars, etc.), and whether/where the market is growing?

Rick Cattell:
The object database market has continued to grow and mature. There has been some consolidation, which is to be expected given the number of vendors that were trying to share the market, but as a result all of the vendors are now profitable; it seems that most markets can support only 3 major vendors.

The market has been quite persistent. I was surprised myself to see significant revenue figures for the ODBMS market after years of nay-sayers who compared RDBMS and ODBMS revenues and extrapolated impending doom for the latter. But comparing ODBMSs to RDBMSs is like comparing the market for pickup trucks to the market for passenger cars; they are for different purposes and markets. It is now generally accepted that "one size does not fit all". ODBMSs are being used in applications where RDBMSs simply don't work.

Robert Greene:
I would put it somewhere in the 150M-200M range considering the increasing number of vendors in this space. For Versant, business is fairly equally spread between Europe and U.S. with Asia Pacific making a sizable contribution. We also see an increasing potential for the China market. At Versant, due to the changing dynamics in application requirements, we believe it is reasonable to assume we can capture 5% of the 10B+ database market in the coming decade.

Leon Guzenda:
Only one of the three largest ODBMS companies publishes revenue figures for their products and services, but if they all have roughly equal shares of the market the total annual revenues would be in the $70M to $80M range. Objectivity has just reported record growth for the year and has done so for most of this century. Versant has also been reporting good growth. My impression is that the ODBMS companies are still making money from new systems embedded in equipment but that deployment in complex IT systems is growing faster.

Luis Ramos:
Progress ObjectStore revenues have continued to beat our expectations year in and year out. It has been one of our most profitable products in the company. We continue to see adoption of object database technologies as people search for ways to increase performance, lower their Total Cost of Ownership (TCO), and shorten their time-to-market.

One new area that we see potential growth in is in cloud-databases. Object-oriented databases are a more natural fit for persisting cloud data for a number of reasons. First, cloud-databases are inherently tuple-based and storing tuple-based data, whose values are arbitrary (strings, integers, double, boolean, etc) could be challenging to do in a relational database. This type of problem is reminiscent of the issues related to modeling a relational model to support an object model that has an inheritance hierarchy. Second, scaling a relational database is challenging. The usual practice to scale a relational database in order to support more load is to use more

powerful hardware. In an object-oriented database such as ObjectStore, it is easier and simpler.
Since the queries are performed at the client (the cloud node or service), scaling the database can be accomplished by simply launching more services.

Carl Rosenberger:
The total ODBMS market size is somewhere between 50 and 200 million dollars. It depends which products are classified as ODBMS. The market seems to be pretty stable. ODBMS companies are profitable.

Q2. *Who is using an ODBMS? For what business domains and applications are ODBMSs being practically used? Could you please give examples of typical applications that use an ODBMS? Also please indicate whether/where ODBMS applications domains are changing.*

Rick Cattell:
ODBMSs are being used in a wide variety of applications where they provide the best solution. They are used in embedded or real-time applications where their performance or rich data model are needed. They are used in web servers for caching or as an alternative to O/R mapping, using JDO or JPA with extensions, where there is no existing SQL database or where an ODBMS's in-memory performance is needed for some of the data. They are used in engineering, scientific, or business applications where an ODBMS's speed at reference-following is needed, e.g., expanding a design's
subcomponents or a large bill of materials.

As the ODBMSs have matured, they have become another "off the shelf" data management option for application developers, along with traditional RDBMSs, text search engines, O/R mapping, indexed files, and other tools.

Robert Greene:
OODBMS are being used across the board in most vertical markets. They end up in the more difficult applications where models are increasingly complex with deep graphs, many to

many and recursive relationships. There are certain areas where you find more of these types of applications than others i.e. Telecommunications and Defense. However, really they are showing up everywhere for scalability, ease of use and performance reasons. In Telecom our customers use us in every aspect of business including: network management, operational support systems, billing, softswitch, content management, provisioning, and more. For Defense, it is also a broad spectrum including: simulation, battle management, intelligence analytics, communications, target tracking, planning, etc.

Our customers are using ODBMS in Finance for risk management, online trading, batch clearing, ticker systems. We are used in Transportation for planning, rail management, airline/car/hotel reservations, communications, logistics and more. Further, our customers use an ODBMS in areas such as rich media content management, work force tracking, home improvement planning, legal document processing, SCADA systems, online gaming, website store fronts, embedded in medical devices, on mobile phones, just so many areas all over the map. So, is the domain of OODB use changing, yes, it is growing in all visible directions.

Leon Guzenda:
The main markets appear to be: Defense and the Intelligence Community; telecom, medical and process control equipment; scientific research; online communities; front end caching of complex IT data; advanced financial systems and manufacturing. Each manufacturer's web site gives concrete examples of applications, ranging from military reconnaissance and targeting systems to space debris tracking and operating theater applications. There seems to have been a shift from engineering applications to complex IT and real-time systems over the past twenty years.

Luis Ramos:
ObjectStore is used in a very wide and diverse range of business domains and applications. Because persistence in ObjectStore is orthogonal to type, you can persist any class or data structure, without any restrictions. Consequently, customers and applications use it to define bespoke (custom)

indexes in order to optimize queries that cannot easily be supported by more traditional indexing (such has plain hash tables and B-trees). With ObjectStore, you have the flexibility to define and persist novel index data structures and algorithms that optimize the application's predominant use cases.

One example of a typical example is a risk management application for a very large financial company. Through its bespoke indexes, the application identifies risks in near real-time (as opposed to a day) and has helped save the company millions of dollars a year. It has proved to be a very important application for the company, most specially in this economic environment.

Carl Rosenberger:
There are applications that benefit so much from the use of ODBMS that they are the only wise choice. This is when either the object domain model is so complex that a mapping to a relational databases is unaffordable or when the performance requirements to store objects are so high that a relational database can't keep up with the speed.
For almost any application ODBMS can be used to improve development speed by avoiding O-R mapping.

Let me give you two examples of db4o use:
- Indra Sistemas caches real-time data of train systems in db4o. They need the performance to store objects.
- Ricoh uses db4o for it's next copier platform. They have low-resource constraints on their machines.

Q3. What are the most innovative features (if any) added/ or improved to/for ODBMS in the last years? And why are they useful?

Rick Cattell:
I would argue that the most important feature of ODBMSs in the past 5 years is the lack of lots of new features; instead the vendors have focused on reliability, performance, and usability. Also, I believe that the recent introduction of an open

source ODBMS (db4o) is a significant development: a much larger audience is now considering an ODBMS option as a result.

Robert Greene:

Scalability and performance are the core value areas of the OODB. So, we have had tremendous focus on the internals of Versant in order to deliver performance and scalability on N-Core processors. It is hard to describe all of the work in detail, but it can best be described as introducing parallel algorithmic processing in the core kernel. Additionally, we made improvements in areas such as online reorganization, which is an area known to bring most relational databases offline in order to address. The architecture of Versant is uniquely suited to move objects around physically so that slow performance due to fragmentation is completely eliminated. Another of the most improved areas have been with the query execution engine. There was a lot of work to add advanced index types and provide set based query operators, aggregations, ordered results, etc.

These improvements make it easier to support existing tooling and provide a means of relational minded personnel to get information from the database.

As an extension of the query work, innovative additions have been in the area of .NET and support for LINQ as a native query language. Architecturally, we keep moving closer to the cluster database concept where the application is completely agnostic to the fact that the database is physically distributed over potentially 1000's of machines and the configuration can be dynamically changed while applications are online.

Leon Guzenda:

Most of the ODBMSs have improved their query capabilities in the past few years. Versant added JDO. Objectivity/DB added a distributed, parallel query engine and db4objects and Objectivity/DB have added LINQ. These facilities make it easier to access increasingly large datasets.

Luis Ramos:

Several features were added to ObjectStore to enable better support for our customers who use it as a cache for relational

data. ObjectStore was integrated with DataXtend Semantic Integrator (DXSI), one of the products in the Progress portfolio. DXSI provides a GUI-based OR mapping tool. The integration is useful because it enables and facilitates the caching of multiple heterogeneous relational databases.

Carl Rosenberger:
The use of LINQ as the querying language for object databases is a very innovative trend. At least on the .NET platform we now have an object-based querying standard that gets adopted.

We also see ODBMS being ported to constrained environments like Android, Silverlight, Powershell, Palm-Pre and iPhone. Because ODBMS store objects more efficiently than relational databases they make a lot of sense when there is little memory available and when the demand for performance is high.

Q4. Scalability, Performance, Support of OO languages, Impedance Mismatch: can you please describe briefly how in your opinion ODBMSs differ (compete) with respect to other databases, such as relational, and O-R mappers?

Rick Cattell:
ODBMSs and RDBMSs have better performance for different kinds of applications. For example, RDBMSs would dominate in simple transaction processing, while ODBMSs have significant advantage when a large portion of the database fits in memory, or in reference-expansion operations as I noted earlier.

O-R mappings are popular because they provide an ODBMS interface on top and RDBMS, which is convenient for programmers. However, they still have a significant performance disadvantage in the relational conversion, so an ODBMS is a better option when an underlying RDBMS is not required.

Robert Greene:
There could be a book written on this question, so hard to do it justice. I think the best way to put it is that with dramatic

increases in data and the number of concurrent users, systems such as relational and O-R Mappers which rely on relational, will not be able to scale when they are required to continually recalculate relationships. In such systems of scale, having a robust model where the relationships are baked in is the only way to achieve reasonable performance and scalability.
The ODBMS is a system which relies on robust models with predefined relationships and as such we compete for these higher end applications and win because there is just no way a system which has to continually recalculate relationships in the face of growing data can compete. This is why some of our customers do "benchmarking" proof of concepts and run into use cases where the relational database cannot even complete the operation.

Of course there is the whole impedance mismatch issue in the application space, but the fundamental difference is, OODBs predefine a model and bake relationships into the database and RDBMs predefines a schema and uses that schema to recalculate (using set based operations) relationships.
We can see evidence of the above assertion on RDB scalability limitation outside of the ODBMS space by examining what people are doing to achieve scale with the larger
social networking, online shopping, entertainment and other type applications.

Many of these guys started on RDBMS technology because it is the incumbent technology and they know how to use it. When they need to scale, they eventually move to more of a map based model where identity begins to get managed in the application space. Some for example Facebook, LinkedIn and others stay with an RDB, dumb down the tables to a very simple schema and build an identity system into the application tier and essentially simulate OODB navigation.
Others move away from RDB and go to something like BigTable from Google or a distributed map technology like Tangosol where again the application is made aware of identity and code is put in place to deal with relationships at the key/value level.
The problem is that when you move to these new paradigms to deal with scalability you are loosing the ability to transparently work with object graphs and your code becomes

cluttered with special handling to manage the identity in the application layer. I think this is largely an education issue, many developers simply don't know they are making a poor man's OODB, sacrificing most of the advantages they once had with their RDBMS and achieving only 25% of the capability which can be found in a commercial ODBMS solution.

Leon Guzenda:
The largest differentiator is the complete elimination of the mapping layer between the object oriented language and the database. O-R mappers still have to cope with inheritance and many:many relationships with additional tables and join tables. In one recent benchmark the RDBMS storage overhead was 200% as compared with Objectivity/DBs 20%. Navigational access is much faster in ODBMSs than in RDBMSs because of the lack of join tables.

Luis Ramos:
Regarding scalability, object-databases have a fundamentally more scalable architecture compared to relational databases because the queries and processing are performed at the client side, where the cache resides, rather than the server side. RDBMSs have a more monolithic architecture. To scale it, you need to utilize a more powerful and expensive box for the database server. In contrast, to scale an OODBMS you can launch more clients (with their cache) and leverage as much Off the shelf hardware as you need.

Object databases differ significantly on how it stores data and how the data is queried and accessed. With relational database, data is fundamentally stored as tables of rows and columns. The data is tabular and two-dimensional. For example, consider the current roster for the Boston Red Sox Baseball team; the roster lists each player and their name, number, position and statistics--it looks like a rectangle with two dimensions. This formulation makes it easy for me to ask the question--what is the current batting average of the 1st baseman for the Red Sox?

Consider a slightly different question--how are individuals in a

Family tree related? Consider Barack Obama's family tree. If you visualize the data, it is not two-dimensional at all. It looks more like a tree of nodes. This would be very natural to store as objects with links to other objects. Consider asking a question like, is George W. Bush related to Barack Obama? Answering this question is quite easy in an object database. You simply follow pointers from the nodes representing Barack Obama and George Bush and see if you can reach a common ancestor node and how many levels up it is (apparently they're 10th cousins, once removed). Following pointers or de-referencing references is certainly a lot more efficient than doing an arbitrary number of joins.

However, instead of being used as an alternative to a relational database, ObjectStore is used in conjunction with a relational database as a cache to optimize performance. Our clients have selected ObjectStore over other data caching technologies such as Memcache and Tangosol because they could not provide one or more of these capabilities: transactional access, durability, and automatic cache replacement. With object databases, there's transactional access to the cache thereby preserving its data integrity--you don't find this in many caching solutions. Also, the cache is durable. If the application is terminated intentionally or otherwise, recreating the cache is fast and efficient because it is populated directly from an object database.
In this case, there is no overhead for running SQL queries to find to the objects to bring into the cache and no overhead to transform between relational data and objects. Third, ObjectStore automatically manages the cache if the amount of data being accessed in the cache exceeds the amount of memory.

Carl Rosenberger:
Object databases "understand" objects and store them in the same manner as they are held in RAM. Avoiding the overhead for relations and joins makes storage much more efficient and results in a performance gain. A more direct integration with the object model also yields better capabilities to handle all kinds of concurrency problems. Less resources are used, which

makes object databases ideal for devices like mobile phones and handhelds.

Q5. *Standards: Why have standardization activities for ODBMSs not progressed?*

Rick Cattell:
ODBMS Java binding and ODBMS query language standardization have continued to progress through JDO, JPA, and LINQ. -As far as the rest of the ODMG standard is concerned (the object model and the Smalltalk and C++ bindings), I believe standardization has not progressed further because the vendors believe we are now adequately "done," as far as is a priority at this time.

Robert Greene:
They don't progress because it does not make good business sense to the vendors. Further, adopting standardization now would not be following the natural process. The process is: innovation, adoption, consolidation, standardization. This is how it has happened in every industry since the industrial revolution. While there has been limited consolidation out of necessity, there has not been sufficient adoption and therefore no business justification for standardization (i.e. it will not help the vendors be more successful to have a standard for such a limited number of users). Unlike innovation, standardization itself does not lead to adoption.

Of course, even in the RDBMs space, there is no true standardization. You cannot unplug an application using Oracle and expect it to work with Sybase or MySQL. Probably more importantly, object persistence has moved past the ODBMS space into the RDBMS space via ORM and manifested itself as a new problem space know as Persistent Object Lifecycle Management. I think this is where Craig Russell with JDO has revolutionized the persistence field and should be recognized for his contribution over time. It was due to efforts such as JDO, to examine cross cutting concerns of object persistence, independent of the storage mechanism, which has lead to a

broader understanding of transactional requirements of persistent objects. If the OODBMS vendors can ever take advantage of a standardization initiative, it should be to get onboard with what is happening in this area of standardization which is also directly applicable to our space.

Leon Guzenda:
There isn't a large user community demanding standards beyond the ones that exist. Most of the ODBMSs also support SQL/ODBC too. However, we did make progress last year by committing to support for LINQ.

Luis Ramos:
Progress invested in providing a JDO API for ObjectStore. We were one of the first vendors to support this standard. Based on our experience, however, we have not really seen a strong adoption of this standard in the market place. More recently, there have been several other emerging API standards and frameworks including JSR-220, JPA, and Hibernate. These still appear to be in a state of flux. We continue to weigh in on which standards to invest in that would provide the best returns.

Carl Rosenberger:
Standards rarely evolve from niche products if the majority of the market chooses to do things differently. If we want a standard for querying object databases, I think it has to be capable to query relational databases as well. For .NET we now have a strong standard with LINQ and it is a great fit for object databases. If we want a standard for Java as well, I think it would have to be like LINQ: A query language integrated with the programming language.

Q6. *Open source: Some Open Source databases that have been successful in their respective domains, have been recently sold. For example MySQL sold to SUN and then to Oracle, and db4objects sold to Versant. Both Oracle and Versant do have proprietary products. Is Open Source a viable business model option for a database?*

Rick Cattell:
Open source is a viable business model. I regard the buy-out of MySQL as a demonstration of its success rather than failure. Look at MySQL's widespread adoption, not just revenue on the database itself. Widespread adoption can be used to drive revenue in other products and services, Sun and now Oracle can benefit from MySQL's ubiquity.

I believe that db4o will see substantial adoption, and that adoption will in turn drive revenue products as customers hit limitations of db4o in functionality and performance.

Robert Greene:
An open source database can clearly be a viable business model as evidenced by the success of MySQL. However, there are different measures of success. One measure is the success of adoption and another is the success of revenues and the corresponding contribution to economic sustainability.
While MySQL had great success in adoption, it had limited success in revenue generation in the face of that adoption (excluding the sale). If Oracle and Versant were to open source their complimentary commercial offerings, I am afraid there would be many thousands of people looking for a new job, because the revenues would simply not be there to support the engineers. So, it seems to me that the approach of both open source and commercial compliment is the right mix.

Leon Guzenda:
Red Hat proved that the Open Source model worked. MySQL showed that it works for DBMSs too. However, the model only seems to work well if there is a heavy emphasis on support and additional options and services. This is labor intensive and doesn't have the leverage that a standard licensing model does.

Luis Ramos:
I agree that Open Source is a viable business model. It is an excellent approach for gaining a significant developer base and market share. It offers revenue opportunities in the form of maintenance support, consulting, and training.

It can also provide opportunities to migrate to a higher end closed source software, when requirements are not met by the Open Source software.

Carl Rosenberger:
With db4o we still follow the open source business model and we are very happy that we see a quickly increasing number of users that use our product.