On NoSQL technologies - Part III: document stores, nosql databases, ODBMSs.

Editor Roberto V. Zicari
ODBMS.ORG
www.odbms.org

March 2010

Content

I was asked to compare & contrast odbms systems to the new nosql datastores out there. So I have asked several people in the last few weeks.... In this report, you`ll find a few more selected replies - I do not pretend this list to be complete, but hopefully it will help triggering a discussion.

...By the way, this list keeps growing so check the ODBMS.ORG *blog on a regular base for new updates: http://www.odbms.org/blog/*

Michael Stonebraker (MIT):
"Greene`s reply is perfectly reasonable. I think the "one size does not fit all" mantra -- which I have been espousing for some time -- is a good way to think about things. After all, the no SQL folks are themselves pretty diverse."

Hamid Pirahesh, (IBM Fellow):
"There is heavy activity in the Bay area in nonsql. There is VC $ going into those companies. They do extensive blogging.
There is also xml DBs, which goes beyond relational. Hybridization with relational turned out to be very useful. For example, DB2 has a huge investment in XML, and it is extensively published, and it is also made commercial. Monet DB did substantial work in that area early on as well."

Mårten Gustaf Mickos (previously CEO of MySQL AB):
" I think Kaj had a great response. (Link to Kaj response). Generally, in the early days of a new term, it doesn't in my mind make sense to try to define it exactly or narrowly. It's only a term, and it will take years before we know how significant it is and whether it is justified as a category of its own. For instance, it took a long time for Web2.0 to become an acknowledged term with a useful and reasonably well defined meaning."

Peter Norvig (Director of Research at Google Inc.):
"You should probably use what works for you and not worry about what people call it."

Dwight Merriman, (CEO of 10gen):
A comparison of document-oriented and object-oriented databases is fascinating as they are philosophically more different than one might at first expect. In both we have a somewhat standardized document/object representation -- typically JSON in currently popular document-oriented stores, perhaps ODL in ODBMS. The nice thing with JSON is that at least for web developers, JSON is already a technology they use and are familiar with. We are not adding something new for the web developer to learn.
In a document store, we really are thinking of "documents", not objects. Objects have methods, predefined schema, inheritance hierarchies.
These are not present in a document database; code is not part of the database. While some relationships between documents may exist, pointers between documents are deemphasized.
The document store does not persist "graphs" of objects -- it is not a graph database. (Graph databases/stores are another new NoSQL category - what is the different between a graph database and an ODBMS? An interesting question.)
Schema design is important in document databases. One doesn't think in terms of "persist what i work with in ram from my code". We still define a schema. This schema may vary from the internal "code schema" of the application. For example in the document-oriented database MongoDB, we have collections (analogous to a table) of JSON documents, and explicit declaration of indexes on specific fields for the collection.
We think this approach has some merits -- a decoupling of data and code. Code tends to change fast. Embedding is an important concept in document stores. It is much more common to nest data within documents than have references between documents. Why the deemphasis of relationships?
A couple reasons.
First, with arbitrary graphs of objects, it is difficult to process the graph from a client without many client/server turnarounds. Thus ,one might run code server-side. A goal with document databases is to maintain the client/server paradigm and keep code biased to the client (albeit with some exceptions such as map/reduce).
Second, a key goal in the "NoSQL" space is horizontal scalability. Arbitrary graphs of objects would be difficult to partition among servers in a guaranteed performant manner.

Eric Falsken (db4o):
"NoSQL database (like Google's BigTable data behind their Gears API) is an awkward sort of "almost-sql" or "sql-like".
But it ends up being a columnar-database. What you call a "document store" is a row-based database. Where records are stored together in a single clump, called the row. By eliminating strongly-typed columns, they can speed up i/o by many factors (data written to one place rather than many places) just in the insert/select operation. By intelligent use of indexes, they should be able to achieve some astounding benchmarks. The complexity of object relationships is their shared drawback. Being unable to handle things like inheritance and polymorphism is what stops them from becoming object databases. You can

think of db4o as a "document-oriented database" which has been extended to support object-oriented principles. (each level of inheritance is a "document" that all gets related together.)"

Dirk Bartels (Versant):
The "NoSQL" movement is a reflection of different and entirely new application requirements that are not orthogonal to SQL and relational databases. What makes this discussion really interesting in my opinion is that it has its roots with application developers.

The least an application developer wants to do is spending (or should I say wasting) time to implement a database system. Application developers want to concentrate on their domain, getting their apps out and compete in their markets. Today, pretty much all data persistence requirements are being implemented with SQL, in particular with several open source choices such as MySQL and PostgreSQL readily and at no cost available. SQL is what developers are learning in college, and therefore, it is today often the only database technology considered.

Now having these application developers stepping out of their own comfort zone, tossing the SQL databases and spending their precious resources inventing their own data management layer should tell us something.

From time to time, advances in compute infrastructure are disruptive. The PC, Client / Server, Internet, and lately Cloud Computing have been or will be catalysts for significant changes. Is it possible that "No SQL" means that certain new types of applications are simply not a good fit for the relational data model provided via SQL?

When taking a closer look, these applications still require typical database functionality also found in a SQL database, for example some query support, long term and reliable data storage, and scalability just to name a few. Nevertheless, there must be issues with SQL to make the pain just too big to stick with SQL. I haven`t done a lot of research on this subject, but suspect that the issues revolve around the data model (too rigid), the transaction processing model (too linear), the scalability model (horizontal scale out solutions too expensive), the programming model (too cumbersome) and probably more.

I remember when IT did`t get the PC, the Graphical User Interface, the Internet etc. I am not surprised that many traditional IT people are not getting it today. I expect the No SQL movement to gain momentum and to continue to evolve rapidly and likely in several directions. These days, applications and their data management requirements are significantly more complex. In my opinion it is just a matter of time that developers realize that the traditional relational model, invented for high volume transaction processing, may not be the best choice for application domains a SQL database is simply not designed for.

Is this a call for object databases, a long overlooked database technology that has matured over the past 20 somewhat years? I think it`s possible the future will tell. No SQL application developers should at least give object databases a good look, it may save them a lot of time and headaches down the road. "

Peter Neubauer (Neo Technology):
"We have not modeled an ODBMS on Neo4j yet, but if you look at e.g. the Ruby bindings , it fits very naturally into dynamic language paradigms, mapping the domain models almost directly onto nodes and relationships. We have written a couple of blogs on the topic, latest Emils classification of the NOSQL space,and there are approaches to turn Neo4j into something that resembles a OODBMS:
1. JRuby bindings , hiding Neo4j as the backend largely form the code, but still exposing the Traverser APIs baside the normal Ruby collection etc for deep graph traversals.
2. Jo4neo by Taylor Cowan , which is persisting objects via Java annotations.
3. Neo4j traversers, and Gremlin , for deep and fast graph traversals (really not OODBMS like, but VERY useful in todays data sets) .
It would be very interesting to have more conversation on these topics!"

Jan Lehnardt (CouchDB):
" For me, NoSQL is about choice. OODBs give users a choice. By that definition though, Excel is a NoSQL storage solution and I wouldn't support that idea :) I think, as usual, common sense needs to be applied. Prescriptive categorisation rarely helps but those who are in the business of categorising."

Floyd Marinescu (Chief Editor InfoQ):
"I think that web development has become so old and mature now that people are discovering that certain types of applications are better off with different solutions than your standard doctrine 3 tier system with SQL database. Plus the whole web services phenomenon has opened people`s minds to the notion of infrastructure as a service and that is driving this as well. This is my anthropological view of things. ;) "

Martin F. Kraft (Software Architect):
" NoSQL is a very interesting topic, though a standardized API like the W3C proposal would be challenging to adopt, and even more to outperform native legacy OODBMS (non-sql) queries.
I see the need to improve SQL performance in object oriented use as with J2EE and understand that some of the SQL performance implementations for OO like GemFire are doing great, but don't solve the underlying root-cause: the SQL overhead in non-sql data queries like object traversal and key/value lookup..
So far I would rather use RDBMS' and OODMBS' where they perform best, as Mr. Greene said "one size does not fit all"....
Some object databases provide (slow) SQL interfaces, and NoSQL should not mean no-QL"

Borislav Iordanov (HyperGraphDB):
"I think we've just realized that different representations are suitable for different domains and that it is possible to persist those representations natively without having to ultimately translate everything into a rigid schema. A very important driver of those NoSQL dbs. is their dynamic nature. I think many people like them for the same reason they like dynamic languages (no static typing, no long compile times etc.): change is easier, prototyping is faster, and getting it into production is not that risky. The problem with them of course is lack of integrity & consistency guarantees, something which ODBMs still try to provide at some level while remaining more flexible and offering richer structures than RDMBs. Another problem with RDBMS is SQL the language itself, which is very tightly coupled to the storage meta-model. Here again ODBMs do better through standartization, but more openness/flexibility etc. and come perhaps as a middle ground between anarchistic NoSQL and totalitarian SQL :) "

Miguel-Angel Sicilia (University of Alcalá):
"The NoSQL movement, according to Wikipedia today promotes "non-relational data stores that do not need a fixed schema". I do not believe ODBMS really fit with that view on data management. Also, other aspects of the NoSQL "philosophy" make ODBMS be far from them. However, NoSQL focuses on several problems of traditional relational databases for which ODBMS can be a good option, so that they can be considered to be "cousins" in some sense. I do not see NoSQL and ODBMS as overlapping, but as complementary solutions for non-traditional data management problems."

Manfred Jeusfeld (Tilburg University):
" I am a bit frightened by this development. Basically, people step back to the time prior to database systems. I heard similar ideas at a Dagstuhl seminar from IT experts of STATOIL (the Norwegian oil giant). They experienced that non-database data stores are much faster, and they appear to be willing to dump ACID for increased performance of some of their systems.
This is typical for a programmer's attitude. They want the data storage & access to be optimized for their application and do not care too much about interoperability and reliability. If every data items can be referenced by a 64bit memory address, why would we need a query language to access the data. Following memory pointers is certainly much faster. OODBs can serve as a bridge between the pure programming view and the database view."

Matthew Barker (Versant Corporation):
"As Robert (Greene) mentioned, when everyone thought "one size fits all", SQL evolved from a "simple query language" to a "do-it-all" tool including creating, modifying, and reorganizing data, not just querying. Many object databases such as Versant have an "SQL-like" query language but the capabilities are limited to actually querying the database, not updates, creates, deletes, etc. When you use SQL to modify data, you break the OO paradigm of safety and encapsulation; in a large application, it very easily becomes a monster that is difficult if not impossible to control. If we reign it and use SQL for it's original

purpose, querying data, then SQL can fit in nicely with object databases - but the monster it has become does not fit into object database technologies."

Kenneth Rugg: (Progress):
" I was at the "Boston Big Data Summit" a few weeks ago and the NoSQL movement came up as a topic. The moderator, Curt Monash whose is on dbms2.com, had a pretty funny quote on the topic. He said "The NoSQL movement is a lot like the Ron Paul campaign - it consists of people who are dissatisfied with the status quo, whose dissatisfaction has a lot to do with insufficient liberty and/or excessive expenditure, and who otherwise don't have a whole lot in common with each other."

Andy Riebs (Software Engineer at HP):
"Interesting blog! (Stonebraker sounds a bit too much like he's defending his baby) Greene's comments are sensible. Following through on the "one size doesn't fit all" theme, just how many simple databases are best implemented as flat text files? The "NoSQL" discussions are reminiscent of the old "RISC vs. CISC" arguments. While people usually understood the notion of a simpler instruction set, no one noticed that pipelining and huge register sets were introduced in the same package. Now you can find all those elements in most architectures. In the same sense, one presumes that many of the good ideas that survive the "NoSQL" debates will, in fact, end up in relational databases. Will that make the resulting products any more or less relational? Some problems will best be resolved with non-relational, non-SQL tools. Best bet for a NoSQL technology that will survive? Harvesting meaningful data from the log files of data centers with 20.000 servers! With a proper MapReduce implementation, it will be a thousand times more effective to distribute the processing to the source of the data, and return only pre-processed results to the consuming application, rather than hundreds of gigabytes of raw data. Of course, the real winner will be the one who can implement SQL on top of a MapReduce engine! "

Tobias Downer (MckoiDDB):
" Interesting read. I don't think you can really nail down exactly what a 'NoSQL' technology is. It's a rejection of the prevailing popular opinion that's been around for the last decade, which is that a data management system that doesn't support SQL is no way to manage 'real' data, and a forum for advocates of certain scalable data technologies to promote themselves.
It's been successful at getting people interested and excited about data systems outside the world of SQL which I think is really what the aim was. I wouldn't count on the word being in our lexicon for very long though or any products seriously branding themselves using this label, because these systems are likely to eventually come back around and support SQL like query languages in the future."

Warren Davidson (Objectivity):
" This is of interest to me, and should be to anyone interested in market change. Change is always difficult, but in the database world it seems especially so when you see how often an RDBMS is used even though it might be a technically poor decision. Since change is risky, in order for Objectivity, or Versant or Progress to get people to adopt its technology, you need momentum and corroborating support. The NoSQL movement lends credence to the first notion of change; one size does not fit all. So to have multiple technologies saying the same thing establishes credibility for change to begin, and from there people can start making the right technical choice, which may or may not be ODBMS. But this is ok, the industry needs the market to embrace a very simple concept of 'the right database for the right application'. They don't do that today, but the cloud movement is going this direction. And the NoSQL movement may help everyone. As they say, a rising tide lifts all boats. :) "

Stefan Edlich (Beuth Hochschule):
" There are some databases which are document oriented and nosql as CouchDB and SimpleDB. And there are document oriented ones which are not nosql as Lotus or Jackrabbit (a really weird system I think). I think the interesting tool and user group is the nosql group which excludes the latter group (hopefully). So the article you mentioned describes nosql with products storing documents as attribute data and not documents as pure byte / data documents (which Jackrabbit does)."

Daniel Weinreb (previously co-founder Object Design):
"They're being used not just as caches but as primary stores of data. There's one called Tokyo Tiger (and Tokyo Cabinet) that I've heard is particularly good."

William Cook (University of Texas at Austin):
" I think it is important! Facebook is using this style of data store. I'm not sure about the performance implications, but it needs to be studied carefully."

Raphael Jolly (Databeans):
"That a database could be designed without SQL is not a surprise to me since Databeans has no query language and is meant to be queried in Java (native queries). In addition, I'll happily believe that "relational databases are tricky to scale". However, the subject of extending Databeans with distributed computing capability has been on my mind for a long time and I presently have no idea how it could be done. What is interesting about NoSQL is how they mean to perform queries, i.e. through MapReduce. I don't know whether everything that can be expressed in SQL is amenable to MapReduce (this is probably not the case), but obviously a fair amount of what is done today on the internet is, the killer app being... search engines.
In summary, I tend to agree with this comment by alphadogg: "The main reason this [relegating to nich usage] will happen with the various key-value stores that are now in vogue is that they, like the predecessors, are built to solve a very, very specific issue on high-volume, low-complexity data, which is

not the norm. The NoSQL moniker is indicative of a misplaced focus. NO structured query language? Really? We want to go back to chasing pointers and recreating wrappers for access? ".
My perception is that even when they literally have no SQL based queries, object databases are very different from NoSQL technologies as currently understood because, as is clearly explained in your references, there is more to ODBMS than just the query language. Specifically : ACID transaction constraints, which "NoSQL" seem to relax quite a bit.
These constraints are difficult to manage in a distributed setting. One has to consider advanced concurrency control techniques. But with a careful design, nothing seems to prevent a fully structured approach.
In this respect, DHTs are clearly limited compared to classical object databases. Recently, I was reading about such an attempt a distributed design, yet with "strict" transactions: (download the book "XSTM: Object Replication for Java, .NET & GWT " as .pdf). "

Steve Graves (McObject):
" I thought alphadogg had good comments, although he has a relational/SQL bias."

Jonathan Gennick (Apress):
" it is an interesting discussion. I have heard the term "NoSQL". I did find the comment about relational databases not supporting key/value stores amusing: "...and index key/value based data, another key characteristic of "NoSQL" technology. The schema seems very simple but may be challenging to implement in a relational database because the value type is arbitrary."
In Oracle, one simply needs a table as follows:

```
CREATE TABLE key_value (
 the_key NUMBER,
 the_value BLOB);
```

There you go! Key/value. How much simpler can you get? "

Related Resources

- On NoSQL technologies - Part II. Roberto V. Zicari (editor) - February 10, 2010.

- The Object Database Language “Galileo" - 25 years later. Renzo Orsini – February 10 , 2010.

- Relational Databases, Object Databases, Key-Value Stores, Document Stores, and Extensible Record Stores: A Comparison.Rick Cattell – January 11, 2010.

- On NoSQL technologies - Part I.R. Zicari (Editor) et al. – December 14, 2009.

Labels: document stores, nosql databases, object databases, ODBMS