

On NoSQL technologies.

Editor Roberto V. Zicari

ODBMS.ORG

www.odbms.org

December 2009

Content

1. Patrick Linskey on "cloud store".
2. Kaj Arnö on "NoSQL databases".
3. Michael Stonebraker on "NoSQL databases".
4. David Chappell on "Introducing Windows Azure".
5. Robert Charles Greene: Are object databases "NoSQL" technologies?.
6. Giuseppe Maxia: On the evolution of "non-relational databases".

1. Patrick Linskey on "cloud store".

***RVZ:** Patrick, there has been recently a proliferation of "data stores", such as "document stores", and "nosql databases". Systems such as CouchDB, MongoDB, SimpleDB, Voldemort, Scalaris, etc. provide less functionality than OODBs but a distributed "object" cache over multiple machines.*

See for example: [wiki/Nosql](#), and [wiki/wiki/Document-oriented_database](#), and the article: [NoSQL: Distributed and Scalable Non-Relational Database Systems](#).

What do you think about it?

Patrick Linskey:

I think that the "cloud store" subset of them are pretty fascinating. Of course, as with so much in the software industry, much of what these projects are doing is old hat. But I think that they're relatively unique in:

- (a) successfully combining compelling complementary sets of features together,
- (b) building solutions for known and needed use cases, rather than the more ivory-tower approach that's all too typical of commercial products, and
- (c) designing and implementing in a manner oriented to cloud-scale deployment from the very start (i.e., lots of data; geographically diverse data centers; high load requirements).

I expect that all the successful cloud store projects will end up with support for declarative queries and declarative secondary keys. I really don't like the "nosql" term -- I think that Geir Magnusson does a good job of pointing out that the cloud store community is more focused on "alongside SQL". That is, there's nothing wrong with using a relational database in the situations where it's the best tool for the job. The new cloud stores are focused on filling

the gaps where most RDB alternatives fall flat.

The way they do it, of course, is by getting rid of problematic features. I think that some of the hype has mis-identified these problematic features, though. Declarative queries (and full metamodel introspection) and secondary key support are really cool and critical features of all the popular relational databases. The cloud store users out there are doing a lot of extra work because of the absence of these features -- essentially re-implementing them in their application code. Imagine how horrible it'd be if you told a modern DB team that they needed to change their app to tune their database!

So: what are cloud stores omitting that enable them to scale so well?

There are two answers:

- cloud stores are intentionally designed to scale. No* single points of failure, built-in support for consensus-based decisions, partitioning / replication as basic primitives, etc. Taking a codebase designed for a single server and evolving it to a multi-server solution is difficult, since single-machine assumptions often calcify into the implementation.

- more importantly, cloud stores aren't fully ACID, in the traditional sense of the term. By re-casting the data storage problem in more amenable terms (eventual consistency, atomic operations (but not atomic sequences of operations), etc.), the different products can make acceptable trade-offs that traditional single-server ACID stores are simply designed to forbid.

I'd love to see a comparison of established products like TeraData and Coherence to the various new cloud store projects. TeraData, in particular, does an interesting job of re-using the familiar SQL/JDBC model while making a lot of the same compromises and architectural decisions as the new set of cloud stores.

(I'm less interested in -- and educated about -- the single-server nosql projects. These days, I believe that all single-server databases are basically equivalent, since if you are using a single server, your application is sufficiently simple that you should be able to be successful with any of a number of data storage models.)

-Patrick

Patrick Linskey has been involved in object/relational mapping and databases for the last decade. As the founder and CTO of SolarMetric, Patrick drove the technical direction of the company and oversaw the development of Kodo, through its acquisition by BEA. At BEA, Patrick led the EJB team in designing and implementing the WebLogic Server EJB 3.0 solution, and represented BEA on the JDO and EJB3 expert groups. He is a contributor to the Apache OpenJPA project.

Since leaving Oracle, Patrick has worked on a number of projects, ranging from traditional three-tier web and mobile applications to C# peer - to - peer client applications with custom-designed distributed storage solutions.

Labels: [cloud stores](#), [document stores](#), [nosql databases](#), [Patrick Linskey](#).

2. Kaj Arnö on "NoSQL databases".

RVZ: What is your opinion of the so called "NoSQL databases"?

Kaj Arnö :

NoSQL is a catchy name, which in char(5) captures a lot of thinking. To be technical, it's not merely about removing SQL, but about removing most relational database overhead (where SQL, although dominant, is just an implementation of a query language). And some of that overhead is clearly not necessary all the time. It's a lot of protocol to implement all aspects of ACID compliance, and it isn't always needed.

Especially in the early days of MySQL, we were accused of cutting corners -- for instance through MyISAM not being fully ACID. Still, MyISAM was used a lot, and it still is. Coming back to the NoSQL debate, I would say that the MySQL idea of cutting overhead is gaining traction in other tools, which may choose to cut larger chunks or different corners. That's a healthy development, since the shortcuts to be taken depend on the class of application.

Kaj

Kaj joined MySQL in 2001, after 14 years as an entrepreneur. Serving as VP Services, VP Engineering and other exec roles at MySQL, he has been the VP in charge of MySQL Community Relations since 2005, continuing that position in Sun Microsystems. A native of Finland, Kaj lives in Munich since 2006. He devotes his free time to launching Runnism, the Religion of Running.

Labels: [Kaj Arnö](#), [nosql databases](#).

3. Michael Stonebraker on "NoSQL databases"

There has been a recent post by professor **Michael Stonebraker** related to the topic "No SQL" databases and their performance with respect to classical relational database systems.

In his post, titled "*The "NoSQL" Discussion has Nothing to Do With SQL*", Prof. Stonebraker argues that "blinding performance depends on removing overhead. Such overhead has nothing to do with SQL, but instead revolves around traditional implementations of ACID transactions, multi-threading, and disk management. To go wildly faster, one must remove all four sources of overhead, discussed above. This is possible in either a SQL context or some other context." The [Link to Stonebraker`s Blog](#) (courtesy of ACM).

Labels: [Michael Stonebraker](#), [nosql databases](#).

4. David Chappell on "Introducing Windows Azure".

I published an article of David Chappell: "Introducing Windows Azure". The paper describes Microsoft's Windows Azure. In fact, the "Tables" abstraction in Windows Azure is similar to some "nosql databases". You can [download the paper \(.PDF\) here](#).

Labels: [Windows Azure](#).

5. Robert C. Greene: Are object databases "NoSQL" technologies?

RVZ: Robert, you represent an ODBMS vendor, what is your opinion of the so called "NoSQL databases"? Are object databases "NoSQL" technologies?

Robert Charles Greene:

I find that lots of folks are getting all worked up over the dubbed "No SQL" movement. I guess it's because one can easily make assumptions and draw a would be obvious analogy to a "No Relational" movement and that would certainly be something to get worked up over.

As the object database guy, I see the core message being conveyed as, "one size does not fit all" when it comes to data management. That's a far cry from abandoning the SQL approach to data management and in my mind leaves little to defend, though some seem to feel threatened enough by the catch phrase to sound the alarm.

In some sense, this notion that "one size does not fit all" is an important change in attitude, because for many years one size fits all was prevalent. Only as the internet gave way to the masses and large scale concurrency and data generation ushered in a new era has the relational way of doing data management truly begun to break down, opening the door to alternatives.

The "right tool for the job" has once again become a mantra of the software development community and equally important, the mantra of the decision makers in Enterprise I.T. As evidence, one has to look no further than the proliferation of data warehousing solutions outside the realm of relational database technology, ironically, to support the adhoc query and analytics, the founding pillars of the past which brought the relational database to such high esteem. Indeed, necessity may well be the mother of invention, for if not, it would most certainly be the father of adoption. So, if the RDB is no longer the king of query, then really, what is there to get all worked up about if necessity drives adoption in yet even more directions.

So, what is this NoSQL movement all about and does it warrant the public espousal of opinions. Well, as stated above, this is an important change in attitude which will bring valuable choices to our industry making us better equipped to deal with today's infrastructure challenges, so yes, indeed it is worth discussion.

Michael Stonebreaker decided it was important to comment on this "movement" and gave an interesting NoSQL perspective here (courtesy of ACM).

I largely agree with the technical elements of his perspective, though I would suggest as in the above, the slightly different perspective that the core message is, "one size does not fit all". I encourage the reader to then keep this in mind as they engage in a broader understanding of what these exciting new technologies provide.

Also, it is worth pointing out, while many of the technologies involved in the NoSQL movement do sacrifice ACID as a means to achieve their end in both performance and scalability, most object databases are ACID compliant and one might argue are the original NoSQL movement.

But lets not digress, as even Michal asserts, the NoSQL movement is not about SQL. So, while object databases are by and large "NoSQL" technologies, they are not a kind of Query-less technology. Indeed, while today`s modern object databases embrace the requirement for distributed parallel query processing, they also hold true to the core tenants of large scale distribution, object clustering and parallel processing all in the context of an ACID compliant transaction. These features surround a robust environment for dealing with arbitrarily complex object models, an area in which many of the NoSQL movement participants fall short.

In summary, the "one size does not fit all" change in attitude is healthy and beneficial for all. To that end, the object database, a continuing NoSQL movement participant, is one more tool in the developers tool chest, enabling successful implementation of complex software systems of scale.

Cheers,
-Robert

Robert Charles Greene, is V.P. Open Source Operations, at Versant Corporation.

Labels: [nosql databases](#), [Robert Charles Greene](#).

6. Giuseppe Maxia: On the evolution of “non-relational databases”.

RVZ: Why NoSQL databases?

Giuseppe Maxia:

The evolution of non-relational databases (NRDB: I prefer this name to no-SQL) is rightfully puzzling. Their usefulness and efficiency are difficult to quantify in general terms and a comparison to relational database system is far to be objective.

There are cases where you can easily demonstrate that NRDB scale better than their relational counterpart. But only with a lot of ifs and buts.

Basically, the highest traffic web sites such as Facebook or Digg can't live with a database alone. There are two factors that limit their simple adoption of a relational schema:

1) the high traffic requires that the same values are fetched several times from the database. This requirement becomes a bottleneck for data. To overcome this limitation, there are auxiliary servers, such as memcached, which keep the most requested items in a fast network of in-memory storage systems. For all practical purposes, this technology converts the majority of the data into a series of key-value records.

2) when a site reaches a high number of registered users (or a high numbers of items to trade), a single server can't contain the database anymore. There is no way of fitting 300 million Facebook users into a single server. Thus, they do "sharding", i.e. a logical split of the data into tables, databases, and remote servers. With such organization, the relational model is conceptually broken, and the data looks more and more like a collection of key-value sets.

In both the above cases, you see that there is a trend to converting the relational data into key-values. The administrators of such sites start asking themselves why they keep bearing the burden of a relational database overhead since they can't have its main advantage, namely the precise and mathematically proven organization of data. In this scenario, the key-value databases are becoming popular among those users who are forced to break relational integrity.

Add to it the large number of developers who never managed to understand the relational model, and you can explain why the non-relational database systems are gaining momentum. The drawback is that NRDB can't retain meaningful metadata information, or, if they do, they achieve it through internal extension to the key-value model that is not easily exportable. The immediate effect of the above points is that more and more systems that are based on non relational storage are now entirely depending on the application that uses them, a situation that brings us back to the COBOL times.

This kind of storage is convenient only for either simple applications or for

organizations that can afford to employ a large number of developers to cope with the increased complexity of the applications. For the rest of us, relational databases are still the best way of storing data.

Cheers

Giuseppe

Giuseppe Maxia is a QA developer in MySQL community team. A system analyst with 20 years of IT experience, he has worked as a database consultant and designer for several years. He is a frequent speaker at open source events and he's the author of many articles. He lives in Sardinia (Italy).