On NoSQL technologies - Part II.

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

February 2010

Content

"NoSQL technologies": Interview with John Clapperton.

RVZ: *John, are object databases "NoSQL" technologies?*:

John Clapperton: The absence of SQL in "NoSQL" databases is less an a priori choice than a consequence of their simplified schema capability, imposed in the interests of higher performance, being unable to support the full set of SQL language constructs. It does not follow, therefore, that object databases from which SQL has been excluded for the opposite reason, as a language unable to address their more general representational capabilities, should be automatically included in the NoSQL classification.

A person thinking of adopting a NoSQL database will have certain capabilities in mind, so the question is really "Might an object database have 'NoSQL' capabilities?"

These include:

1) Data partition (which is application dependent).
2) Optimistic locking (which helps only if most accesses are read-only).
3) Relaxation of ACID transaction rules by:
   a) Data replication with eventual consistency, and/or
   b) Suppression of transaction logging and/or flushing, and/or
   c) Data storage in fast but volatile memory, sacrificing durability.
4) Fast navigational access to arbitrary data structures.

and in principle, an object database is capable of any or all of these.

The characteristics of an object database are its ability to manage arbitrarily complex object structures and to represent relationships by explicit named references. These have the potential for better performance by, respectively, reducing the required number of file writes for (de-normalised) data structures, and fast navigation of direct references instead of relational joins. However, against that must be set the cost of serialising arbitrary object structures for durable storage and instantiating the same on retrieval, compared with the simpler handling of pre-defined rows in a relational database.

Normalisation of behaviour, encapsulated in class definitions in language persistence odbms such as Logic Arts' VOSS for Smalltalk, reducing implicit replication in application programs and queries, may have an advantage in NoSQL applications, but it's not clear to me how significant that might be, given that its benefit is in managing complexity whereas NoSQL applications tend to be simpler.

***John Clapperton**, BSc CEng MBCS CITP, is proprietor and author of the 'VOSS' virtual object storage system, which extends Smalltalk with integrated database management, providing transparent access and transaction processing of persistent, versioned, Smalltalk objects. Previously, John has worked on database applications and research at Standard Telephones & Cables, Unilever Research, Acorn Computers and Deductive Systems.*

Carl Olofson on “Nonschematic" databases.

Carl Olofson:
“NoSQL” Databases, I would shy away from this term. A number of analysts (including myself) consider it a somewhat sloppy term intended to convey a certain spirit of rebellion. It actually derives from the core idea that the so-called "No-SQL" databases do not require schemas, and since most DBMSs are relational, it is simpler to say "NoSQL" than the more obscure "NoDDL".

In fact, OODBMS does require a schema, and the data structure, which is tied to the application object model, is key to how it operates, and especially to its transparent operational nature.

The so-called "NoSQL" database, which I call a "nonschematic" database, is one that requires no schema to be defined before data is loaded. One usually does define a schema afterward, through a process of data discovery and definition. If you know of a OODBMS that can accept undefined data, and allow schema definition after the fact, that could qualify.
Otherwise, I would shy away from the term altogether.

***Carl Olofson**, Research Vice President, Database Management and Data Integration Software Research, at IDC, sent me this note, where he argues about the term "NoSQL" in relation with object databases.*

Are object databases "NoSQL" technologies? Interview with Luis Ramos.

RVZ: Luis, how do you position yourself with respect to the so called "NoSQL" databases?:

Luis Ramos: We view many of the characteristics of the growing "NoSQL" movement as a market reaction to the realities of present day cloud-based data requirements, where ACID properties are not as important as performance, the bulk of the data's schema is not as complex, and the corresponding queries are relatively simple. Gone could be the days of complex relational schemas and the DBAs that are needed to maintain and administer them.
Similar phenomena have been seen in other areas. For example in programming languages, the reaction against the very complex and error prone C++ led to the popularity of Java.

In many respects, object databases can be classified as "NoSQL" technology. It satisfies many of the pivotal characteristics of today's "NoSQL" data stores. Object databases have been around since the late 1980s in response to the needs and requirements initially of the CAD market. At that time, the CAD practitioners needed an approach to data management that was fundamentally different than that provided by the relational databases.

Consequently, a whole new breed of non relational (object-oriented) databases emerged. Customers from other markets, whose requirements could not be met by SQL databases, followed. Call it the original "NoSQL" movement?
We certainly agree with Robert Greene's stipulation that "one size does not fit all."
An alternative way to put it is "You can put lipstick on a 'relational table' but its still a 'relational table'".

The schema-free characteristic that one finds in many "NoSQL" technologies is not entirely new. This is a requirement of many eCommerce applications developed in the 90s. There are object databases that support this nicely, enabling applications to store, manage, 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.

The horizontal scaling characteristic is another key requirement that object databases more easily supports. Multiple terabytes databases have been successfully deployed. These object database systems have a client-centric (rather than a server-centric) architecture. Data is distributed to the client and queries are performed on the client instead of on monolithic servers. Consequently, the data can be partitioned, replicated, and scaled much more easily without being tied down to the hardware limitations of a single server computer.

So indeed, object database systems could be considered "NoSQL" technologies. They can be utilized either as a persistent store for data as well as a cache.

Luis Ramos *is Principal Systems Engineer at Progress Software.*

Data Stores vs. ODBMSs

RVZ: Anat, systems such as CouchDB, MongoDB, SimpleDB, Voldemort, Scalaris, etc. provide less functionality than OODBs, but a distributed "object" cache over multiple machines. How do they compare with respect to ODBMSs?

Anat: We can categorize "stores" and see if they are more similar or different than OODBs, in a couple of ways.

By each dimension of the purpose of OODBS:

1. Persistent (could be accomplished by other methods like: replicating to other machines, using non-volatile caches, etc.)
2. Being queriable
3. Scalable (beyond what can be in cache, but could be distributed instead)
4. Objects vs. Relations

Arguable properties:
5. can express and query based on complex relationships among data items
6. can be shared among multiple "clients".

Many of these “other” databases are similar to oodbs in items 1, 3 and 4.
I am not sure they have capabilities in 3, 5 and 6 above.

There is a lot of interest in the USA in particular, w.r.t. to internet based applications and cloud computing. Big_table, and such look to me more like an algorithm based on traditional stores, rather than a db.

Dr. ***Anat Gafni*** *is VP of Engineering at db4objects.*