5-18-2013

NOSQL VS RDBMS - WHY THERE IS ROOM FOR BOTH

Cory Nance
Georgia Southern University, cory_a_nance@georgiasouthern.edu

Travis Losser
Georgia Southern University, travis_w_losser@georgiasouthern.edu

Reenu Iype
Georgia Southern University, reenu_m_iype@georgiasouthern.edu

Gary Harmon
Georgia Southern University, gary_r_harmon@georgiasouthern.edu

NoSQL vs RDBMS - Why There is Room for Both

Cory Nance
Department of Computer Science
Georgia Southern University
cory_a_nance@georgiasouthern.edu

Travis Losser
Department of Computer Science
Georgia Southern University
travis_w_losser@georgiasouthern.edu

Reenu Iype
Department of Computer Science
Georgia Southern University
reenu_m_iype@georgiasouthern.edu

Gary Harmon
Department of Computer Science
Georgia Southern University
gary_r_harmon@georgiasouthern.edu

ABSTRACT

The relational database or RDBMS has been the dominant model for database management since it was developed by Edgar Codd in 1970 (Shuxin and Indrakshi, 2005). However, a new database model called NoSQL is gaining significant attention in the enterprise. NoSQL databases are non-relational data stores that have been employed in massively scaled web site scenarios, where traditional relational database features matter less, and the improved performance of retrieving relatively simple data sets matters most. The relational database model and the NoSQL database model are each good for specific applications. Depending on what problem the organization is trying to solve, it will determine if a NoSQL database model should be used or if a relational database model should be used. Also, some organizations may choose to use a hybrid mix of NoSQL databases and relational databases.

Keywords

database; NoSQL; RDBMS

INTRODUCTION

In June of 1970 Edgar Codd presented a paper titled "*A Relational Model of Data for Large Shared Databanks*" (Codd, 1970) that introduced the relational database model. Since that time the relational database has been the dominant model for database management (Shuxin and Indrakshi, 2005). A new database model called NoSQL is gaining significant attention as an alternative model for database management. Most agree that NoSQL stands for "Not Only SQL". The goal of NoSQL is not to reject SQL, but to be used as an alternative data model for applications that do not work well with a relational database model.

Relational databases are not well suited for modern web applications that can support millions of concurrent users by spreading the load across a collection of application servers behind a load balancer. NoSQL database technologies have instead emerged to enable the cost-effective management of data behind new modern web and mobile applications. NoSQL databases can be used with applications that have: large transaction volumes, have the need for low-latency access to massive datasets, and have the need for nearly perfect service availability while operating in an unreliable environment. Companies using relational databases to achieve these goals, started building a large cluster of databases. When adding more hardware failed companies tried to scale down existing relational solutions. Companies began simplifying the database schema, de-normalizing the schema, relaxing durability and referential integrity, introducing query caching layers, separating read-only from write dedicated replicas, and data partitioning. These techniques helped with the existing relational databases, but none of the techniques addressed the core limitations that companies were trying to solve and added additional overhead and technical tradeoffs. The schema in the relational databases devolved from many interrelated fully expressed tables to simple key/value look-up (Padhy, Patra, Satapathy, 2011).

Trying to improve scalability in relational databases meant adding bigger servers. Bigger servers are highly complex, proprietary, and disproportionately expensive. They are in contrast unlike the low-cost, commodity hardware in web and cloud-based architectures. At this point, companies had taken relational databases so far outside their intended use cases that they were unable to meet performance requirements. Companies began building in-house databases that were tailored to

their particular workloads. These in-house custom solutions are the inspiration behind the NoSQL products that are currently seen on the market.

The most important question in evaluating the needs of an application and whether to use NoSQL databases or relational databases depends on the type of application being written, the type of queries that are expected, and the regularity vs. variability of the data's structure.

PROS/CONS OF NOSQL

NoSQL databases are highly scalable, reliable, have a simple data model, extremely bare query language, no mechanism for handling consistency and integrity amongst data, and almost no support for security at the database level (Okman, Gal-Oz, Gonen, Gudes, and Abramov, 2011). One of the most important advantages of NoSQL databases is that the databases can handle unstructured data. Unstructured data can be word documents, emails, audio, video, or even social network data. Also, NoSQL databases tend to scale very well on commodity hardware. Some even claim that NoSQL databases enable better performance (Leavitt, 2010), which is crucial for companies with large amounts of data. To enable faster performance, NoSQL databases typically don't adhere to ACID (atomicity, consistency, isolation, durability) restraints that are used in relational databases. While this is listed as a pro for NoSQL in terms of performance and processing time; we note that this also has undesirable results that will be addressed later. An example of NoSQL database's performance is Facebook's implementation (Cassandra) that is capable of handling over 100 million users continuously (Okman, Gal-Oz, Gonen, Gudes, and Abramov, 2011).

NoSQL databases are not without their faults and limitations. A common issue with these databases are lack of encryption support for data files, weak authentication between the client and the servers (and server members), lack of client communication encryption, and vulnerability to SQL injection or DOS (denial of service) attacks. Some NoSQL implementations are attempting to tackle these issues but their solutions are not yet production ready. Cassandra provides an IAuthenticate interface as an answer to authentication (Okman, Gal-Oz, Gonen, Gudes, and Abramov, 2011). Passwords are housed in a flat Java properties file and can be stored in plain text or as an unsalted MD5 hash. Storing passwords in plain text is a security risk for obvious reasons. MD5 hashing is not considered to be cryptographically secure and can be defeated using pre-calculated lists or rainbow tables (Okman, Gal-Oz, Gonen, Gudes, and Abramov, 2011). It is also important to note that Cassandra uses Apache's Thrift framework for client communication. As an unfortunate side effect of doing so, all passwords are transmitted in plain-text. This allows an attacker the ability to use a packet sniffer to view a password in transmission. Implementations such as MongoDB only support authentication when running in a standalone or replica-set modes. Currently MongoDB's sharded mode does not support authentication.

Table 1 shows a comparison between Cassandra and MongoDB and some of the key issues that these NoSQL databases face in terms of: data file encryption, authentication, client communication encryption, and injection attacks.

Category	Cassandra	MongoDB
Data at rest	Unencrypted	Unencrypted
Authentication	Not production ready	Unsharded configurations only
Client communication	No encryption	No encryption
Injection attacks	Possible	Possible

Table 1. CASSANDRA AND MONGODB

NoSQL databases do not adhere to ACID restraints like relational databases. This can lead to problems with applications that require great precision (Leavitt, 2010). All of this can be overcome but not without adding extra overhead and complexity. Additional programming must be done to ensure ACID compliance. After implementing additional rules, database performance is likely to take a hit. This is the main reason developers have chosen to leave ACID out of NoSQL databases and also an important aspect to keep in mind when selecting a database for a particular application.

A study conducted by Floratou et al (Floratou, Teletia, DeWitt, Patel, and Zhang, 2012) showed that NoSQL systems still lag behind optimized RDBMS in some cases. While this study was only done comparing one popular RDBMS with two popular NoSQL implementations, the results are nevertheless noteworthy for comparison. The study used the TPC-H DSS and YCSB benchmark to compare Microsoft's SQL Server PDW (Parallel Data Warehouse) with MongoDB and Apache's Hive. The study concluded that the SQL Server PDW was nine times faster than the MapReduce based data warehouse (Hive) when running TPC-H at a 16TB scale, even when indexing was not used in PDW (Floratou, Teletia, DeWitt, Patel, and Zhang, 2012). SQL Server was able to utilize cost-based optimizations and sophisticated query evaluation techniques to run more efficient plans than the NoSQL systems (Floratou, Teletia, DeWitt, Patel, and Zhang, 2012). This goes to show that in some cases RDBMS still has merit over their NoSQL counterparts.

Companies today tend to be weary of exploring their NoSQL options because they are unfamiliar with them. They do not feel knowledgeable enough to pick the correct flavor of NoSQL for their problem, and therefore choose to stay away from them altogether. Many of the NoSQL options currently available are open source and do not have customer support or management tools that IT departments look for when making decisions (Leavitt, 2010). Over the next five years NoSQL proponents will focus on developing better application compatibility and management tools (Leavitt, 2010).

NOSQL DATA MODELING TECHNIQUES

NoSQL databases take on a variety of modeling techniques depending on which database is used. Among those are key-value stores, document databases, and graph data models. Each one has its own benefits and downfalls. Overall a common theme is that developers will benefit from not having to map the relational tabular data to a non-tabular object within their programming language.

A key-value store is the simplest form of NoSQL databases. The schema consists of a key and value. The key is a string and the value is a BLOB. The database doesn't care what data type the BLOB is; this means it is up to the developer to enforce any data integrity rules. The flexibility of this model allows it to scale very well and allows developers to map values directly to programming objects. An example of this can be seen by how a developer can convert an object to JSON to store it and parse it back into an object when querying values. JSON is short for JavaScript Object Notation, and is a way to store information in an organized, easy-to-access manner. JSON is syntax for passing around objects that contain name/value pairs, arrays and other objects.

The document data model is particularly appealing to developers. In the relational model used by RDBMS, objects need to be transformed back and forth between the database and objects in the programming language. This can slow down development which will add extra complexity to the program. In a document database the document can map almost directly to the programming languages class structure, and this saves the programmer time (Harrison, 2010). Documents can be stored in a binary form (i.e. PDF, DOCX) or in an ASCII format (i.e. XML, YAML, JSON, BSON). This model isn't without its drawbacks though. Data integrity becomes an issue. For example, in a simple shopping cart we may store the items in a document and users in another document. It would be typical to store the order contents under the user's document; however, if an item's descriptive name changed then it would need to be updated in the item document and each user's document that ordered the item.

The graph data model is based on graph theory and uses nodes, edges, and properties to represent data. In comparison to the relational data model used by RDBMS, nodes are entities, edges are relationships, and properties are attributes. Data is traversed using pointers stored in each element. This model mimics the way we conceptually think of data and its relationships (Eifrem, 2012). Graph databases also have a speed advantage in regards to associative data sets. Similar to the document data model, the nodes map directly to the structure of programming objects. One of the big advantages to this model is that data can be represented and stored inherently; whereas, in the relational model data has to be transformed from non-tabular to tabular data (Eifrem, 2012).

RELATIONAL VS NOSQL DATA MODEL

There has been a lot of optimism surrounding the NoSQL movement. This comes along with some harsh criticism of the SQL based RDBMS system that has held dominance for the last forty years. The name NoSQL may suggest that there is no

need for SQL any longer. However, relational databases like Oracle and SQL Server are not going anywhere any time soon for the following reasons:

- Relational Databases have forty years of production experience. Big banks and other large institutions are not going to transition their transactional systems from Relational Databases to NoSQL databases.
- It is easy to find programmers with Relational Database experience. That is not the case for NoSQL databases.
- NoSQL databases and Relational Databases were meant to solve different problems.

Instead of deciding to use NoSQL over relational databases, an organization may want to consider migrating to a storage architecture that takes advantage of the strengths of both NoSQL and relational databases. Fowler and Sadalage call this architecture "polyglot persistence" to express the idea that applications should be written using a mix of languages and database technologies to take advantage of the fact that different databases are suitable for tackling different problems (Sadalage and Fowler, 2012). An enterprise will have a variety of different data storage technologies for different kinds of data. Figure 1 shows how an ecommerce system may be split between a NoSQL database and a relational database (Sadalage and Fowler, 2012):

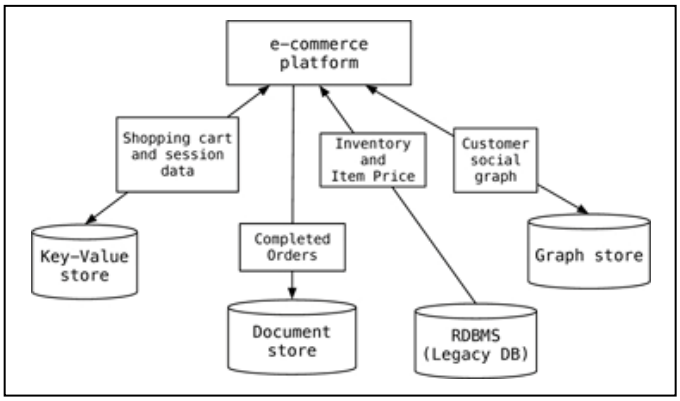

Figure 1. e-commerce platform using both RDBMS and NoSQL

An e-commerce platform has a lot of temporary data that does not belong in the main data store like a relational database. For example, a shopping cart and session data is temporary data that has a better fit in a NoSQL database, such as a key-value store. Also, completed orders and the customer social graph in an e-commerce platform both may have a better fit in a NoSQL database, such as a document store and graph store. Finally, the inventory and item price in an e-commerce platform has a better fit in a RDBMS system (Sadalage and Fowler, 2012).

The NoSQL database Amazon Dynamo, was born out of the need for a highly reliable, ultra-scalable key/value database. The Amazon Dynamo database was targeted at use cases that were core to the Amazon e-commerce operation. These included the shopping cart and session services. Any downtime or performance degradation in these services would have an immediate financial impact and their fault-tolerance and performance requirements for their data systems are very strict. Amazon has adopted a polyglot persistence like storage architecture, because Amazon uses a service oriented architecture. All the different content areas on their site are served by a different service. There is a: wish list service, a related to service, items customers have viewed service, a bestsellers service, a shopping cart service and many more. Each service has its own set of requirements and features. The requirements include things such as response time and availability. Internally, each service is implemented using whatever database best suits the needs of the service.

In the e-commerce platform the shopping cart could be served by a key-value store before the order is confirmed by the customer and the session data can be stored. The orders and products could be stored in a relational database like Oracle. The order history could be in a document store like MongoDB. Customer suggestions or products recommended to customers could be maintained by a graph database. The important thing to know when deciding how to split the storage architecture using the idea of polyglot persistence, is to be familiar with the various database products and to research the different approaches to be able to determine their strengths and weaknesses. The e-commerce example shows that it is not

necessary for an application to use a single database technology like a relational database when different database technologies are designed for different purposes and solve different problems. Not all data store problems can be elegantly solved by a single database.

WHY THERE IS ROOM FOR BOTH

NoSQL is a great tool for use in an online e-commerce platform. For example, stores such as Amazon and Barnes and Noble are in need of a highly reliable, ultra-scalable key/value database. This is because of the high volume of bandwidth coming through their servers. NoSQL is not only limited to e-commerce, as other businesses can benefit from NoSQL as well because of its ability to steadily evolve on many different platforms (North, 2010). "Some companies have already traded SQL's rich functionality for these new options that let them create, work with, and manage large data sets. A big reason for this movement, dubbed NoSQL, is that different implementations of web, enterprise, and cloud computing applications have different requirements of their databases. For cloud computing and high-volume web sites such as eBay, Amazon, Twitter, and Facebook, scalability and high availability are essential" (North, 2010).

A NoSQL only implementation is not the correct solution for all data storage issues because some companies have different needs than others when it comes to how they use their database and what it is used for by the company. The popular social networking site Facebook uses a combination of NoSQL and MySQL to run its online network (Diehl, 2011). Facebook has a tremendous amount of users that come to their site every day; and they need to be able to store all of the information coming in at a fast and reliable rate. Facebook uses a combination of NoSQL and MySQL that they have been modified to fit their needs.

Hadoop is a NoSQL database technology that some say could topple traditional database players like Oracle, IBM, and Microsoft. Hadoop allows companies to store and analyze petabytes of unstructured data and information in all sorts of formats such as documents, tweets, photos, etc; its technology appeal lies in its ability to break up very large data sets into smaller data blocks that are then distributed across a cluster of commodity hardware for faster processing. A regular relational database cannot deal with a lot of unstructured data. Hadoop adoption is gaining importance as more unstructured data flows into the enterprise arena, but its rising importance in the enterprise will not replace traditional relational database management systems. The open source data processing framework of Hadoop will complement relational databases with additional analytical capabilities. Companies that have adopted Hadoop, are doing so because it enables them to perform new kinds of data analytic capabilities.

Using strictly SQL or NoSQL for all business practices is not an option. Every business has a different need for how they plan to use a database and the needs depend on what a business wants out of its database. The SQL and NoSQL models both have their own set of pros and cons that each business has to identify, and then decide which one is better for their company; or if they should use a combination of both SQL and NoSQL (Bartholomew, 2010). Two main points companies should look at when comparing SQL to NoSQL is cost and performance. For many of today's high performance workloads, the lack of performance by SQL can cost a company a catastrophic hit in revenue. With the challenging economy today it is easy to see why some companies may not want to set up new systems and how emerging companies would opt for new NoSQL setups from an economical perspective.

Companies do not need to settle for either SQL or NoSQL; rather, they can use both which enables them to be able to perform complex queries or engage in multi-table transactions while ensuring that their atomicity, consistency, isolation, and durability remain intact. This ACID assurance gives companies the ability to use SQL interfaces and NoSQL interfaces without having to worry about many of the security fears that come with using only NoSQL.

CONCLUSION

The demand for relational databases will not go away anytime soon, nor will the reality of today's more distributed and commodity-based IT infrastructure. Large, public and content-centric applications will tend to be best served by NoSQL databases. In contrast, internal line of business applications that are supporting business operations will very often be best served by relational databases and may even be served exclusively by relational databases. Many applications will fall in the middle of that spectrum that may choose to use both relational databases and NoSQL databases or pick one or the other depending on the application. It is important to pick the right database technology for the task at hand. It is always important to deconstruct the facets of the various database technologies to make a more informed decision on the best approach to use for the data store needs.

REFERENCES

1. Padhy, R. P., Patra, M. R., and Satapathy, S. C. (2011) RDBMS to NoSQL: Reviewing some next-generation non-relational database's, *International Journal of Advanced Engineering Sciences and Technologies,* 11, 1, 15 - 30.

2. Sadalage, P. J. and Fowler, M. (2012) NoSQL distilled - A brief guide to the emerging world of polyglot persistence, 1, 1-77.
3. Harrison, G. (2010) NoSQL and document-oriented databases, *Database Trends and Applications*, December , 32.
4. Eifrem, E. (2012) Graph databases: The new way to access super fast social data, *http://mashable.com/2012/09/26/graph-databases/*, September 26.
5. Diehl, M. (2011) Facebook application development, *Linux Journal*, 208, 42-46.
6. Bartholomew, D. (2010) SQL vs. NoSQL, *Linux Journal*, 195, 54-59.
7. North, K. (2010) The NoSQL alternative, *Information Week, http://www.informationweek.com/development/architecture-design/the-nosql-alternative/224900559*, May 22.
8. Vijayan, J. (2011) Hadoop growing, not replacing RDBMS in enterprise, *http://www.computerworld.com/s/article/9218729/Hadoop_growing_not_replacing_RDBMS_in_enterprises*, July 29.
9. Codd, E. F. (1970) A relational model of data for large shared data banks, *Communications of the ACM*, 13, 6, 377-387.
10. Shuxin, Y. and Indrakshi, R.. (2005) Relational database operations modeling with UML, *Proceedings of the 19th International Conference on Advanced Information Networking and Applications,* 927-932.
11. Leavitt, N. (2010) Will NoSQL databases live up to their promise?, *Computer*, 43, 2, 12–14.
12. Okman, L., Gal-Oz, N., Gonen, Y., Gudes, E., and Abramov, J. (2011) Security issues in NoSQL databases trust, security and privacy in computing and communications (TrustCom), *IEEE 10th International Conference*, 541-547, 16-18.
13. Floratou, A., Teletia, N., DeWitt, D. J., Patel, J. M., and Zhang, D. (2012) Can the elephants handle the NoSQL onslaught?, *Proceedings of the VLDB Endowment*, 5, 12, 1712-1723