Week 4 Written Report

NoSQL Databases

Trevor Thomas

College of Computer and Information Sciences, Regis University

MSDS 610: Data Engineering

Paul Andrus

May 31, 2020

NoSQL Databases

Standard relational databases (RDBMS) can only store data in extremely structured formats, using predefined schema organized as columns and rows. They have been researched, developed, and optimized for decades, allowing for faster access to data and more efficient processing (Mohan 2013). Despite their tenure and high level of fine-tuning, unfortunately, roughly 80% of enterprise data is produced in unstructured formats, and it is estimated that 90% of new data in the upcoming decade will be unstructured (Das & Mohan Kumar 2013). An alternative approach is required if companies and researchers wish to capture and learn from this vast pool of unstructured data, and that is where NoSQL databases are useful. Unlike RDBMS before them, NoSQL databases do not force a schema onto the data and therefore provide additional flexibility for organizations with rapidly changing data needs. They take various forms utilizing different data structures and versions of query languages. Some are hybrid databases that can handle both relational and non-relational data, others only use unstructured key*value pairs. NoSQL itself stands for "not only SQL" and represents the broad category of databases that are not relational SQL databases. They have even been criticized by experts in RDBMS for eschewing standards, although even the critics themselves admit the diversity of NoSQL databases make it so no one criticism can apply to all of them (Mohan 2013).

Regardless of the perceived issues with NoSQL databases, their benefits and uses are undeniable. In order to work well with "big data," they are designed to run on distributed systems so their hardware is affordable to both replace and add, making both maintenance and scaling significantly cheaper than with RDBMS (Mason 2015). The use cases are diverse, but all stem from the ability to work with unstructured data in large quantities and scale systems easily and horizontally with changing needs. Even if the criticisms levied are accurate and the lack of standards will cause problems in the future, it also seems the lack of standards has given tech companies the latitude necessary to develop this new technology for an important need going forward. In addition to outlining the use cases above, this report seeks to discuss the different versions of NoSQL and their benefits, as well as compare and contrast the pros and cons of SQL vs NoSQL. There will also be a brief discussion of the results from the technical portion of this assignment.

Versions of NoSQL

As previously indicated, NoSQL lacks any development standards and is loosely defined as the set of unstructured distributed databases that offer an alternative to RDBMS. These qualities have made it so there is far more diversity among NoSQL databases than relational databases. Sometimes the differences are significant enough that comparisons are difficult and feel more like a comparison of apples to oranges. Rather than looking at all NoSQL options as different versions of a consistent category, it might be more useful to remove the category from the discussion and just talk about the different uses a particular distribution is designed for. There are far too many options to go through all of them, but some of the largest and most well-known are (in no particular order) Redis, MongoDB, Couchbase, and Cassandra. Redis is an open source distribution with in-memory storage, making it the best option when speed is a priority. It is touted as the fastest database in the world by many, and its in-memory storage system is unique enough that Redis Labs calls it a "data structure store, used as a database" rather than an actual database (Redis). The distinction is clearly semantic, but it illustrates the perceived uniqueness of Redis. It is used more often as a supplement to an enterprise's database rather than as a primary database itself, providing organizations with the flexibility for increasing speed when necessary (Sarig 2017).

MongoDB is another open source option defined primarily by its flexibility. The real focus of MongoDB is the “unstructured” aspect of NoSQL databases, storing everything in JSON-like documents with various structures that effectively allow the data to be completely unstructured (Sarig 2018). Rather than fill a specific niche, MongoDB’s flexibility makes it one of the best general-use databases, providing enterprises with a single-solution to diverse and rapidly changing data needs (Mongodb). Another general-use NoSQL database is Couchbase, which was specifically developed as an alternative to MongoDB that addresses some of its weaknesses. Specifically, Couchbase uses a masterless structure as opposed to MongoDB’s master-slave architecture, which Couchbase argues allows for full hardware utilization through workload isolation (Couchbase).

If MongoDB focused on the “unstructured” aspect of NoSQL databases, then Cassandra focused on the “big data” aspect. Cassandra is an open source option maintained by Apache, and it is the best option for databases that need to be able to scale rapidly and reliably (Apache Cassandra). It is widely used by large web companies that deal with massive amounts of data, such as Facebook and Instagram, the latter of which has over 80 million photos uploaded daily (Sarig 2018). Cassandra Query Language (CQL) is very similar to SQL, so there is less of a learning curve associated (jericevans 2012).

Pros and Cons of NoSQL and SQL Databases

NoSQL and SQL databases are not competing technologies – they are simply different solutions to different problems, and those problems will drive the pros and cons of each. A single feature could be seen as a pro in a certain context and a con in another, so the data goals will govern the conversation. In short, SQL databases are highly developed and standardized around ACID principles, making their primary benefit over NoSQL the integrity of data. RDBMS do not handle massive amounts of data well, and the data needs to fit into the predefined schema. This is a strength when the terminal goal is the storage of high-integrity data, but it is a weakness if the size and structure of data will change quickly. NoSQL handles this situation well at the expense of the integrity and consistency of the data. That is a pro for companies like Instagram where clickstream, location, and photo data are constantly changing and need real-time writing and analysis. In a perfect world, there would be a database that handles big, unstructured data well while simultaneously guaranteeing the integrity of the data and providing easy scaling, but that doesn’t exist. Organizations must determine what is most important to them – if the integrity of the data is paramount (as with online banking), there is no situation in which the cost-savings of a NoSQL database would be acceptable. On the other hand, if there is so much data coming in that the integrity and availability of it all doesn’t matter that much at any given time, the cost of an RDBMS would be prohibitive.

MongoDB Query Results

The MongoDB queries revealed several things about LA parking tickets and their qualities. The most expensive parking tickets were for $505, and there were only 6 of them in the entire database. Several tickets did not come with an associated fine. Of the tickets that had fines associated, the cheapest was for $10. The most common vehicle make was Toyota by quite a bit, followed by Ford, Honda, Chevy, and Nissan. It would be interesting to cross reference that with the most common vehicles in LA to see if there is any ticketing bias. The 5 most common non-CA license plates were AZ, TX, NV, FL, and WA. None of these are particularly surprising – 3 of them are in close proximity to CA, and TX and FL are extremely populous. Again, it would be interesting to cross reference this with licenses on the road to determine ticketing biases. See the technical lab write-up for a more detailed discussion of these results.

References

Apache Cassandra. *Manage massive amounts of data, fast, without losing sleep.* [Official Cassandra Website]. https://cassandra.apache.org/

Couchbase. *Compare Couchbase vs. MongoDB.* [Official Couchbase Website]. https://www.couchbase.com/comparing-couchbase-vs-mongodb

Das, T.K., Mohan Kumar, P. (2013, February). *BIG Data Analytics: A Framework for Unstructured Data Analysis.* School of Information Technology and Engineering, VIT University. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.411.6697&rep=rep1&type=pdf

jericevans. (2012, June 22). For all intents and purposes, CQL *is* SQL [Comment on online forum post *Cassandra CQL - NoSQL or SQL]*. Stack Overflow. https://stackoverflow.com/questions/11154547/cassandra-cql-nosql-or-sql

Mason, R. T. (2015). *NoSQL databases and data modeling techniques for a document-oriented NoSQL data-base.* Proceedings of Informing Science & IT Education Conference (InSITE) 2015, 259-268. http://Proceedings.InformingScience.org/InSITE2015/InSITE15p259-268Mason1569.pdf

Mohan, C. (2013, March). *History Repeats Itself: Sensible and NonsenSQL Aspects of the NoSQL Hoopla.* IBM Almaden Research Center. https://worldclass.regis.edu/d2l/le/content/249230/viewContent/3552443/View

MongoDB. *The Database for Modern Applications.* [Official MongoDB Website]. https://www.mongodb.com/

Redis. *Introduction to Redis.* [Official Redis Website]. https://redis.io/topics/introduction

Sarig, Matan (2017, May 29). *Redis Vs MongoDB.* https://blog.panoply.io/redis-vs-mongodb

Sarig, Matan (2018, May 1). *Cassandra Vs MongoDB In 2018.* https://blog.panoply.io/cassandra-vs-mongodb