**SQL vs NoSQL in Big Data**

Ryan Kaup

**Abstract**

Modern day Internet applications like YouTube, Facebook, Twitter, have dominated the web with popularity. Part of the reason is due to massive amount of user-generated content, and in effect, this has created large volumes of data called Big Data. Social networking isn’t the only source of Big Data, businesses holding a large amount of data pertaining to their practice also is held accountable. Attempting to hold this vast amount of data in a single database was difficult and expensive. NoSQL was scaled horizontally which allowed for easier and cheaper means of storing data in large amounts at a time. This paper goes over the different kinds of NoSQL databases, namely: document-oriented, key-value, column-oriented, and graph-oriented. Each database has its own strengths and weaknesses that notifies the reader which one is best suited for their database needs. NoSQL is a powerful tool for managing and storing data of mass volumes at a time, however SQL should not be cast aside. Despite the name NoSQL, it supports traditional SQL. That said, using SQL when appropriate is sometimes necessary as NoSQL does not support the ACID (Atomicity, Consistency, Isolation, Durability) properties.

KEYWORDS: NoSQL, SQL, Big Data, document, key-value, column, graph

1 **Introduction**

SQL stands for Structured Query Language, it was created for the purpose of storing and managing data in a relational database management system. Large amounts of data are produced every day from all different sources, such as social networking, business, or any other user-generated content. Managing data like this was challenging for SQL alone and so the answer was NoSQL. NoSQL stands for “Not only SQL” or “non-relational” and provides ways to store and fetche data non-relationally. The name NoSQL was first used by Carlo Strozzi in 1998 with his *Strozzi NoSQL open-source relational database* [11]. NoSQL gained a surge of popularity with the rise of Web 2.0 companies such as Facebook and Amazon.com. Web 2.0 refers to the World Wide Web websites that emphasize user-generated content and usability, and the need for being able to store and manage data of that scope was at its peak. Now there are hundreds of NoSQL systems out there anyone can download and use. NoSQL’s scalability and simplistic design solved data management issues relational databases couldn’t handle. The next section will go over the structure of NoSQL and how it is equipped to handle big data.

2 **The Structure of NoSQL**

NoSQL databases have been designed to provide higher scalability and availability than conventional relational databases. That said, many NoSQL systems do not support SQL natively, but can still support it separately [12]. The data models of NoSQL databases are divided in four categories: document-oriented, key-value, column-oriented, and graph-oriented. This is different from SQL as SQL is solely table-oriented. Strozzi aimed his first design of his NoSQL database towards simplicity, after stating: “Several times I have found myself writing application that needed to rely upon simple database management tasks. Most commercial database products are often too costly and too feature-packed to encourage casual use. There is also plenty of good free databases around, but they too tend to provide far more than I need most of the times” [11]. The design was meant to be easy to use by non-computer people and provide straight forward logic. For instance, to select rows of data, the 'row' operator is used; to select columns of data, the 'column' operator is used [11], but more operations will be discussed in section 3. NoSQL databases are essentially designed to have no arbitrary limits and work where SQL fails to, this is a key part to the success of NoSQL, being able to handle, in a sense, limitless data [11].

In contrast to SQL, which usually hosts a schema based design, meaning the structure of data should be known beforehand, NoSQL eliminates the need of any kind of fixed table schemas [5]. One major difference between the two is SQL is typically more vertically scalable while NoSQL is horizontally scalable. When something is vertically scalable, it means it can only hold more data by adding more power (CPU, RAM), thus making managing Big Data very expensive. NoSQL is the opposite, horizontally scaling can be done by multiple servers or by multiple machines and isn’t as costly [3]. NoSQL systems, as mentioned, are designed to be able to allow the connection of multiple machines be easy. An example of this is the vast amount of servers Google runs on, as it would be very unsafe to run all of Google on one machine that is vertically scaled. Why? If the one machine that was running Google were to crash, that would be very, very bad.

3.1 **NoSQL Data Management**

Big Data, as we have discussed, would be very difficult to run concurrently with RDBMS (Relational Database Management System) because like any Web 2.0 application, the software is likely going to need to run in parallel with thousands of servers and users. In order for a Web 2.0 application to be successful, it needs to be able to manage and, on some occasions, manipulate data in the Terabyte volumes. That said, it should be noted that NoSQL should not be used to manipulate sensitive data or very specific values. Social networking or online stores are likely to have the greatest volume of data, and in order to address their size, non-relational databases are used. The next few sections will cover the different types of NoSQL databases.

3.2 **Document-Oriented Databases**

Unsurprisingly, values in document databases are referred to as documents. Encoding is done at the document level by a markup language like XML or Java Script Object Notation [10]. In Figure 1 shown below, we observe that the document contains information about a bank. The document database also stores other information related to the column structure along with the main content, called metadata, in each column. This is what separates it from the key-value database. Metadata is used to query the data based on that information. For example, the NoSQL document-oriented database MongoDB features a method called find(). So, if the user wanted to search on LocalBanks to get information about banks within a certain area, they can do find() to retrieve information based on the metadata [6].

With document-oriented databases, these types of databases are the best for event logging, online shopping, list making, content management, and analytical practices. It has also been noted that due its schema flexibility, it can be useful for projects that rapidly change [10].

|  |  |
| --- | --- |
| BankName | “College Bank” |
| StateName | “CA” |
| CityName | “Example Village” |
| StreetName | “123 Testers St.” |
| ZipCode | “99999” |

Figure 1: Document Stores

3.3 **Key-Value Databases**

This kind of database is simplistic when compared to the others. This is a database where values are joined with keys. In Figure 2 shown below, the key represents states and the values are words often associated with that city. Keys are used to search a certain value in large volumes of data very fast. Keys can be associated with multiple values or single values. You can even store a value like “Las Vegas” as a key of another value, then it would be considered to be an object [10].

There are operations you can perform on a key-value database. Operations include Get (K): return the value(s) stored in key K, MultiGet (K1, K2, K3): return the collection of values stored in given keys, Put (K, V): store the new value V in key K, and lastly Delete (K): remove the value(s) from database [10].

Key-value databases are great for handling multiple sessions. Because of its simplicity, branching out on multiple machines isn’t difficult and won’t require much redesign [8]. A key-value database is best implemented when the values pertaining to the data can be done using a single key. It is related to the document approach, both are flexible but the key-value style doesn’t record metadata. Instead, this database utilizes a key to get access to the data needed, and the data can differ between each stored value. An example of a key-value database is DynamoDB which is used (and created) by Amazon.com when managing a buyer’s shopping cart [1].

|  |  |
| --- | --- |
| Key | Value |
| “California” | “Sacramento”, “Beach”, “Disneyland” |
| “Nevada” | “Las Vegas”, “Hot”, “Casinos” |
| “New York” | “City”, “Central Park”, “Busy” |

Figure 2: Key-Value stores

3.4 **Column-Oriented Databases**

Column-oriented databases store each database table column separately with attribute values belonging to that column stored contiguously and then is compressed, this is different from RDBMS where they store entire rows of values [10]. Searching a column’s values is faster, Figure 3 shown below illustrates this. If the user was only interested in values: 10, 20, and 30, they would need to select three different rows just to view those values. In a column-based database, the user would only need to select one column.

One popular column-oriented database is Google’s BigTable which is available for public use [2]. Bigtable hosts three dimensional mapping; two arbitrary values, row key and column key, and one called timestamp [10]. The first and second dimension, row and column, are used to store the data values and the third dimension is used to store timestamp values. The timestamp values maintain the history of the stored values so you can track them whenever they change. BigTable is what Google uses to manage major Big Data applications like Gmail, YouTube, and Google Drive and is helpful when tracking URLs or usernames [2].

Column-oriented databases are similar to documents, where they work well with event logging, content, and analytics. They are also very useful in fields where they may need to keep millions of records spread across multiple tables, like for instance a major business with many clients [4]. One thing to keep in mind when choosing this database approach is data manipulation is entirely inefficient. In traditional row-oriented databases, data manipulation is much easier. Executing a SELECT command in a column-oriented will give you an entire column of values that you selected, executing a DELETE command on an entire column would not be ideal especially for Big Data. Therefore, column-oriented databases are aimed more towards “warehousing” meaning: most operations are intended for read-only.

|  |  |  |
| --- | --- | --- |
| Column1 | Column2 | Column3 |
| 10 | 40 | 70 |
| 20 | 50 | 80 |
| 30 | 60 | 90 |

|  |  |  |
| --- | --- | --- |
| Column1 | Column2 | Column3 |
| 10 | 40 | 70 |
| 20 | 50 | 80 |
| 30 | 60 | 90 |

Figure 3: Row-based vs column-based

3.5 **Graph-oriented Databases**

Graph databases are useful when wanting to scale across multiple entities, these entities are connected by relationships [10]. In these databases, elements contain a pointer to an adjacent element and no index searches are needed. Perhaps the most common use for a graph database is for showing the relationship between users [9]. Applications like LinkedIn, Facebook, or Twitter utilize this function very often. In doing so, they measure the “degree of separation” between users [9]. The reason NoSQL includes graph-oriented databases is because they are related to semantic analysis. The semantics may be far greater volume than a relational database can handle [9]. As one could imagine, the volume of data between relationships of Facebook users is very large. The most used NoSQL graph database is Neo4j [7].

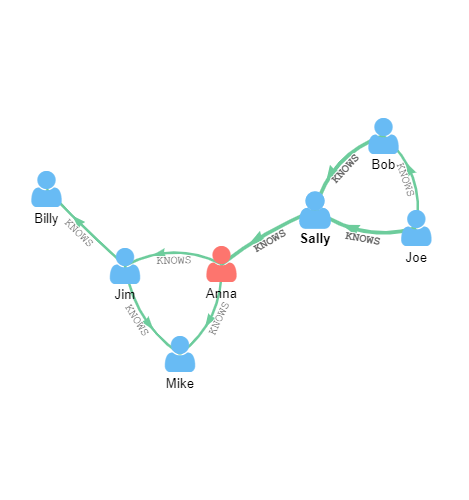


Figure 4: Graph Database Image. Retrieved from https://neo4j.com/developer/?ref=home-2

4 **Conclusion**

NoSQL is a very powerful tool when it comes to managing Big Data. Whether or not a user should use a NoSQL database is entirely based upon actual need. If the database is expected to accumulate large volumes of data, then implementing a NoSQL database will be a lot more beneficial. NoSQL offers much higher scalability and availability than SQL does, and can store and retrieve large amounts of data very quickly. While these are major benefits NoSQL offers over SQL, it is important to note that SQL is not by any means worse than NoSQL. In fact, relational databases should still exist where sensitive or private data exist because of its level of security. NoSQL, with all of its benefits, does not support the ACID properties, which is why it is encouraged that users implement relational databases for sensitive transactions.

References

[1] Amazon DynamoDB – NoSQL Cloud Database Service. (n.d.). Retrieved November 05, 2017, from <https://aws.amazon.com/dynamodb/>

[2] Bigtable - Scalable NoSQL Database Service | Google Cloud Platform. (n.d.). Retrieved November 20, 2017, from <https://cloud.google.com/bigtable/>

[3]Cloud. (2017, May 02). Horizontal Vs. Vertical Scaling: Which Is Right For Your App? Retrieved September 21, 2017, from https://www.g2techgroup.com/horizontal-vs-vertical-scaling-which-is-right-for-your-app/

[4] Daniel J. Abadi, Peter A. Boncz, and Stavros Harizopoulos. 2009. Column-oriented database systems. *Proc. VLDB Endow.* 2, 2 (August 2009), 1664-1665.

[5] Geomar A. Schreiner, Denio Duarte, and Ronaldo dos Santos Mello. 2015. SQLtoKeyNoSQL: a layer for relational to key-based NoSQL database mapping. In *Proceedings of the 17th International Conference on Information Integration and Web-based Applications & Services*(iiWAS '15). ACM, New York, NY, USA, , Article 74 , 9 pages.

[6] MongoDB Atlas - Hosted MongoDB as a Service. (n.d.). Retrieved October 29, 2017, from <https://www.mongodb.com/cloud/atlas/lp/general?jmp=search&utm_source=google&utm_campaign=Americas-US-MongoDB-to-Atlas-Brand-Alpha&utm_keyword=mongodb&utm_device=c&utm_network=g&utm_medium=cpc&utm_creative=229921764935&utm_matchtype=e&_bt=229921764935&_bk=mongodb&_bm=e&_bn=g&gclid=CjwKCAiAu4nRBRBKEiwANms5W-SrHfpZ-3lvxIhicd6kzzcdlqUCr4kyJ7_0ArhaUKj2I0GyDUhHgBoCCRsQAvD_BwE>

[7] (Neo4j)-[:LOVES]-(Developers). (n.d.). Retrieved November 01, 2017, from https://neo4j.com/developer/?ref=home-2

[8] Reeve, A. (2013, November 25). Big Data Architectures - NoSQL Use Cases for Key Value Databases | EMC. Retrieved November 10, 2017, from <https://infocus.emc.com/april_reeve/big-data-architectures-nosql-use-cases-for-key-value-databases/>

[9] Reeve, A. (2013, December 17). Big Data Architectures - NoSQL Use Cases for Graph Databases | EMC. Retrieved November 10, 2017, from https://infocus.emc.com/april\_reeve/big-data-architectures-nosql-use-cases-for-key-value-databases/

[10] Sharvari Tamane. 2016. Non-Relational Databases in Big Data. In *Proceedings of the Second International Conference on Information and Communication Technology for Competitive Strategies* (ICTCS '16). ACM, New York, NY, USA, Article 134 , 4 pages.

[11] Strozzi, C. NoSQL. Retrieved October 16,2017, from http://www.strozzi.it/cgi-bin/CSA/tw7/I/en\_US/nosql/Home%20Page

[12] Yasin N. Silva, Isadora Almeida, and Michell Queiroz. 2016. SQL: From Traditional Databases to Big Data. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*(SIGCSE '16). ACM, New York, NY, USA, 413-418.