Matthew Golden

[CS-300-01]

Professor Johnson

Spring 2023 Assignment 1

Non-Relational Data Storage and Retrieval Systems

NoSQL databases are considered non-tabular databases that store data in a unique way than relational tables, where stands for “not only SQL.” A NoSQL database is typically used to refer to any non-relational database. NoSQL databases come in several types based on different models of data. These can provide adaptable schemas and measure without difficulty with substantial amounts of data and high-level user loads. NoSQL databases can manage massive volumes of swiftly changing, unstructured data in separate ways than a relational database with tables. NoSQL technologies started in the 1960s but are experiencing a rise in use as the data environment shifts and developers need to adapt to oversee the pure volume and immense array of data produced from the cloud, mobile, social media, and other forms of big data. New data and data types are being created at a fast pace. NoSQL databases have been expanding in order to help developers quickly create database systems to store the information and make it readily available for analysis. NoSQL databases began their surge in the 2000s as the expense of storage diminished. It was no longer needed to create difficult to manage data model to avoid data duplication. In result, developers became the primary cost of software development, so NoSQL databases are adjusted for developer efficiency.

While storage expenses decreased, the volume of data that applications needed to store, and query grew. NoSQL databases permit developers to store massive quantities of unstructured data, providing them a lot of flexibility. Developers recognized the need to promptly adapt to varying conditions where the need of capacity to iterate quickly and adjust through software stacks and all the way down to the database; NoSQL databases gave them this lee way. With cloud computing rising in application, developers began using community clouds to host data. This gives the ability to disseminate data across many servers to make their applications robust to expand out and to logically place their data. There are NoSQL databases, MongoDB to name one of them, that provides these capabilities. Data is what brings about information systems and is at the heart of those systems. A company’s efficiency in organization and operation of data is of the highest priorities. The technical environment is evolving very quickly in modern times; companies must continue to assess and choose their databases so that they will meet their future needs and support their growth to match current trends.

Relational databases during the late 20th century were used to store data, and today they remain to stand for a sustainable solution for myriad in useful scenarios. But consequently, NoSQL databases were conceived out to answer to the shortcomings of relational database technology. When comparing relational databases to that of NoSQL databases, we see that the latter possesses the ability to scale and present superior performance unlike the former. The NoSQL data model corrects several limitations of the relational model. In most cases, NoSQL databases are intended to solve problems of data managing in bulk, from multiple sources, and in multiple formats, in many intense conditions. This was able to offer a pioneering methodology to take on capacity needs and new types of data.

A NoSQL database manages information using primary data models. First discuss here is the Document database. Document databases store data in a document data model using JSON or XML objects. With each document each contain their own markup that identifies fields and values. Values take the quotidian data types such strings, doubles, ints, Booleans, etc. Augmenting its usability, this data model can fluctuate from record to record. It is widely held with developers because JSON documents depict structures that characteristically line up with objects developers that are working with in code. These databases are implemented with a wide scaled architecture that gives developers a path to supporting vast the amounts of data traffic that is encountered.

Another primary data model is the Wide-column store. This model handles data using a revised table model. Wide-column stores are used in storing significant amounts of data. The data in this configuration is stored using key rows linked with one or more columns. The model is flexible in comparison to other models, we see that the structure of column data can adjust from row to row.

An additional primary data model is the graph database. The particular database here is typically a system for housing data from a graph. Data features are put in storage as nodes, edges, and properties, where edges define the relationship between the nodes. Graph databases can be used for storing and controlling a system of relationships between items inside of the graph.

The last primary data model to be discussed is the Key-value databases which use a quite straightforward schema. A distinctive key is paired with a compilation of values and the values here can be take on data types: strings, doubles, ints, Booleans, etc. One way that databases using this structure gain in performance is that there are no complex queries.

The number of popular databases systems available is numerous. To choose the right management system for you or your organization, it is important to have an idea about what exists in the market. Also, plentiful options of free or purchasable options. Some examples of NoSQL databases are: CouchDB; MongoDB; HBase; Redis; Cassandra.

The principal reason for using not only SQL is that you have data quantities that are reaching the performance limits of the relational database management system. But there are resolutions to this with that two main characteristics of NoSQL databases that make a significant distance. The attributes data flexibility and scalability. If the need ever arises to get into a comprehensive assortment of data formats in a single system, the developers at a company can spend copious amounts of time trying to create a regulated schema accept all the dissimilar sources of data formats. In a variety of cases data can simply increase at ongoing and unpredictable rates. Prices when scaling up a relational database management system to oversee that capacity be able to be exorbitant. All in all the NoSQL databases provide the data functionality and scalability to manage the big data environment in our modern world in a surprisingly a cost-effective manner.

Although the need to discuss the NoSQL disadvantages develops when having any legitimate discussion. Specifically, this discussion is about what situations where NoSQL databases are not ideal. To be very clear, you can NOT use NoSQL as a proxy for a relational database management system. Obviously, SQL is not fully supported in these systems. It do not possess the multi-row transactional arrangement which can ensure changes that make relational database management system so powerful. Another big disadvantage to the NoSQL is there is no API standard across NoSQL tools, so creating applications cannot be easily ported to other NoSQL databases which creates a lot of different challenges. With all being said we see NoSQL remain to succeed because of the significant advantages it provides a cost-effective scaling and performance benefit.

In many of our individual experiences we know once data we have is too complicated to fit into a single table, we usually move on to relational databases. But when the data becomes too complicated for this solution, we have to move on to the next one. This solution is the graph database. In a graph database, any data point can be connected to any other data point, and the connections are and can be determined at any time by users without the need for database administrators to rewrite the entire schema. The creation of graph databases intended them to be scalable, which is extremely useful in a variety of different manners, but especially big data applications. A lot of graph databases are fast, allowing the users to swiftly move down a chain of interlinking, allowing all kinds of companies to find intuitions more rapidly and more effectively.

We can define a graph database in terms of a specialized, singular purpose program for creating and manipulating graphs. Graphs contain nodes, edges, and properties, all of which are used to represent and store data in a way that relational databases do not know how to do. Graph databases are also commonly referred to as graph analytics, and this is used to refer to the known process of analyzing data in a graph presentation using relationships as edges and data points as nodes. Graph analytics does require a database that is capable of supporting graph formats. A dedicated graph database, or a converged database supports multiple data models, including graph.

In many of our personal experiences we know once data we have is too complicated to fit into a single table, we usually move on to relational databases. But when the data becomes too complicated for this solution, we have to move on to the next one. This solution is the graph database. In a graph database, any data point can be connected to any other data point, and the connections can be established at any time by business users without the need for database administrators to rewrite the entire schema. Graph databases are designed to be scalable, making them well suited to today's big data applications. And they are fast, allowing users to quickly move along a chain of connections, allowing businesses to find insights faster and more efficiently.

An example of a graph database product is the Amazon Neptune platform. Amazon Neptune is a fully-managed graph database service that lets you build and run applications that work with highly connected datasets. The foundation for Neptune is a purpose-built, high-performance graph database engine optimized for storing billions of relationships and querying the graph quickly. It supports graph models like Property Graph and W3C’s RDF and their respective query languages Apache Tinker Pop Gremlin and SPARQL as well. Neptune is recommended for use cases like fraud detection and network security.

An example of a graph database product is the IBM Graph platform. IBM Graph is an enterprise-grade property graph as a Service that is built on open-source database technologies. The product enables you to store, query, and visualize data points, connections, and properties in a property graph. IBM Graph was built to ensure always-on service while experts monitor, manage, and optimize everything in a customer's stack. Organizations can start small and scale on-demand as data and complexity increase as well.

Unsurprisingly the graph format gives more functional platforms discovering the distant connections needed to connect or to start analyzing data. Graphs give leave to investigations and discoveries of connections and patterns in social networks, complex webs, hierarchical structures, and big data. It allows complex transactional data for all sorts of companies or organizations. In our modern age, we see the graph databases are progressively being utilized as a component of information science to make relationships in interactions clearer. As graph databases explicitly store the associations, queries and procedures operating the connectivity amongst apexes can be completed in seconds instead than longer periods of times such as days or weeks. Users do not require to perform limitless connects and the data can be able to more easily be used for analysis and machine learning. Graph databases are exceptionally adaptable whilst also being an exceedingly impressive tool. The graph format allows these complex relationships to be determined for greater understandings with a lesser amount of energy. Graphs can emphasize relationships between data, this makes them the ideal for various analysis procedures and techniques.

To summarize the advantages: Graphs exceed at attaining the short paths between two nodes; analyzing the state of the network or community based on connection distance/density in a group; examining connectivity and identifying the feeblest points of a system; determining whether and which nodes generate the most significant impact.

The common disadvantages of graph databases are there is not a consistent query language; the language is contingent on the platform used (platform-dependent); graphs are unsuitable for transactional-based systems; in the world it is found that the user-base is insignificant comparatively speaking and makes it difficult to attain support when running into an issue.

Work Cited

"Apache Cassandra." *Apache.org*, 2016, cassandra.apache.org/.

"Apache HBase – Apache HBase^{TM} Home." *Apache.org*, 2013, HTTPS://HBASE.APACHE.ORG/.

"Graph Database: How It Works, Its Uses & Benefits." *InfluxData*, www.influxdata.com/graph-database/.

MongoDB. "The Most Popular Database for Modern Apps." *MongoDB*, 2019, www.mongodb.com/.

"NoSQL Database: The Definitive Guide to NoSQL Databases." *Pandora FMS Monitoring Blog*, 20 Apr. 2017, pandorafms.com/blog/nosql-databases-the-definitive-guide/.

"NoSQL or Not NoSQL?" *CodeProject*, 6 Dec. 2016, www.codeproject.com/Articles/1158951/NoSQL-or-not-NoSQL. Accessed 23 Jan. 2023.

Redis. "Redis." *Redis.io*, redis.io/.

"What Are NoSQL Databases? | IBM." *Www.ibm.com*, www.ibm.com/topics/nosql-databases.

"What Is a Graph Database?" *Oracle.com*, 2021, www.oracle.com/autonomous-database/what-is-graph-database/.

"What Is a Graph Database? {Definition, Use Cases & Benefits}." *Knowledge Base by PhoenixNAP*, 22 Apr. 2021, phoenixnap.com/kb/graph-database.

"What Is NoSQL and What Are Database Operators?" *Ubuntu*, ubuntu.com/blog/what-is-nosql. Accessed 23 Jan. 2023.

"Why Do Developers Prefer NoSQL Databases?" *Oracle.com*, 2021, www.oracle.com/database/nosql/what-is-nosql/.