Garrett Ferguson

Dr. N. Johnson

CS-300-01ON – NoSQL and Graph Databases

23 January 2023

Non-Relational Data Storage and Retrieval Systems

For the past few decades, computer science as a field has grown exponentially in many fields. Computers have gone from massive rooms with massive racks containing massive parts to having the sum total of human knowledge in our pockets. Advancements that used to take years to develop seemingly happen overnight. Development in the field of data science occurred concurrently with computers as the need of creating, storing, and analyzing data became ever growing. Even today, the search for more efficient data has taken the focus of many data scientists. But how does one manage all of this data? But creating and utilizing database systems. Through the usage of advanced database systems, scientists have been able to do such amazing things such as put the sum total of human knowledge at a person's fingertips within a person's phone. However, there is no "free lunch" when it comes to designing anything for general usage.

As the complexity of databases has grown, new management techniques were needed to properly handle the data. One such solution that became very popular was SQL or Structured Query Language for management. SQL manages relational databases and performs operations on the data. Relational databases store things tabularly which presents an issue: they cannot scale well. NoSQL is Not Only SQL and is designed to be non-tabular. This allows for the database to be scalable and can take on larger workloads than a standard tabular relational database (MongoDB).

NoSQL became popular in the late 2000s due to the drop in the cost of storage. This means that more data could be stored for less cost than previously. It eliminated unnecessary complexity in the database and created a paradigm shift in development that focused more on productivity rather than problem chasing such as ensuring there was no data duplication or bad data in the database (MongoDB). As data science evolved, newer data structures were utilized that complicated schema and made tables difficult to create since data could be in many different forms. NoSQL enabled flexible data storage of non-standard and unstructured data (MongoDB).

As is commonplace within the realm of Computer Science standards, there are a large variety of NoSQL products that can be utilized to create large, scalable, and efficient databases optimized for multiple types of schema and data. Three examples of products that are NoSQL based and are widely used by companies are MongoDB, Cassandra, and Elasticsearch (datacamp). Each product has its own unique advantages and disadvantages.

MongoDB is by far the most popular and commonly used NoSQL product. It is open-source and is highly flexible on the types of data it can take. It can store structured, semi-structured, and non-structured data with relative ease. It uses a document style data storage format that enables a large amount of scalability and flexibility unlike tables. A normal relational database uses a system of tables and rows while MongoDB uses what they call collections and documents where documents are filed into collections and the documents are accessed via a key-value pair system (PureStorage). Due to the nature of the key-value pair, unique data structures can be put into documents including whole arrays of other documents that are simply accessed via its associated key. To search, the user uses a text operator to search for the desired collection. Because of this, there are inherent benefits to using MongoDB. These include flexibility of architecture, sharding of scaling, and improved performance (PureStorage). MongoDB is dynamic with its architecture because the database schema does not need to be predefined before being initialized. This allows for rapid modification without having to temporarily shut down for maintenance. This also allows for one collection to have documents comprised of different data types such as pictures, music, videos, etc. With sharding, data can quickly and effectively be shared across multiple machines. This allows for ready access of data for multiple users across a multitude of locations. Lastly, MongoDB utilizes a machine's RAM rather than its disk drive in order to access data faster from an operation. Less power is also used from accessing RAM than the hard disk (PureStorage).

Cassandra is the second most popular NoSQL product and is used by Netflix and Apple to run very large operations. It is best known for its ability to essentially infinitely scale and still perform well. It operates on a peer-to-peer connection system without the usage of a leader node (datastax). Cassandra takes the approach of having a decentralized architecture and creating partitions to handle very large data loads. With decentralization, the data is not limited to one platform or another allowing it to be run by anyone and anywhere in the world, hence why it is used by companies like Netflix who stream all around the world. Because of the lack of a centralized node, there is less of a risk of sudden and complete outages from the nodes being reliant on one leader node. Each peer-to-peer

node is an instance of Cassandra and can store data. Data gets replicated infinitely per each node that is used so if one node goes down, another node can be connected to to prevent downtime. The replication is referred to as the replication factor. A user can connect to any node and will receive the same accurate information and very fast speeds. Cassandra uses key based partitioning of its data (datastax). It still utilizes a table system but this table is in a keyspace that is accessed by the key. Within the table is a partition that has rows of data within them. This allows for large scalability via storing chunks on different servers. Scaling is handled by Cassandra via token assignments of the partitions and is adjusted accordingly. However, once a key is created for a table, a new table needs to be created to create a different key (datastax). Comparatively speaking between MongoDB and Cassandra, Cassandra's ability to use peer-to-peer is superior to MongoDB's inability to do so, but MongoDB's flexibility is far superior because of its ability to take in and then subsequently process that data. The value of the flexibility cannot be understated when comparing these two very popular NoSQL products.

The third most popular NoSQL product is called Elasticsearch and is another large scale, open-source library. It is an analytics search engine and is the most popular database search engine (sematext & DB-Engines). Elasticsearch can data of any structure from any place and then handles it based on mapping that is unique and derivable from the data itself. It is distributed like MongoDB and can therefore analyze large projects across multiple machines (sematext). Elasticsearch is primarily used for text based searches and Amazon has a distribution that it uses. Since it is built using Java and Java is a program that is platform agnostic, Elasticsearch is also platform agnostic like MongoDB and Cassandra. Another advantage is that it is multi language compatible so it has libraries for use with Java, Python, C#, and other popular coding languages. Elasticsearch seems to be very useful with use in online shopping. It can use analytics and data slicing to effectively search through a large database of items. Out of the three popular NoSQL libraries, Elasticsearch seems to be useful only in searching through databases, Cassandra is very useful for streaming based databases and other forms of entertainment databases, and MongoDB is the best all around database. This is probably the reason why MongoDB is so popular and widely used in the world of data science and databases.

As mentioned earlier, there is no free lunch with coding and databases. NoSQL has its benefits and its downsides that one has to consider when designing and then implementing a database. In terms of advantages that NoSQL has, it is flexible, scalable, fast, and easy to use. One issue is that since they are so easily replicated, they are

not very isolatable and are not fully consistent across multiple documents. The biggest issue is the integrity of the data (MongoDB). The benefit of having highly accessible and distributable data is that anyone, anywhere can access the database and interact with it. However, this presents the possibility of inducing clerical errors that can reverberate for all users of the database if it is not handled properly and swiftly. Using this incorrect data can cause exponential errors in accuracy and could cost time, money, and effort to fix.

Along with NoSQL, a type of database called the graph database was developed to address issues with standard SQL databases. Graph databases are purpose built for relationships between data. They are designed to store and navigate them effectively and their value is defined by their relationships. Within the graph, there are nodes that store the data and edges that store the relationships between the data. The edge contains the two related nodes, the relationship type, and which is the parent node between the two (Amazon AWS). Since these relationships can be infinitely defined, the number of relationships a node can have is also infinite. These databases are excellent for social networking as a user can be created as a node and then relationships are built around each person to connect nodes (people) to other nodes. Other use cases include fraud detection and recommendation engines. All of these use cases can be done in near real time as the graph relationship system is extremely efficient and can quickly traverse each relationship.

The problems that can be solved by graph databases is the necessity of real time connections. In the case of social media, this means that a person can be turned into a node of data as mentioned above. Using node relationships, a person can be instantly connected to other people via node to node connection. Relationships can then be established based on friend status and how often the two people interact. Data can then be gathered on each person and further relationships can be created to determine more node connections with other users and also make connections with advertiser nodes. This allows companies to create personalized advertising for users and create better connections. In the case of fraud detection, it can, in near real time, verify that a node or person is who that node says it is. Fraud is hard to handle once it happens and heading off the issue before it can occur can save precious time and money of both companies and the average person. Graph database relationships can be checked and verified in a near instantaneous manner and can therefore be used to create essentially real time assessments on security. A person could have card information stolen and using a relationship between that person's typical spending habits and/or that person's relative geolocaltion, can determine if a purchase made is a good transaction. If

a person who lives in Arkansas typically does not buy alcohol and does not visit Washington state, and that person's card is used to buy alcohol in Washington state, the card provider can very quickly and easily cancel the transaction and turn off the card to prevent further fraud. With modern graph databases, not only can this be detected in the moment but it can also, ideally, be detected at the card reader, both preventing the fraud and frustrating the person who is attempting to commit the fraud. Another common use case, is the usage within a local network at a branch building in a company. While a lot of modern databases have excellent cybersecurity, a big issue is still onsite security. Using graph databases, a company's IT Department can monitor the data access within the building in near real time and get alerts when local systems are accessed in a suspicious manner and then can be reported to company security as well as have that machine's local access cut off to prevent further attack on the database. Graph databases have numerous real world use cases that fix many problems that other databases could take a lot of time to chase. All that needs to be done is check relationships.

Two examples of graph database products are Amazon Web Service's Amazon Neptune and Cambridge Semantics AnzoGraphDB. Both have their pros and cons and both have their best use cases. Similar to standard NoSQL products, each graph database product is optimized for one of the real-world use cases mentioned above and others not previously mentioned in this paper. When considering what a company needs for their graph database, comparing the optimizations is critical for saving time, money, and creating the best user experience.

Amazon Neptune is a service that lets one build and manage highly connected sets of data. It is optimized for storing extremely large amounts of relationships between nodes and then querying at rapid paces (SolutionsReview). Amazon Neptune is best optimized for fraud detection and network security due to its ability to check all the relationships in near real time. It also allows for continuous backup on Amazon's servers. It is self-managed which allows for self-configuration of hardware. Since everything is built-in, it is easy to implement, scale, and integration within a company.

The Cambridge Semantics AnzoGraphDB is designed as a parallel processing graph database that is optimized with data analytics expediation. It contains a library of functions as well as the ability to build and use new specialized functions for specific data analysis within a field (SolutionsReview). The AnzoGraph scales by meaning rather than location to create better relationships between nodes. Using the semantic process creates a more

user friendly database interface and creates better views. The AnzoGraph can also be utilized to better train machine learning and AI functions since the relationships are based on meanings (Cambridge Semantics).

Continuing in the theme of no free lunch, it is important to discuss the advantages and disadvantages of graph databases. Some advantages include query speed, near real time results, manageable relationships, and flexible structures (Ionos). The query speed is very fast because it is based on the number of relationships between nodes. This allows for the results to be in near real time to allow for quick turn around. Due to the nature of the relationship mapping in graph databases, it is easy to identify and then manage those relationships. All of this combines to create a flexible database for use in a variety of fields. However, there exist two primary disadvantages. Those disadvantages are scalability difficulties and lack of uniformity in languages. These databases are harder to scale vertically because there is typically no uniform set hierarchy. The lack of uniformity in languages is both a benefit and a downside because it allows the program to work with any coding language but makes the data have different structure standards that might have translation problems from one structure to another.

Works Cited

Bova, Ben. “Neptune.” *Amazon*, Tor, 2021, aws.amazon.com/neptune/.

“Cambridge Semantics.” *Cambridge Semantics*, 28 Feb. 2022, cambridgesemantics.com/.

“Engines Ranking.” *DB*, db-engines.com/en/ranking/search+engine.

Follow TimTimothy KingSenior Editor at Solutions ReviewTim is Solutions Review's Editorial Director and leads coverage on big data. “The 12 Best Graph Databases to Consider for 2023.” *Best Data Management Software, Vendors and Data Science Platforms*, 5 Jan. 2023, solutionsreview.com/data-management/the-best-graph-databases/.

“Graph Databases Explained.” *IONOS Digital Guide*, www.ionos.com/digitalguide/hosting/technical-matters/graph-database/.

“Introduction to Apache Cassandra - the ‘Lamborghini’ of the Nosql World.” *DataStax*, www.datastax.com/blog/introduction-to-apache-cassandra-the-lamborghini-of-the-nosql-world.

Keita, Zoumana. “NoSQL Databases - Types of NoSQL Databases and How to Use Them.” *DataCamp*, DataCamp, 24 June 2022, www.datacamp.com/blog/nosql-databases-what-every-data-scientist-needs-to-know.

MENEGASSO, ANDRE EDUARDO. “NOSQL.” *Amazon*, NOVAS EDICOES ACADEMICAS, 2018, aws.amazon.com/nosql/graph/.

“NoSQL vs SQL Databases.” *MongoDB*, www.mongodb.com/nosql-explained/nosql-vs-sql.

Sematext. “What Is Elasticsearch: Getting Started Tutorial for Beginners.” *Sematext*, Sematext, 31 Aug. 2022, sematext.com/guides/elasticsearch/#:~:text=Elasticsearch%20takes%20in%20unstructured%20data,data%20in%20near%20real%20time.

"What Is Mongodb and How Does It Work?: Pure Storage." *What Is MongoDB and How Does It Work? | Pure Storage*, www.purestorage.com/la/knowledge/what-is-mongodb.html#:~:text=How%20Does%20MongoDB%20Work%3F,tables%20in%20a%20relational%20database.

"What Is Nosql? NoSQL Databases Explained." *MongoDB*, www.mongodb.com/nosql-explained.