Brian Johnston

Assignment 1

CS-300-ON

Database Management Systems

Non-Relational Data Storage and Retrieval Systems

When it comes to database management, the typical tools are query writing software tools such as SQL or R. These fall under the more traditional “relational” systems, but they do have their limitations on what they can achieve. There are other systems designed to overcome those issues, generally called NoSQL.

To understand what NoSQL means, a little background on what SQL is needs to be addressed. SQL, or Structured Query Language, is a tool, created by IBM in the 1970s, which is used to write queries that search, retrieve, or otherwise manipulate data stored in a database. This type of tool is seen as a relational system, meaning that the database that SQL performs its queries on holds its data in tables where a relationship between the data can be determined (FutureLearn, 2023). Thus, we can extrapolate that NoSQL would define a tool to be used on a non-relational database, where the data itself is not structured in a table where there are connections between the data itself. There is very clearly a need for both types of systems, as both relational databases and non-relational databases are immensely prominent in our current technology landscape.

With a better understanding of what NoSQL is for, the question then falls to ask what types of queries would need to be performed on a non-relational database. The simple difference in how the data is stored between the two different types of database is one of the issues that NoSQL handles well: “Instead of the typical tabular structure of a relational database, NoSQL databases, house data within one data structure, such as JSON document. Since this non-relational database design does not require a schema, it offers rapid scalability to manage large and typically unstructured data sets.” (IBM) This generally means that NoSQL databases can hold much more data than a relational database. While that is a major problem that is overcome by the use of a different style of database, the more interesting use cases are the management of data relationships which can be achieved with graph based NoSQL databased, or the low latency performance needed for many modern products that use databases for their general execution. An example of this would be stock market updates; since they need to be addressed very quickly

for consumers to properly engage with the services, a low-latency amount is necessary, and a NoSQL database can provide that. (IBM)

NoSQL is a prevalent product type, with the versatility it provides being a reason to download a certain product. A prominent example would be MongoDB, which can be used for more complex data types than other similar databases. In order to accomplish this, MongoDB uses BSON documentation rather than the typical JSON for storage purposes. BSON files allow for a much wider range of data to be stored in databases using their product, but that's not all that MongoDB is useful for. Because of its nonrelational nature, it has the capability to store much larger quantities of data, but it also has the ability integrate this data into various storage types, including cloud or hybrid storage. Further, MongoDB can be run across multiple different servers that are connected, meaning that the data can be accessed and manipulated from many different places at the same time. Finally, because of the way it stores data, MongoDB has an advantage in reducing the need for joining databases, potentially reducing costs for companies using their product. (Gillis, 2023)

Another example would be Couchbase Server, which is meant to be used for interactive web applications. The advantages Couchbase provides are, of course, its NoSQL origins, but they go far beyond this as well. This is because it is actually a merging of two other popular services, CouchDB and Membase. Both of these two products had their benefits, and upon combining their products to create Couchbase, they launched an incredibly powerful and effective tool. Couchbase is capable of JSON storage, memory caching for better persistence replication and fragmenting, and overall runs at high performance even at 100% uptime. Moreover, the flexible modeling that JSON storage provides allows for many different advantages provided by many other NoSQL databases, including horizontal scalability and simple developer integration.

A final general example comes in the form of Amazon's take on a NoSQL database, DynamoDB. Amazon is a massive company, so it only makes sense that they would have need of database tools. It only serves, then, that they would eventually create their own tool for in-house use as well as for distribution. DynamoDB is serverless, meaning that what it loses in accessibility over large distances, it gains in security. DynamoDB claims to have "… consistent single-digit millisecond performance, [and] nearly unlimited throughput and storage, …" (Amazon) Since it is an Amazon product, it of course comes with the ability to interact with other Amazon products, including

Amazon Web Services, which gives it an advantage when compared to other databases for projects that would benefit from such easy accessibility.

As discussed already, there are both benefits and demerits to having a NoSQL database over a relational database. The main advantages come down to the scale of data available and the conditions; NoSQL databases are capable of utilizing other data structures, such as JSON files, for storage. This not only brings about flexibility in the type of data available to store, but also in the amount of data storable, since it is not tied down to the schema of a table or other arrangement of data that is required for a relational database. This allows for a reduction in costs, as the horizontal scaling of NoSQL databases is much cheaper than the vertical scaling of a relational database (IBM). They also have quite a bit of flexibility, as they can easily handle large influxes of data, be it changes or additions, and code changes to go along with agile development. More than that, they also boast a hefty increase in speed for database operations.

However, NoSQL databases are not perfect, and have disadvantages in certain cases. The main concern is due to ACID across multiple documents. ACID stands for atomicity, consistency, isolation, and durability, and can be required for certain applications, which unfortunately means that a NoSQL database for those kinds of problems would be ineffective. On top of that, NoSQL databases tend to be larger, and are prone to having duplicate data, which is by design; NoSQL is designed to be optimized for queries, and not for reduction of duplication, so what it excels in for speed, it sacrifices in storage space. While this is a minor issue, as storage is relatively cheap all things considered, it is still a drawback nonetheless. Lastly, not all NoSQL databases are created equally. There are major benefits to some that leave certain features to be desired, and there isn't a catch all database that fits every problem in one use case. It is possible that for one project, you may need to use multiple databases to have a proper solution.

A popular type of database that fits into the Non-relational database definition is that of a graph database. These databases hold data that points to other positions of data with a condition, all of which can be described with a graph: a series of nodes with edges, either directed or undirected, connecting two nodes. The general look of a graph would be that of a three-dimensional object, where between different nodes there are edges that connect them, all of which happens to form an object. In terms of a database, this would be as if there are many points of data, and they have varying interactions with other data points in the database as a whole. Based on how they interact, they would have an edge between them with various properties. These graph databases come in two distinct types, one suited for

queries and general versatility known as property graphs, and the other, RDF graphs, conforms to Worldwide Web Consortium (W3C) standards to represent more complex data. (Oracle)

Graph databases can solve a wide variety of problems that require graphs to solve. This includes problems like the travelling salesman or shortest path type problems from graph theory, but they can also be used for social networking analysis, road data, and financial analyses. A specific example of this would be in a particular use case of a graph database which is graphing a social network from the connections provided by a social media site. The ideal representation of this comes from the idea of six degrees of separation, where every person can be connected to anyone through their mutual connections within six degrees. Social media networks can easily model this data using a graph database to show the connections, and show what an average connection between two random users might be. Connections aren't the only things that the graph database of a social media network would be limited to, however. There are more things tracked which could be used to create more useful tools for the platform if there is enough user demand.

Neo4j is one of the currently popular graph database tools. The reason for its popularity may stem from it being the first to properly use the term 'graph database' (Neo4j). Some of the other major benefits of Neo4j come from its performance power, as well as its general accessibility. It also is ACID compliant, something rare to be found for other marketed graph databases. The biggest draw, however, comes from the possible applications across the field of data science that Neo4j employs; as asserted by Google itself, network theory and graph databases are the future of data science (Graphable).

Other reasons for use come about as the same reasons one would use any graph database, the connections between datapoints. The amount of data and other structures available with the connectedness of data within the database itself bring about much more to be gleaned when analyzing.

Another example of a graph database is JanusGraph. Developed by Linux, it has a focus on compact serialization, rich modeling, and efficient execution of queries (Hackolade). It has extensive tools to define the schema for the connections as well as the data itself, with multiple types of data being attached to any given edge if needed. What JanusGraph includes, however, is an additional component of being able to edit the schema without interrupting the database operations themselves. A high degree of mutability and other such implications allows for a very well defined database to be compiled with less time invested in the proper setup.

Graph databases share a large amount of properties with other database types, like relational and NoSQL databases, but they each have their own strengths and weaknesses. Graph databases are considered very good at mapping problems that can be solved with network or graph theory, such as the aforementioned traveling salesman and shortest path problems, with an example of the latter being the ‘six degrees of separation’. There is also a case of modelling the data itself being a boon of this type of database, where a nicely laid out network can show the connections in a specific way to efficiently answer any question asked.

Graph databases also can incorporate many of the other aspects of other databases, in that they still have the otherwise general data collections, and can be queried without the need to look at the connections between data. This type of operation would be inefficient, however, as the database would need to parse through more data than necessary to complete a given query. Another drawback comes in the form of misuse of the database from a user perspective: If data is mislabeled or improperly designed, the entire database might end up not only confusing but wrong in the way it presents the data, creating bias or other distortions. A similar issue is that different graphs will very likely have different standards, meaning each graph database would need to be individually interpreted rather than having a simple schema to work with and a generalized interpretation. (GeeksforGeeks, 2023)

Overall, there is a good amount of information regarding the creating and manipulation of databases that are not as simple as relational databases. There are many different reasons for these differences, including the need for higher scalability with larger datasets, differing purposes in the need for data, as in graph databases, and further with the potential for improvements in performance when searching the databases themselves. NoSQL databases and graph databases provide these types of benefits in exchange for some of the benefits of relational databases, allowing for a vast world of database management systems to explore and use for various real-world issues. Their applications are only growing in number as our technological world advances, leading to a growing field of data science.

Works Cited

"Applications, Advantages and Disadvantages of Graph." *GeeksforGeeks*, GeeksforGeeks, 17 Apr. 2023, www.geeksforgeeks.org/applications-advantages-and-disadvantages-of-graph/.

"The Birth of Graph Databases: How Neo4j Built Its Product and Category." *Graph Database & Analytics*, 23 Aug. 2019, neo4j.com/news/birth-graph-databases-neo4j-built-product-category/.

FutureLearn. "What Is SQL and What Is It Used For?" *FutureLearn*, 6 Feb. 2023, www.futurelearn.com/info/blog/what-sql-used-for.

Gillis, Alexander S., and Bridget Botelho. "What Is Mongodb? Features and How It Works – Techtarget Definition." *Data Management*, TechTarget, 7 Mar. 2023, www.techtarget.com/searchdatamanagement/definition/MongoDB.

"Introduction to Couchbase - Nosql Document Database." *Today Software Magazine*, www.todaysoftmag.com/article/1506/introduction-to-couchbase-nosql-document-database. Accessed 29 Oct. 2023.

"Janusgraph." *Hackolade*, hackolade.com/help/JanusGraph.html. Accessed 29 Oct. 2023.

"NoSQL vs SQL Databases." *MongoDB*, www.mongodb.com/nosql-explained/nosql-vs-sql#:~:text=and%20fewer%20bugs.-,What%20are%20the%20drawbacks%20of%20NoSQL%20databases%3F,acceptable%20for%20lots%20of%20applications. Accessed 29 Oct. 2023.

Rangel, Derek. "DynamoDB: Everything You Need to Know about Amazon Web Service's NoSQL Database." *Amazon*, Derek Rangel, 2015, aws.amazon.com/dynamodb/.

Robinson, Sean, et al. "What Is Neo4j (Graph Dabatase)? Complete Overview of Neo4j." *Graphable*, 1 May 2023, www.graphable.ai/software/what-is-neo4j-graph-database/.

"What Are NoSQL Databases?" *IBM*, www.ibm.com/topics/nosql-databases#:~:text=NoSQL%2C%20also%20referred%20to%20as,structures%20found%20in%20relational%20databases. Accessed 29 Oct. 2023.

"What Is a Graph Database?" *Oracle*, www.oracle.com/autonomous-database/what-is-graph-database/#:~:text=of%20graph%20databases-,Graph%20Database%20Defined,are%20not%20equipped%20to%20do. Accessed 29 Oct. 2023.