Jake Holovka

CS-300

Assignment 1

Databases have been around for a long time. With the technology rapidly changing to meet demands of the companies that employ them, new designs have been implemented to keep up. NoSQL is one such design. It's commonly called "not only SQL" or "non-SQL" and it gets this name because it can support SQL like languages ("Introduction to NoSQL"). This type of database design gained popularity during the late 2000s as the storage cost for containing these systems drastically decreased (Schaefer). There are many kinds of NoSQL databases, all with a wide range of features. Each allows the users to store data in various ways so that they may be used in all sorts of business models.

Before NoSQL databases were used, SQL was a common system utilized. SQL stands for "Structure Query Language" and it is a programming language used for retrieving information from a database ("NoSQL Databases"). It is a relational database meaning it organized data into rows and columns which form a table. ("NoSQL Databases"). In this relational model, the tables would manage the various data associated with it and join it together with a unique common key. This allows it to store and retrieve the data very quickly. The downside, however, is that it requires a lot of memory and can only scale vertically meaning to add more memory to your system, you must upgrade your hardware. NoSQL doesn't have this problem because it is a non-relational system. The data is stored across multiple servers giving you the ability to scale your system horizontally ("NoSQL Databases").

There are numerous advantages for using the NoSQL database. Stated earlier, NoSQL allows for high horizontal scalability as it utilizes sharding, which is the ability to split up data and spread across multiple databases (“Introduction to NoSQL”). This is advantageous because it allows you to grow your system out, accommodating the higher data flow you experience as your company expands. It is also a cheaper option than running commercial relational database management systems as you significantly cut down the cost it takes to upgrade the hardware required for them (“NoSQL Databases”). NoSQL also has a replication feature that copies and stores your data across numerous servers allowing you to have good reliability. This ensures you will have access to your data incase one of the servers goes offline and helps protect you against data loss (“NoSQL Databases”). The speed of this database type is also suited for all sorts of big business models from mobile applications to shopping sites like amazon.

There are some disadvantages though to using this database model. For one, this type has a narrower focus than the relational databases as it is mainly designed for storage of large amounts of data (“Introduction to NoSQL”). Other problems associated with it are that it is an open-sourced system so there is no reliable standard for the database model yet. This makes it more challenging to work on since your experience with one NoSQL database model might not transition to the other. The management of these systems is also more complex than the relational counterparts and with little to no GUI support, this can lead to more issues that take longer to solve (“Introduction to NoSQL”).

NoSQL supports a wide range of types of databases, the simplest being the key-value store model. This is a schema-less model and is organized into items that contain keys and values (“NoSQL Databases”). The data is stored as hash tables where the keys have a unique identifier and the value of the keys stored could be anything like a string or a number or could even be its own key-value pair (“Types & Examples of NoSQL Databases”). An example could be a key being a shopping cart ID with the value associated with it being an array of the items in the shopping cart. This is the closest NoSQL model type that is most related to relational databases out of the bunch. This model is generally good for caching and storing session information like shopping carts. Redis and Memcached are database systems that run off this model (“NoSQL Databases”).

Another type is the document store model, and it is similar to the key-value store model. This database type stores data as documents instead of tables or graphs and contains both fields and values. The values can range from strings, numbers, booleans, arrays and to objects (Schaefer). The data is stored in JavaScript Object Notation (JSON) which helps keep the data together when it is used by application. This is beneficial because it reduces the number of times the data is translated making it more efficient (“NoSQL Databases”). SQL data is often assembled and disassembled when moving around from application to storage, so this is where the document store type gets an edge over its older counterpart (Shaefer). Another benefit to this model is that document schemas do not have to match one another. This gives the developers more room to work with since common themes like names and first names can associate with one another. The downside to this is that it can lead to more complex transactions corrupting the data (“NoSQL Databases”). Data stored in documents is also found to be easier to work with and more intuitive than that stored in tables or graphs (Schaefer). This database model is well suited for customer data management, product catalogs, and content management. MongoDB is an example of one of these document-oriented database systems.

The next type of NoSQL database models is the wide-column store. In this model, information is stored in columns which allows the users access to only the data they need (“NoSQL Databases”). This is important because it prevents you from using more memory on data you do not want. Like the key-value store, it still uses keys but instead of pointing to a particular item of data, it points to multiple columns (“Types & Examples of NoSQL Databases”). This model’s rows and columns are not fixed within a table but if they contain similar information with another column of data, they will be joined together as a column family and stored separately from other column families (Williams). The first column in each of the column families is considered the row key and this is used as an identifier. The columns that come after that have what is called a column key which is usually a name, and this is the identifier for the columns within the rows. Values and timestamps come after the column key, giving you information on when the data entry occurred or was modified (Williams). This model is useful for extreme large amounts of data and can be compressed easily to help with storage spaced. The downfall of this system is that finding and retrieving small specific sets of data, if they are in different column families, is less efficient than other models. It is also a more complex system to manage so teams who lack experience may find it difficult to run (“NoSQL Databases”). This model is used by social media website and by data analytic companies. Apache HBase and Apache Cassandra are examples of this type of database.

One of the last model types is the graph database. This model uses knowledge graphs to show relationships between the data it stores. It puts an emphasis on examining the connection between data sets rather than the individual set itself. The data is stored as nodes, edges, and properties. The nodes can represent an object, an individual and a place. The edges are used to define the relationship between the different nodes (“NoSQL Databases”). The connections are illustrated with lines and show a direction that can be one way or two way depending on the relationship between them (Dancuk). A node could be Facebook, while another is an ad agency they employ. An edge could be the relationship of their social media users with Facebook or the ad agency. The properties represent information that is descriptive of the nodes. Edges can also have properties associated with them, but this is not always the case. These graph databases are valuable because they show a conceptual view of data that more accurately depicts the real world.

Graph databases offer a lot of advantages and disadvantages to the users. They can be used to solve problems that are both practical and impractical for their relational database counter parts (Wu). They can update in real-time while also supporting the queries they are getting. You can use them in fraud detection (Schaefer). They can even be used to structure AI interfaces (Wu). Few businesses can run solely on this database system though. Graphs are not the most useful for transactional-based systems, so it usually has to be paired with another more traditional system to help support it. It also has a smaller user base than other systems leading to difficulties finding support when you need it (Dancuk).

There are products on the market that allow you to use these graphical databases without having to build your own. One such product is called Neo4j. It is an open-source project that is written in Java and Scala. It runs like most graphical database but uses a true graph model all the way down to the storage level to boost performance ("What is a Graph Database?"). It uses a declarative query language much like SQL called Cyber to read and write data to the graph. It also implements a method call Fabric that lets you shard graph data and break up bigger graphs into smaller components. This allows you to easily visualize and examine smaller portions of large-scale data. Since this product is open source, you can get the program off GitHub or use the premium version if you so wish letting a wide range of people use and experiment with the graphical database.

Another product is built by the shopping giant Amazon called Neptune which is another graphical database on the market. It uses Gremlin and SPARQL as its query languages and even supports Neo4j’s Cypher to build your queries. It boasts a high-speed functionality and the ability to encrypt your database management system. It will also automatically scale your database as the data volume grows (“Amazon Neptune Features”). In case of failure, it saves chunks of your data across numerous systems and will continually update and repair errors. It saves up to 15 replicas of your data to ensure in a complete catastrophe state, your information is still back up. Neptune offers a free tier with limited features for those interested in messing around with the system and a paid tier for more advanced users.

As data becomes more and more valuable in our society, these various database models will continue to expand and improve. With all sorts of companies from healthcare to shopping centers trying to find ways to effectively store their large sums of data, they will undoubtedly continue to be used. There are many options at your disposal for database management, all of which help provide a better understanding of our world and the connections between them.

Works Cited

"Amazon Neptune features." *Amazon Neptune*, aws.amazon.com/neptune/features/?pg=ln&sec=hs.

Dancuk, Milica. "What Is a Graph Database?" *PhoenixNAP*, 22 Apr. 2021, phoenixnap.com/kb/graph-database.

"Introduction to NoSQL." *GeeksforGeeks*, 8 Aug. 2022, www.geeksforgeeks.org/introduction-to-nosql/.

"NoSQL Databases." *IBM Cloud Learn Hub*, 6 Aug. 2019, www.ibm.com/cloud/learn/nosql-databases.

Schaefer, Lauren. "What is NoSQL." *MongoDB*, www.mongodb.com/nosql-explained.

"Types and Examples of NoSQL Databases." *BigDataAnalyticsNews*, 24 Feb. 2014, bigdataanalyticsnews.com/types-examples-nosql-databases/#:~:text=MongoDB%2C%20CouchDB%2C%20CouchBase%2C%20Cassandra,are%20Document%2Doriented%20NoSQL%20databases.

Williams, Alex. "NoSQL database types explained: Column-oriented databases." *TechTarget*, 22 Sept. 2021, www.techtarget.com/searchdatamanagement/tip/NoSQL-database-types-explained-Column-oriented-databases.

"What is a Graph Database?" *neo4j Developer*, neo4j.com/developer/graph-database/.

Wu, Mingxi. "What Are the Major Advantages of Using a Graph Database?" *TigerGraph*, 4 Nov. 2019, www.tigergraph.com/blog/what-are-the-major-advantages-of-using-a-graph-database/.