Chris Stirneman

Professor Garcia Almaguer

CS300

29 October 2023

Non- Relational Data Storage and Retrieval Systems

Traditional relational database management systems typically are based on a form of SQL. This type of system can still be found in numerous applications today, both cloud and web based. While relational database models have many pros, such as ensuring data integrity and accuracy, preventing data duplication, inconsistency, and corruption, and supporting complex queries and operations ("What are the pros", 2023), there are also several cons. Some issues with relational databases include difficulty scaling horizontally, meaning more servers need to be added to scale and distribute the workload, along with inflexibility of data model changes, and the struggle to store unstructured or semi-structured data like images and videos ("What are the pros", 2023). In the recent years, a new type of data retrieval and storage system has emerged, called NoSQL. This type of system has been created to address several of the issues associated with the typical relational databases. In this paper, I will look at several questions surrounding NoSQL databases, including what problems NoSQL aims to address, the pros and cons of NoSQL, and graph databases.

NoSQL is short for "Not only SQL", and is a different type of approach to database design than the typical relational database model. The NoSQL database design enables the storage and querying of data from outside of the typical relational database structure ("What are NoSQL", n.d.). While NoSQL can store data that is found in relational databases, the difference is how the data is stored. With NoSQL databases, data is stored within one large data structure, like a JSON document, rather than being stored in the typical tabular structure of relational databases ("What are NoSQL", n.d.). This difference is very beneficial when it comes to being able to scale and manage large unstructured datasets rapidly, since this type of design does not require a schema ("What are NoSQL", n.d.). Another important characteristic of NoSQL databases is that it is a type of distributed database. This means that information of data is copied and stored on multiple servers, ensuring that data is reliable and available even if some servers go offline. In todays technological environment, companies are relying more on the ability to store, manage, and retrieve data rapidly. NoSQL databases are an ideal approach to handling this dynamic and rapidly changing data environment, which makes them an ideal choice for applications with large amounts of user data where the schema is not predefined, and for applications that need to store a large amount of unstructured or semi-structured data.

NoSQL databases were designed to combat several of the issues that relational databases experience, primarily focusing around the issues of scalability, flexibility, and performance. With typical relational databases, handling large amounts of data can express the issue of scalability. Relational databases typically do not support horizontal scaling. Instead, relational databases rely on vertical scaling, which refers to adding more power to an existing machine or migrating to a larger, more expensive server ("NoSQL vs. SQL", n.d.). NoSQL databases, however, have the ability to scale horizontally. This refers to adding more servers to the database cluster, allowing the ability to better handle a large volume of data quickly. Another issue that relational databases experience are having inflexible schemas. With relational databases, schemas are usually predefined, meaning that it can be hard to modify once a database is in use ("NoSQL vs. SQL", n.d.). On the other hand, NoSQL databases offer more schema flexibility, allowing the ability to add data without defining its structure. This can be very beneficial when it comes to scenarios where the data structure is unknown or changing in advance. Queries may also be an issue with some large relational databases. Data in SQL databases may require joining data from multiple tables ("NoSQL vs. SQL", n.d.). As the dataset becomes larger, joining data together from different tables can be less efficient. However, NoSQL databases can be an ideal choice to combat this issue. NoSQL databases can be optimized for many cases and patterns, and can excel when querying a large amount of data is needed. NoSQL databases typically don't require joins, making the queries faster ("NoSQL vs. SQL", n.d.).

With the many benefits that a NoSQL designed database can bring to an application, it is no surprise that there are several products out there for engineers and anyone wanting to design a NoSQL database. One of the most popular NoSQL database products is MongoDB. MongoDB is a document based NoSQL database. This means that MongoDB allows developers to store data in very flexible, JSON-like documents ("What is MongoDB", n.d.). This allows data fields to vary from each document, as well as data structure, to make any changes easy over time ("What is MongoDB", n.d.). MongoDB is also considered a distributed database, meaning that the data is partitioned across multiple nodes ("What is MongoDB", n.d.). Each node can then store a portion of the data, helping to achieve more scalability and improve the performance of the database. Distributed databases are also important in achieving high availability of data, as the partitioning of data allows for more uptime ("What is MongoDB", n.d.). Another popular NoSQL database product is Cassandra. Cassandra is a wide-column stored NoSQL database, meaning that data is stored in tables as sections of columns rather than rows ("Cassandra Basics", n.d.). Cassandra also has many of the same features like MongoDB, including being a distributed database. Cassandra also uses partitions to split data into

smaller sections, making it easier to use and apply (“Cassandra Basics”, n.d.). Another feature of Cassandra is data replication. Replication allows data to be replicated to multiple replica nodes. This allows for additional reliability and fault tolerance (“Cassandra Basics”, n.d.). Amazon DynamoDB is another popular NoSQL database, which uses key-value design. Similar to other databases, DynamoDB works by creating a table of a collection of items (“Amazon DynamoDB”, n.d.). Each item in the table has its own primary key, but DynamoDB offers the option to have both global and secondary indexes (“Amazon DynamoDB”, n.d.). This allows developers to query data from the table using a secondary key. Similar to the other NoSQL database products, Amazon DynamoDB also provides data replication, meaning the availability of their databases are very high (“Amazon DynamoDB”, n.d.). These are just some of the most popular NoSQL databases products out, however, as NoSQL continues to grow, more and more products will become available.

NoSQL database design comes with many advantages over relational databases. Some of the top advantages include scalability, schema flexibility, and a variety of data models. With NoSQL, databases are easier to scale. NoSQL can be scaled horizontally instead of vertically, across distributed clusters of hardware. Since NoSQL lacks a specific data structure, each item can be self-contained and independent, resulting in data not having to be linked (“What are the pros”, n.d.). This allows NoSQL databases to handle large volumes of data by adding more nodes to the system, allowing for easy scalability. NoSQL databases also have more flexibility with datatype schemas. NoSQL allows you to store and retrieve data regardless of the predefined schema (“What are the pros”, n.d.). This is an advantage over relational databases as applications can adapt better to new types of data without having to adjust the schema, allowing unstructured and semi-structured data to be added to the database. Another advantage NoSQL databases have over relational databases is that they can support a variety of data models. Many NoSQL products, like the ones I have mentioned, support different models such as document, key-value, and graph databases. This allows developers the ability to choose the best model for their application, which is something that relational databases can struggle with (“What are the pros”, n.d.).

While NoSQL databases have several advantages over relational databases, there are still some disadvantages. One major disadvantage to NoSQL databases is that queries are less flexible than relational databases. With NoSQL databases, you won’t be able to use many types of standard queries like you can with SQL databases. NoSQL databases, for example, cannot enforce uniqueness for keys within documents like relational databases (“What are the pros”, n.d.). This can become an issue for applications that require enforcement of unique

key values, meaning that advanced queries can be challenging for developers. Another disadvantage that NoSQL databases face is data integrity. NoSQL databases may lack built-in integrity constraints that relational databases have. This can be an issue as additional effort would be required to ensure the application and database runs correctly (“What are the pros”, n.d.).

A graph database is another popular type of NoSQL database. With graph databases, data is stored in nodes and relationships, instead of tables like relational databases. This makes graph databases more suited for real-world scenarios as flexibility is better since it is not constrained to rigid structures. Graph databases work by using graph algorithms, graph query language and by storing data as entities, represented by nodes, and relationships between the entities, represented by edges. These algorithms analyze relationships and behaviors between data. They are able to traverse paths between nodes and edges, helping to identify patterns and paths that connect data. Graph query languages are used to interact with the database and provide an interface that allows users to ask questions to see the relationships between data. This makes it easy to process complex queries quickly.

Graph databases were designed to solve many situations such as social networking, fraud detection and recommendation engines that relational databases struggle with. Social networking is a popular use case for graph databases as it makes it easy to identify relationships. By representing users as nodes and relationships as edges, graph databases make it easier to identify social interactions. Graph databases allow developers to better understand social platform interactions, behavior, and engagement, like friendships and interaction patterns, making it easier to increase user engagement and offer a better experience to users. With relational databases, multiple tables and join operation are typically needed to identify patterns, making relational databases less efficient. Graph databases are also very useful for fraud detection. This is especially useful for transactions, where graph databases can use relationships to process transactions details and user accounts to detect any anomalies. Patterns of fraudulent activity can be easily identified when using transactions and connections as nodes and edges. By identifying clusters of suspicious activities, companies can locate fraudulent networks, making customers safer against fraud. Relational databases, once again, would require complex queries and multiple join operations to identify fraud. This is less efficient, especially as the database continues to grow. Graph models are also a good choice for applications that provide recommendations. With graph databases, you can store relationships in different categories like interests, friends and purchase history. This makes it easier to identify patterns between users with similar interests and produce algorithms that can then recommend products or services to a user. Relational databases may struggle with

recommendation engines as they often require data normalization, which can make it difficult to identify user preferences. Often times with relational databases, incorporating new factors may require schema modification, which can be difficult. Therefore graph databases are the best option when it comes to recommendation engines and the other examples provided and solves many of the issues relational databases experience.

There are several graph database products available for developers. One of the most popular graph database products is Neo4j. Neo4j works by using a graph model to represent data as nodes, relationships, and properties. Nodes represent entities, relationships represent the connection between entities, and properties represent information about the nodes and relationships ("What is a graph", n.d.). Neo4j also uses Cypher, which is a query language designed for querying graph data. This language allows developers to perform more complex operations, find patterns between data, and filter data to make it easy to understand the graphs relations ("What is a graph", n.d.). This product also makes use of several built in graph algorithms through their libraries, which makes it easier for developers to explore and analyze their data ("What is a graph", n.d.). Another popular graph database product is Amazon Neptune. Amazon Neptune is a graph database that is serviced by Amazon Web Services (Vogt, 2000). Like Neo4j, Amazon Neptune also uses nodes and edges to store data. Neptune, however, also supports the Resource Description Framework (RDF) model, which can be used to describe resources on the web (Vogt, 2000). Neptune uses highly available storage systems to replicate data automatically across several Availability Zones and supports low latency read replicas to provide reliable and high performance (Vogt, 2000). Neptune also uses a shared storage architecture that automatically grows in size as the database grows. Neptune supports several query languages, such as Gremlin and SPARQL, which increases the versatility of queries you can perform (Vogt, 2000). While there are several graph databases products available, these two are some of the most popular choices. These products enhance the power that graph databases can provide for applications and provide an easy and reliable database for developers.

Graph databases come with several advantages such as flexibility, performance, and scalability. With graph databases, there is more flexibility than relational databases because there is not a predefined schema ("What are the advantages", 2023). This means that they can easily adapt to different data types. Graph databases also are great for quickly querying data dynamically as it happens thanks to the graph algorithms that power it. With these algorithms, graph databases can examine the nodes and edges that are relevant to the developer, instead of traversing through the entire database ("What are the advantages", 2023). Another advantage of graph databases is it can easily be scaled.

Graph databases can partition large amounts of data into smaller graphs, while spreading them out across multiple servers (“What are the advantages”, 2023). In turn, this reduces latency and the workload of the servers, while providing high availability. Graph databases can also provide a great deal of integrity by using replication, transaction, and synchronization methods, like ACID (“What are the advantages”, 2023). For these reasons, graph databases have an advantage over relational databases. Graph databases do come with a few disadvantages. One being that the complexity of creating a graph database can be more difficult than relational databases. Graph databases require a different query language and API than relational databases (“What are the advantages”, 2023). These query languages have their own syntax, which can make the learning curve steeper when dealing with graph databases (“What are the advantages”, 2023). Graph databases also are limited in their capabilities of handling concurrency, security and backup of data (“What are the advantages”, 2023). For these reasons, developers using graph databases need to have a good understanding of the product they are using as well as the concepts and algorithms that make a graph database work.

In this paper, I have covered NoSQL and graph databases. With NoSQL, data is stored in a singular data structure rather than a tabular structure, which is how relational databases store data. NoSQL is also more advantageous than relational databases when it comes to flexibility, as no predefined schema is required. This allows unstructured and semi-structured data to be added easily and allows the database to adapt to changing data. Querying large datasets may also be easier with NoSQL as it typically does not require complicated join methods, like relational databases do. Several products are available to developers, with some of the most popular being MongoDB, Cassandra, and Amazon DynamoDB. Graph databases are also another popular choice among developers for their ability to analyze relationships between data using nodes and edges. Graph databases excel at analyzing large amounts of data and discovering relationships between the data quickly without traversing the entire database. Graphs also have an advantage with data availability and replication as the data can be spread out across multiple servers to reduce latency and increase availability. Many products are also available for graph databases, such as Neo4j and Amazon Neptune. NoSQL and graph databases were designed to solve some of the problems relational databases have. While these types of databases are still relatively new, I expect more and more companies to adopt these types of databases.

References

Amazon dynamodb features | nosql key-value database | Amazon Web Services. (n.d.-a). https://aws.amazon.com/dynamodb/features/

Cassandra Basics. Apache Cassandra. (n.d.). https://cassandra.apache.org/_/cassandra-basics.html

Management, D. (2023, August 28). *What are the pros and cons of using relational vs. non-relational databases?*. Relational vs. Non-Relational Databases: Pros and Cons. https://www.linkedin.com/advice/0/what-pros-cons-using-relational-vs-non-relational-1e

NoSQL vs SQL databases. MongoDB. (n.d.-a). https://www.mongodb.com/nosql-explained/nosql-vs-sql#differences-between-sql-and-nosql

Vogt, G. (2000). *Neptune*. Amazon. https://aws.amazon.com/neptune/features/?pg=ln&sec=hs

What are NoSQL databases?. IBM. (n.d.). https://www.ibm.com/topics/nosql-databases

What are the pros and cons of nosql. Adservio. (n.d.). https://www.adservio.fr/post/what-are-the-pros-and-cons-of-nosql#:~:text=The%20top%20advantages%20of%20NoSQL,less%20mature%2C%20less%20flexible%20queries.&text=Queries%20are%20less%20flexible.&text=NoSQL%20isn't%20designed%20to%20scale%20by%20itself.

What is a graph database? - aws.amazon.com. (n.d.-b). https://aws.amazon.com/nosql/graph/

What is a graph database? - getting started. Neo4j Graph Data Platform. (n.d.). https://neo4j.com/docs/getting-started/get-started-with-neo4j/graph-database/

What is mongodb?. MongoDB. (n.d.-b). https://www.mongodb.com/what-is-mongodb

www.linkedin.com. (2023, March 22). *What are the advantages and disadvantages of using a graph database for social network analysis?*. Graph Database for Social Network Analysis: Pros and Cons. https://www.linkedin.com/advice/1/what-advantages-disadvantages-using-graph-database