Patrick Valsted

CS300-ON

Non-Relational Data Storage and Retrieval Systems

While SQL databases may be the most common choices, NoSQL databases are rising in popularity. Unlike SQL databases, which are strictly relational in nature, NoSQL databases are generally non-relational, distributed, flexible, and scalable. However, most NoSQL databases today employ some SQL and support relational data. This allows you to combine the usefulness of both models. Because of this, “NoSQL” doesn’t mean “No SQL” nowadays as the term implies, it means “Not Only SQL.” Since NoSQL models aren’t strictly relational, the flexibility allows data to be stored in different ways, rather than the pre-defined schemas traditional databases such as SQL-structured ones use. The flexibility also allows for adaptability in data structures and growing amounts of data, making them a popular choice for database management systems today.

SQL databases are helpful and quick, and they are still very commonly used today. However, the main limitation they had was that they only could process small amounts of STRUCTURED data, which worked out for a while. As Internet traffic grew, a need for a new database was quickly apparent. NoSQL databases are much more efficient at handling the enormous amounts of data Internet users bring with them, and the fact that they are capable of processing data in many different structures is a huge advantage as well. In the days of only SQL, large amounts of people (meaning bigger waves of data) flooded a company’s page at once and these databases couldn’t handle the strain, causing them to either break completely or require tedious amounts of maintenance. It’s especially important to consider these problems today, where reportedly “63% of enterprises and service providers today are managing storage capacities of at least 50 petabytes—and more than half of that data is unstructured.” (Lacefield) It’s clear huge amounts of data are being processed and in order to do this efficiently, we need to use a system that can handle it and doesn’t require a structure such as NoSQL. NoSQL databases were also developed to tackle the high-maintenance issues developers ran into with traditional SQL databases. These databases require far less work than their SQL counterparts, and many companies choose to use these because that also translates to them having to spend far less money on the work required to maintain them.

There are many examples of NoSQL products out there. One of the frontrunners is MongoDB, an open-source NoSQL database management system. This product specializes in managing humongous (the word that inspired the name) amounts of data, making it good for organizations that need to process a lot of data very quickly. Some of the companies that use MongoDB include MetLife, Forbes, Nokia, eBay, and Shutterfly. Established in 2009, MongoDB uses multiple drivers for multiple programming languages, including Java, Python, and anything in the C language family. These drivers transmit data from the application to the mongo server and then one of many storage engines. MongoDB relies heavily on JavaScript Object Notation (JSON) documents to store data, which is why it is capable of storing amounts of data much larger than relational systems like SQL databases. It also uses BSON formatted documents to store media. However, MongoDB also employs aspects of relational systems, such as the ability to easily locate data by using primary and secondary indexes. Because it uses a consistent structure, data can be read immediately. Even so, the product also allows for flexibility and scalability, which are non-relational traits that set it above the relational databases out there. Something else that makes it appealing to companies is that it prides itself on the fail safes that exist in the event of server crashes. Because it replicates data across multiple servers, it's extremely rare that any data stored on these servers would ever be lost. Due to its capabilities of dealing with large sets of data while making it easily retrievable, it's no wonder MongoDB is used by popular companies.

Another popular NoSQL product is Apache CouchDB. This database is also open-source, meaning users can access it for free. This one is focused on user control, meaning it's flexible for the needs of different users for different purposes. Different businesses can easily adapt this code to integrate into their already-existing databases for any specific purposes they can come up with. The database is web-oriented, and like MongoDB it relies on JSON documents. It uses JavaScript for indexing, transforming, and combining documents while using HTTP for its API. It's schema-free, which is a NoSQL trademark, and this makes data management much simpler.

Oracle also has a NoSQL database. Unlike the previous two examples, in addition to using JSON documents, this database supports key-value and column data models. This one can also exist on-site in addition to over the cloud. The developers boast super quick response times (within milliseconds) and duplication (just like MongoDB). It's designed to have simple, user-friendly access and support for a wide variety of business models. While not open-source like the other two, Oracle NoSQL Database offers low pay-per-use pricing, and being a big

and established name, customers can rest easy knowing their data will stay secure. Additionally, development tools allow support for many languages such as Python and Java and integration with programs like IntelliJ and Eclipse.

As for NoSQL databases as a whole, there are definitely advantages and disadvantages. One of the main advantages of going NoSQL is that these databases are flexible and scalable. They way they store data makes them super viable for cloud-based applications. While SQL databases solely scale data vertically, NoSQL databases offer horizontal scaling as well, which is possible due to lack of data structure. This allows every item to be contained and stored on separate servers. Another clear advantage is the ability to work with different data types and structures. This means those who are maintaining the databases don't constantly need to modify the database or data to adapt to structured rules, as the database will already be set up to do so. This system allows data to not only be processed in an unstructured manner, but in a quick and efficient manner. This saves on both time and money. The general lack of maintenance is something else that makes NoSQL databases appealing, as they do a lot of those tasks (such as data replication) automatically. Because they are so efficient at this data replication, that means the NoSQL databases don't suffer from much downtime. When one node is down, the database will access the required data from a different node. Another advantage of NoSQL is simply the fact that it can process such a large amount of data efficiently, which is something relational databases struggle with. This makes them ideal for storing immense databases, such as analytics that need to be updated in real-time. In addition to being able to store large amounts of data, NoSQL databases can have a much wider range of data distribution. SQL and other relation databases rely more on centralized applications that rely on location, whereas the NoSQL databases can be stored across the cloud in different global regions. There are many more advantages to using these databases.

There are, of course, disadvantages to using NoSQL databases. While SQL focuses on queries, NoSQL lacks much functionality in that realm. Therefore, fewer safeguards exist when it comes to using unique key values or really strict enforcement of any kind. This leads to more work on the end of the application developer, and less for those managing the database. This also means there is a lot less consistency in the data, which is bad news for transactions that are more immediate like online purchases. Also, while one advantage of NoSQL is its scalability, work and resources will have to be put in to make it that way, which can be costly. In general, there is not nearly as much documentation or information out there when it comes to NoSQL databases because it's a much newer and less widely accepted concept today. It can be much harder to figure out problems with NoSQL and it's not as highly

regarded as the traditional SQL database simply because it hasn't been around as long. As with anything, whether a user or business uses SQL or NoSQL depends more on their specific needs and capabilities; there is not one option that is overall better or worse.

There are said to be four categories of NoSQL databases: document databases, graph databases, key-value stores, and wide-column stores. Document databases store data in document format using collections of key-value pairs in JSON files. There is some flexibility here, as users don't need to define data or reference any universal schema. Document databases are used by MongoDB and CouchDB as mentioned before, and they are also good for content management and mobile application data management, including blogging, web analytics, and e-commerce.

Key-value stores are potentially the simplest category of NoSQL databases, as they are simply key-value pairs organized on hashtables. Each value has its own unique identifier that will return very quickly when the key is entered. This makes them super scalable, and if you are storing a ton of simple raw data, this is the way to go. They are good for applications such as dictionaries and collections, and specific uses include shopping carts on e-commerce sites, session management and caching in web applications, and managing session details in multiplayer gaming sessions. Some of the databases that use key-value stores include Riak, Aerospike, and DynamoDB.

Wide-column stores rely on tables with columns and rows. Each column is treated independently, and also stored separately on the disk. The primary use for this type of NoSQL database is to store large sets of data distributed across numerous computers. Keys also exist for this type of database, and they point to multiple columns. These databases are usually used for data warehouses, business intelligence, recommendation engines, and fraud detection. Specific examples of databases that use this are Accumulo, Amazon SimpleDB, Cassandra, and HBase.

Graph databases organize data into nodes and edges that extend out from them. These edges represent the relationships between the nodes, and both of these values are given unique identifiers. Graph databases are treated as multi-relational. The relationships are already a part of the database, so the process is sped up because this information is already calculated. This graph model is very flexible and can be stretched across as many systems as the users want. This system can be as complex or as simple as the user wants, and there is no cap on how many relationships a node can have. This type of NoSQL database has been useful for logistics, social media platforms, making reservations, and customer relationship management.

Graph databases are possibly the most unique of the four categories of NoSQL databases. They were designed to focus on relationships between things. While most other databases focus solely on storing lists of data, these were developed in order to handle the relationships between the points of data in a quick and efficient manner because the relationships are stored as data right next to the nodes. There are numerous real-world applications for these databases that the other types of NoSQL databases are inefficient at dealing with. One of the applications at the forefront is discovering patterns, especially those that are otherwise fairly hidden. This specifically allows social media companies to detect the activities of bots, for example. Since the analytics that use graph databases as their backbone detect what normally happens on a user's account, it can differentiate between an actual user and a bot account. Another real-world problem these databases aim to tackle is route optimization. This can apply to anything comparing the number of links between nodes. Using these databases you can determine the shortest route between points on a map, a suitable employee for a job based on optimal skills and availabilities, and the best machinery to use for a job based on specific parameters.

Companies also use graph databases to tackle fraud detection. When you look at information based on relationships, you can see things such as multiple people using the same email address or credit card number. If this information is associated with a fraud case it can be quickly identified, whereas any other database not optimized for relationships would have a much harder time finding those links. Recommendation engines also use graph databases. Because you can store data and link it to a node representing a specific person, you can get a better picture of what this person likes or what they've purchased in the past. This can be used to make recommendations used for advertising or suggestions for new friends on social media sites. One other example of something graph databases are being used to solve is the realm of knowledge management. Relationships between words, for example, can be used in machine learning algorithms to distinguish between two words that are spelled the same but have two entirely different meanings.

One specific product that uses a graph database is Neo4j.Written in Java and Scala, this product is completely open-source and free to use since 2007. It is a native graph database, which means every level of the process implements a true graph model. It uses a declarative query language called Cypher, similar to SQL but optimized for graphs. It also uses flexible property graph schema, and it's possible to add new relationships at any time because the technology is adaptable. It also has drivers for popular programming languages such as Java and

Python. This product is used by educational institutions and large enterprises such as Adobe, eBay, and Microsoft. Also making use of Neo4j is the U.S. Army. The Army has many uses for this type or technology, such as forecasting the need for replacement parts, general budgeting, and planning for contingencies. According to Neo4j, the product is used for a 3TB database with over 5.2 billion nodes and 14.1 billion relationships. One of the representatives from the Army stated that the database load time was reduced from 60 hours to seven or eight thanks to this product.

Another product that uses graph databases is Amazon Neptune. This engine is built to store billions of relationships and query very quickly. It supports multiple query languages and allows you to build queries. This product is actually serverless, which Amazon claims will save up to 90% on database cost. The product is also designed to be durable, with continuous backups, self-healing storage, and point-in-time recovery. It's self-sufficient, so you don't have to do tasks such as backups and software patching yourself. One example of something Amazon Neptune specifically boasts it can do is use machine learning models to not only identify fraudulent transactions but query a graph of all those involved with the fraud. It claims to also use machine learning to improve the accuracy of predictions by over 50%. With this product, you are choosing a big-name brand, and that tends to suggest reliability. Some customers of Amazon Neptune include Cox Automotive and ADP.

Graph databases have more situational pros and cons. There are a lot of advantages to using a graph database. As with the other categories of NoSQL, these databases scale horizontally, so it's efficient to spread out the load between multiple machines to increase efficiency and the data capacity. They are fantastic at handling complex relationships where their counterparts fail, making them the best tools for specific jobs. They can be updated in real-time no matter how large the imported data is, and queries can be made while that's happening. They can also navigate data efficiently by using graph traversal, which is vital because these databases can get very complex. The cons are very situational. You wouldn't use this type of database for any set of data that doesn't need to be analyzed as part of a bigger picture. Structured data would fit much more nicely into any of the other three categories of NoSQL databases, so trying to use a graph database with these sets of data would be overly complex and not make any sense. The only other disadvantage to using a graph database other than using it in an inappropriate situation is that these are the most complex type of NoSQL database, so novice users will probably struggle with them and more knowledge would be required.

Works Cited

Adservio. (2021, March 15). What are the pros and cons of nosql. Adservio. https://www.adservio.fr/post/what-are-the-pros-and-cons-of-nosql#el2

Amazon. (n.d.). What Is a Graph Database?. Amazon. https://aws.amazon.com/nosql/graph/

Ashtari, H. (2022, October 18). NoSQL basics: Features, types, and examples. Spiceworks. https://www.spiceworks.com/tech/artificial-intelligence/articles/what-is-nosql/

Ayusharma0698. (2023, October 13). Introduction to NoSQL. GeeksforGeeks. https://www.geeksforgeeks.org/introduction-to-nosql/#

BasuMallick, C. (2022, August 17). What is mongodb? features and use cases. Spiceworks. https://www.spiceworks.com/tech/cloud/articles/what-is-mongodb/

Lacefield, J. (2018, August 29). The evolution of NoSQL. DataStax. https://www.datastax.com/blog/evolution-nosql

Mullins, C. S., Vaughan, J., & Beal, B. (2021, April 8). NoSQL (Not Only SQL database). Data Management. https://www.techtarget.com/searchdatamanagement/definition/NoSQL-Not-Only-SQL

Neo4j. (2022, July 26). Neo4j Keeps the Army Running by Tracking Equipment Maintenance. Neo4j.com. https://neo4j.com/case-studies/us-army/?ref=web-solutions-government

Neo4j. (n.d.). What is a graph database?. Neo4j.com. https://neo4j.com/developer/graph-database/

Sharma, J. (2023, March 29). Graph databases vs relational databases: What and why?. DEV Community. https://dev.to/documatic/graph-databases-vs-relational-databases-what-and-why-5d6g#pros