**nosql_graph_overview_LHS**

Lawrence Schoch

Department of Computer Science, Bellarmine University

CS-300-ON: Database Management Systems

Dr. Nathan Norton

March 14, 2022

## What does NoSQL mean?

Foote provided a clear, concise answer to this question: "The acronym NoSQL was first used in 1998 by Carlo Strozzi while naming his lightweight, open source “relational” database that did not use SQL. The name came up again in 2009 when Eric Evans and Johan Oskarsson used it to describe non-relational databases. Relational databases are often referred to as SQL systems. The term NoSQL can mean either “No SQL systems” or the more commonly accepted translation of “Not only SQL,” to emphasize the fact some systems might support SQL-like query languages" (Foote, 2018). In other words, NoSQL refers to a data store that uses a programming language other than SQL to construct its queries. In practice, however, any non-relational database may be thought of as NoSQL even though many of these do support SQL compatible queries.

## What types of problems were NoSQL databases designed to solve?

The explosion of Web traffic in the 1990's saw the generation of ever-increasing amounts of data, data that had to be stored and processed. Much of this data was unstructured, e.g., images and human text such as IM messages, social media posts, and emails. Traditional relational database management systems relied on structured data, which reduced data duplication. Less duplication translated into less data and thus lower storage costs. (At a time when storage costs were significant.) This came at a price, however. Building structure into data is expensive in terms of time and manpower. And with the expense comes other drawbacks including the risk of losing access to some of the information contained in the data, and decreased flexibility in processing.

Concurrent with the data explosion of the 1990's storage costs began to decline. Less constrained by storage considerations, database designers were freer to develop systems that could process large amounts of unstructured data at reasonable speeds, without losing any of the information contained, and without succumbing to the rigidity inherent in traditional relational systems. According to the MongoDB web site:

> NoSQL databases emerged in the late 2000s as the cost of storage dramatically decreased. Gone were the days of needing to create a complex, difficult-to-manage data model in order to avoid data duplication. Developers (rather than storage) were becoming the primary cost of software development, so NoSQL databases optimized for developer productivity. (MongoDB, *What is nosql? NoSQL databases explained*)

In summary, NoSQL databases were designed to solve the problems of handling large amounts of unstructured data, even Big Data, at faster processing speeds, and without having to make the data conform to a rigid, structured data model. As a result, these databases are scalable, flexible, and highly performant.

What are three examples of NoSQL database products and how do each of them work?

1. "MongoDB is primarily a document store containing JSON-like structures and a JavaScript interface. It's known for being user-friendly (less administration overhead), performant for simple queries, and flexible thanks to its NoSQL underpinnings. Great for hierarchical data storage, it also supports familiar relational concepts from indexing, to aggregation, to some measure of ACID compliance.[1] Like MySQL, it is compatible with many platforms and programming environments, despite relative recency" (Talend, *SQL VS NOSQL: Differences, databases, and decisions*).
2. DynamoDB, by Amazon, is a NoSQL database product that is based on the key-value model. This is the most basic type of database in which data is stored as a key-value pair where the key is used to access the data. This simplicity is an advantage - because of the direct data access, reading and writing is fast. However, the simplicity can also be a disadvantage because it restricts the complexity of the data that can be stored. According to Amazon, DynamoDB is a "fully managed, serverless, key-value NoSQL database designed to run high-performance applications at any scale. DynamoDB offers built-in security, continuous backups, automated multi-region replication, in-memory caching, and data export tools" (Rangel, 2015).
3. Cassandra is a NoSQL wide-column database management system. It was originally developed at Facebook to power their Inbox search feature. Facebook released it as an open-source project on Google code in July 2008. In 2009 it became an Apache incubator project graduating to top-level project in 2010. The wide-column data format stores data in tables, rows, and dynamic columns. Cassandra is essentially a hybrid between a key-value and tabular database management system. The rows are partitioned, and the consistency of a row is tunable (presumably as a trade-off in performance). Rows are organized into tables and the first component of a table's primary key is the partition key. Tables may be created, dropped, and altered at run-time without blocking queries and updates. (Wikimedia, 2022)

What are the advantages and disadvantages of NoSQL databases?

Major advantages of NoSQL databases:

1. Flexible data model - they can store and combine any type of data, structured and unstructured.

[1] ACID is an acronym that stands for four desirable database properties: atomicity, consistency, isolation, and durability.

2. Evolving data model - these databases allow you to dynamically update the schema while assuring no interruption or downtime of the application.
3. Elastic scalability - the ability to expand to accommodate any type of data growth without adding to cost.
4. High performance - measured in terms of throughput and latency.
5. Open source - no expensive licensing fees plus they can run on inexpensive hardware making their deployment cost-effective. (Admin, 2021)

Major disadvantages of NoSQL databases:

1. Lack of standardization - the design and query languages of NoSQL databases vary widely. NoSQL languages lack the standard interface which SQL provides, so more complex queries can be difficult to execute.
2. Backup issues - it can be hard to get a complete data backup of a NoSQL database.
3. Consistency is secondary to scalability and performance. (Admin, 2021)
4. Support and communities - SQL databases represent massive communities, stable codebases, and proven standards. Multitudes of examples are posted online, and experts are available to support those new to programming relational data. NoSQL technologies are being adopted quickly, but communities remain smaller and more fractured. However, many SQL languages are proprietary or associated with large single vendors, while NoSQL communities benefit from open systems and concerted commitment to onboarding users. SQL is available to most major platforms, from operating systems to architectures and programming languages. Compatibility varies more widely for NoSQL, and dependencies need to be investigated more carefully. (Talend, SQL VS NOSQL: Differences, databases, and decisions)

## What is a graph database?

"A graph database stores nodes and relationships instead of tables, or documents. Data is stored just like you might sketch ideas on a whiteboard. Your data is stored without restricting it to a pre-defined model, allowing a very flexible way of thinking about and using it" (*What is a graph database? - developer guides*).

Amazon's Menegasso summarized graph structure clearly and concisely: "Graph databases use nodes to store data entities, and edges to store the relationships between the entities. An edge always has a start node, end node, type, and direction, and an edge can describe parent-child relationships, actions, ownership, and the like. There is no limit to the number and kind of relationships a node can have" (Menegasso, 2018). Nodes are often depicted as circles, or

some other geometric shape and they are connected by lines (edges). Both nodes and edges can have properties that can be used to facilitate the process of querying the database. It bears repeating that a graph database does not restrict the user to a rigid, pre-defined schema. Not every data object or relationship has to have a value for every property and properties can be added or modified after the database has been created and in production.

What types of problems were graph databases designed to solve?

Much of the vast stores of data that have been generated since the explosion of web traffic beginning in the 1990's does not fit into a neatly structured table paradigm, nor is the relational database ideally suited to harvesting the information contained in such data. With some data, the knowledge to be gained is stored in the relationships between the data objects, more so than in the data objects themselves. Graph databases are all about relationships, to the point that relationships are as important, if not more important, than the data objects themselves. "While existing relational databases can store these relationships, they navigate them with expensive `JOIN` operations or cross-lookups, often tied to a rigid schema" (*What is a graph database? - developer guides*). With graphs, the relationships are objects along with the data entities. This means that traversing and querying the graph database is fast because the relationships do not have to be generated at runtime - they are already stored with the data.

Emil Eifrem, CEO of Neo Technologies, observes that "knowledge is all about relating unknown concepts to previously known concepts. So, knowledge is all about relationships and we build technology around that insight – that what matters are the relationships in data and that is what today is called the graph database approach" (Dryburgh, 2017). Eifrem further notes that the information to be gleaned from modern data is akin to the information (knowledge) contained in the human brain. The human brain could not function without its connections, the billions of synapses that connect one neuron to one or many other neurons. Enter graph databases, where knowledge is gained by observing patterns in relationships. It is interesting that Eifrem, the touted creator of the graph database, was inspired Mark Zuckerberg, CEO of Facebook (now Meta). Eifrem was struggling to find just the right name for his new database technology "and then Mark Zuckerberg came along, and he said on Facebook's front page ‘we are a utility for the social graph’" (Dryburgh, 2017). Out of this comment a name was born.

What are two examples of graph database products and how do each work?

Based on Google search results, Neo4j appears to be the most popular graph database platform available, and this is not surprising since the CEO of Neo Technology was the creator of the graph database concept. "Neo4j is an open-source, NoSQL, native graph database that provides an ACID-compliant transactional backend for your

applications that has been publicly available since 2007" (*What is a graph database? - developer guides*). "Native graph database" means that the graph model is implemented without abstraction, i.e., the model is maintained without relying on layering on top of another technology. Neo4j is written in Java and Scala.

One valuable feature of Neo4j is that data objects and their relationships can be tagged with labels to designate their role. For example: `Person`. Node labels may also attach metadata (such as index or constraint information) to certain nodes. Nodes can hold any number of key-value pairs, or properties. For example: `name`. In contrast to relational databases, it is much easier to add new properties or modify existing properties at any time during the life cycle of the database.

In Neo4j (and graph databases in general), relationships provide directed, named connections between two node entities. For example: Person `LOVES` Person or Person `IS_MARRIED_TO` Person). Relationships always have a direction, a type, a start node, and an end node, and they can have properties, just like nodes. Nodes can have any number or type of relationships without sacrificing performance. Although relationships are always directed, they can be navigated efficiently in any direction. (*What is a graph database? - developer guides*)

Although important patterns are often apparent on simple visual inspection of the data diagram, Neo4j does provide a robust declarative query language called Cypher which uses labels and properties to navigate the database. Cypher queries are touted as being easier to read and write than SQL queries and this is a major attraction of the Neo4j platform, along with the faster traverses seen with graph databases in general. See *Figure 1*.

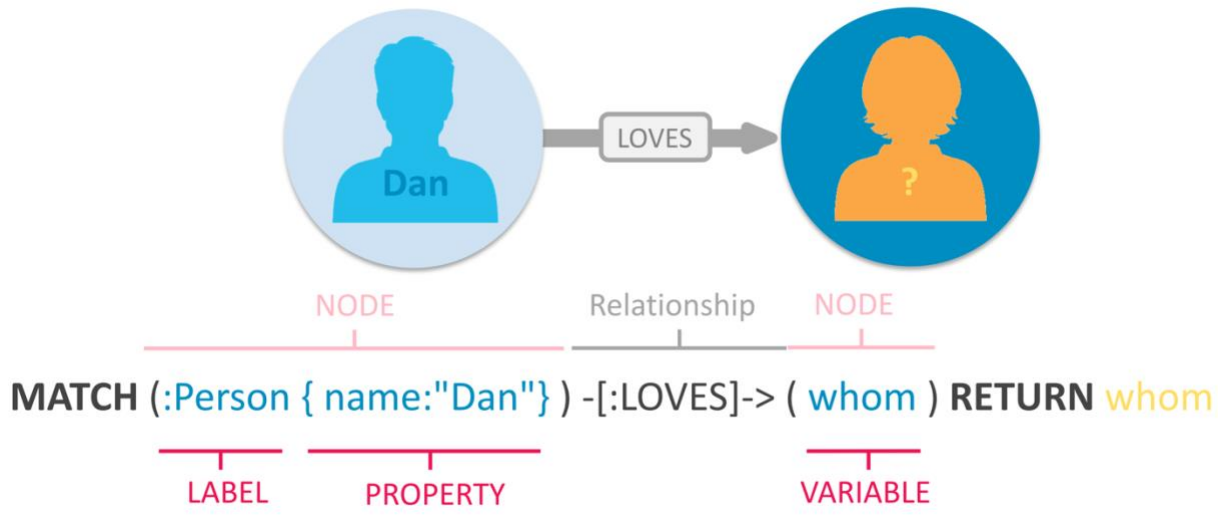


*Figure 1. Neo4j property graph with Cypher query (What is a graph database? - developer guides)*

OrientDB is another example of a graph database platform. Originally authored by Luca Garulli in 2010, it is a Java rewrite of one layer of the ODMS that was written in C++ by Garulli in 1999. OrientDB is an open source, multi-model system. In addition to graphs, it supports documents, key-value pairs, and other object models to store the data. In fact, in OrientDB graphs are written on top of documents but the relationships are managed as in graph databases with direct connections between records, without pointers. The models are integrated into the core without layering, making it a native multi-model database. Like Neo4j, it supports ACID transactions. Unlike Neo4j, it does support SQL queries with extensions to handle relationships without `JOINs`. (Recall that Neo4j provides its own query language called Cypher.) "OrientDB handles every record / document as an object and the linking between objects / documents is not through references, it's direct linking (saving a pointer to the object). This leads to quick retrieval of related data as compared to joins in an RDBMS" (Wikimedia Foundation 2021).

What are the advantages and disadvantages of graph databases?

Perhaps the major advantage of graph databases is the speed and simplicity with which they can be queried. Relational databases reach capacity limits, but the query speed of graphic databases is only dependent on the number of relationships, and it is not significantly impaired by the complexity or quantity of the data. With graph databases, important patterns can be apparent on simple visual inspection of the data diagram and queries allow the user to navigate deep hierarchies to find hidden connections. Another advantage of graph databases is the flexibility of their data structures. Relational databases demand rigid schema which are difficult to modify after the database has been created and put into production. On the other hand, it is relatively simple to add to or modify data structure in a graph database, even after it is up and running. For example, adding a property to a data object or relationship. "Also, with the graph database model, real facts can be stored in a natural way. The structure used is very similar to human thinking, and this is why the links are so clear" (*Graph databases explained*). Disadvantages of graph databases include limited scalability, there is no standardized query language, and they are relative latecomers to the spectrum of database platforms and therefore do not enjoy as much in the way of community support as some of the other database technologies, especially relational models, and Structured Query Language.

"Graph databases should not be considered generally to be an absolute better replacement for conventional databases. Relational structures remain entirely reasonable standard models, guaranteeing high data integrity and stability, and permitting flexible scalability. As so often, the same applies here: It all depends on the intended purpose!" (*Graph databases explained*).

References

Admin. (2021, September 24). *Advantages and disadvantages of NoSQL databases*. Technology Point. Retrieved March 10, 2022, from https://technologypoint.in/advantages-and-disadvantages-of-nosql-databases/

Dryburgh, A. (2017, March 22). *Growth stories: The magical power of a name*. Forbes. Retrieved March 12, 2022, from https://www.forbes.com/sites/alastairdryburgh/2017/03/22/growth-stories-the-magical-power-of-a-name/?sh=5dd8c3cd6db9

Foote, K. D. (2018, June 13). *A brief history of non-relational databases*. DATAVERSITY. Retrieved March 10, 2022, from https://www.dataversity.net/a-brief-history-of-non-relational-databases/#:~:text=In%20the%20mid%2D1990s%2C%20the,often%20referred%20to%20as%20NoSQL.

*Graph databases explained*. IONOS Digitalguide. (n.d.). Retrieved March 13, 2022, from https://www.ionos.com/digitalguide/hosting/technical-matters/graph-database/

Menegasso, A. E. (2018). *NOSQL*. Amazon. Retrieved March 11, 2022, from https://aws.amazon.com/nosql/graph/

OrientDB. (n.d.). *OrientDB*. Home · OrientDB Manual. Retrieved March 13, 2022, from https://orientdb.com/docs/last/

Rangel, D. (2015). *DynamoDB: Everything you need to know about Amazon Web Service's NoSQL database*. Amazon. Retrieved March 10, 2022, from https://aws.amazon.com/dynamodb/?trk=ff72fd6b-7a5f-4511-8a45-4ad45aeffbbe&sc_channel=ps&sc_campaign=acquisition&sc_medium=ACQ-P%7CPS-GO%7CNon-Brand%7CDesktop%7CSU%7CDatabase%7CElasticache%7CUS%7CEN%7CText&s_kwcid=AL%21442213%21549827768373%21e%21%21g%21%21redis&ef_id=CjwKCAiA4KaRBhBdEiwAZi1zzjMgxLOQ4VSqNq_urLCREBO-uViEQrkGIredpdWZ7hazZsvnAQRdYBoCVlcQAvD_BwE%3AG%3As

*SQL VS NOSQL: Differences, databases, and decisions*. Talend. (n.d.). Retrieved March 10, 2022, from https://www.talend.com/resources/sql-vs-nosql/#:~:text=SQL%20is%20the%20programming%20language,generally%20do%20not%20use%20SQ.

*What is a graph database? - developer guides*. Neo4j Graph Data Platform. (n.d.). Retrieved March 12, 2022, from https://neo4j.com/developer/graph-database/

*What is nosql? NoSQL databases explained*. MongoDB. (n.d.). Retrieved March 10, 2022, from https://www.mongodb.com/nosql-explained

Wikimedia Foundation. (2021, December 30). *OrientDB*. Wikipedia. Retrieved March 13, 2022, from https://en.wikipedia.org/wiki/OrientDB

Wikimedia Foundation. (2022, February 17). *Apache Cassandra*. Wikipedia. Retrieved March 10, 2022, from https://en.wikipedia.org/wiki/Apache_Cassandra