Ashleigh Rufus

CS 300-ON

NOSQL Databases

NoSQL, originally created as a combination of the two words “no” and “SQL,” a movement beginning in the early years of the 21st century as the focus came on web-scale databases creation. According to Loshin (2013), NoSQL can have two connotations, “one implying that the data management system is not an SQL-compliant one, while the more accepted implication is that the term means ‘Not only SQL,’ suggesting environments that combine traditional SQL (or SQL-like query languages) with alternative means of querying and access” (p. 83). No-SQL, according to Vaish (2013), is a term “used to address to the class of databases that do not follow relational database management system (RDBMS) principles, specifically that of ACID nature, and are specifically designed to handle the speed and scale of the likes of Google, Facebook, Yahoo, Twitter, and many more” (p. 8). Based on these definitions, NoSQL was created to give a name to the specific databases that do not conform to the traditional RDBMS model nor use SQL.

NoSQL databases were designed to solve the problem of “scalability and availability against that of atomicity or consistency” (Vanish, 2013, p. 9). This can be seen in the processes used in web-scale programs that may be used worldwide—such as Amazon—with multiple users needing similar processes at the same time. Along with solving the problem of availability and scalability, NoSQL databases are able to give more flexibility without as much dependency on the formal database administrations. According to Loshin (2013), “NoSQL databases have more relaxed modeling constraints, which benefit both the application developer and the end-user analysts when their interactive analyses are not throttled by the need to cast each query in terms of a relational table-based environment” (p. 84). This can be seen in the variety of analyses of information (key-value stores or graph databases).

NoSQL databases are continuously being created and according to Vaish (2013, p. 9), most of the mentioned databases are “open source and community-driven.” A few examples of this include Redis, CouchDB, and SimpleDB. As mentioned before, each of the examples is an open-source database available to the public. Redis—a key-value data model--provides data structures with range queries, per the company’s website. Redis also works with an in-memory dataset that has the ability to periodically dump the dataset to disk or append each command to a disk-based log. *Image A* shows a diagram of Redis shared by the services running on the gateway and their

relationship. Next, CouchDB is a document data model. According to the website, CouchDB will store documents with unique names that will allow the client to read and update (add, edit, delete) database documents. The primary unit of data for CouchDB is documents and includes metadata that will be maintained by the database system. *Image B* shows the architecture of CouchDB to demonstrate how CouchDB works. Finally, Amazon's SimpleDB is a column-based data model. According to their website, SimpleDB will organize the data in domains that consist of items which are described by attribute name-value pairs. Though the components are set up similarly to a spreadsheet, the multiple values can be associated with a cell. Also, SimpleDB does not require the presence of specific attributes therefore a single domain can be created with completely different types. *Image C* takes a look at the data organization of SimpleDB at a basic level.

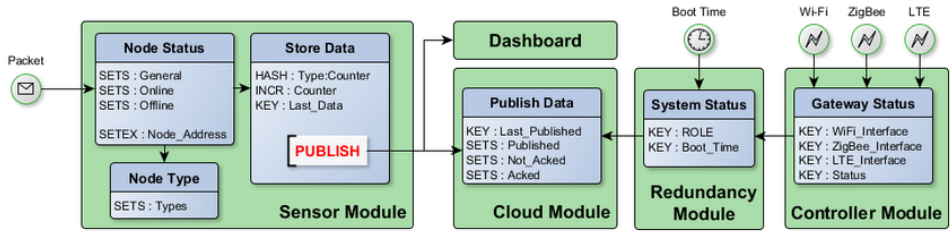

Image A: Redis diagram (Scazzoli, Mola, Silverajan, Magarini, Verticale, 2017)

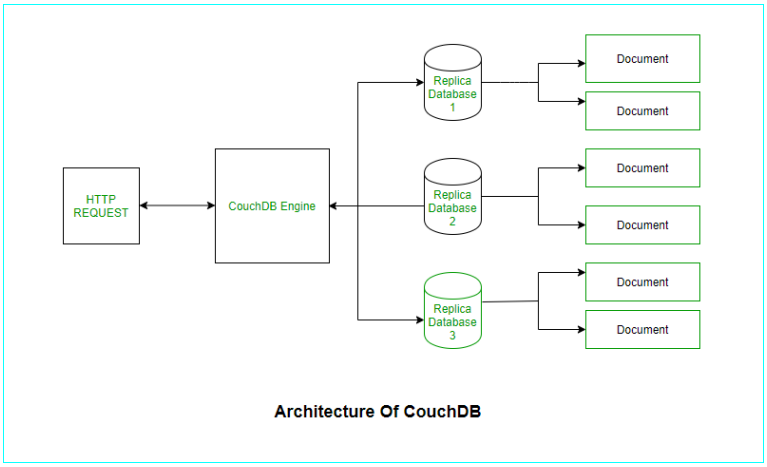

Image B: CouchDB diagram (*Introduction to Apache couchdb, 2020)*

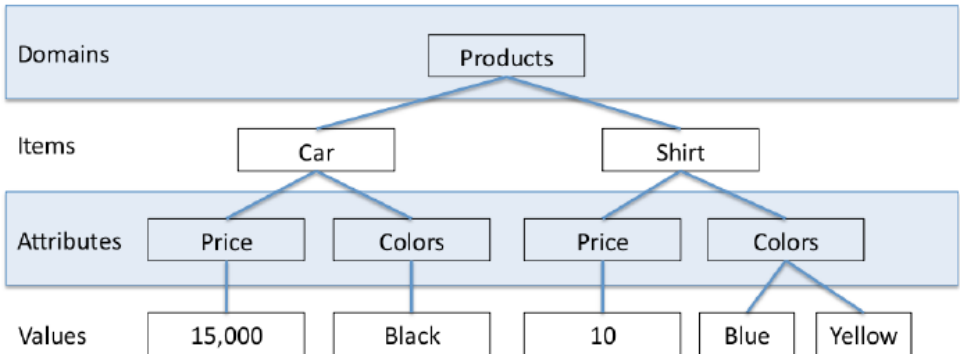

Image C: A glance at SimpleDB data organization (Stein & Zacharias, 2010).

Like many other programs and options, NoSQL databases have their benefits and disadvantages. Some of the advantages include:

- Allowing structure to be defined when it comes to transactional applications and helping to implement JOIN or create views.
- Helping to define rigorous schema, providing speed and scale while fetching partial data, allowing computational performance right up close to data-tier which also increases execution speeds for computational application.
- For web-scale applications, it will provide scale of operations since ACID operations are not implemented.

On the opposite end, NoSQL has a few disadvantages such as:

- Unable to define relationships, transactions being absent, ACID properties in operations are unavailable, some of the databases can be overkill in comparison to other options, and support for JOIN and cross-entity query are absent for transactional applications
- Defining relationships can be difficult and leading to data being inconsistent through computational application.

GRAPH Databases

Graph databases, although categorized in the NoSQL category this form of database is relatively different. Graph databases, as figured by the name, are graph-oriented databases. According to Bruggen & Mohanta (2014), graph databases "aim to provide its users with a better way to manage the complexity of the dense network of the data structure at hand" (p. 32). Graph databases were created to solve three problems complex queries, in-the-clickstream queries on live data, and path finding queries. Complex queries focus on questions for your data that are composed of complex join-style operations. An example of this would be the idea of finding "all the restaurants in a certain London neighborhood that serve Indian food, are open on Sundays, and cater for kids" (Bruggen & Mohanta, 2014, p. 37). From this example, multiple tables of data would have to be joined together to provide an answer to the user. The in-the-clickstream queries on live data solve the problem that complex queries can run into with poor performance on live database management systems and responsiveness of an entire application. Graph databases are then able to answer a wider variety of complex queries that can be updated in basically real time. Path finding queries solve the problem of figuring out how data elements relate to one another, which can be said as "finding the paths between different nodes on your graph" (Bruggen & Mohanta, 2014, p. 39).

Because graph databases are relatively new among the NoSQL category, examples are much further between. Two graph database products are Neo4J and InfiniteGraph, which are open-source and community-driven like the other NoSQL databases. Neo4J, originally created as a graph library, will connect to drivers that support a variety of languages, provide open source, and is extended by integrations that will plug into tools and frameworks—all mentioned on the Neo4J website. *Image D* shows a visualization of a Neo4J graph database. InfiniteGraph contains multiple processes and files that can be distribute across multiple host machines, as mentioned in the documentation guide. The federated database coordinates a particular set of system resources that will manage access to the data in its databases. *Image E* shows an example of an InfiniteGraph system.

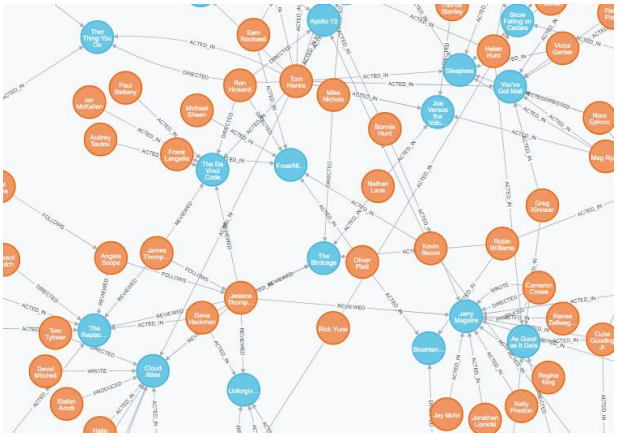

Image D: Neo4J graph database visualization (yWorks, the diagramming company, n.d.).

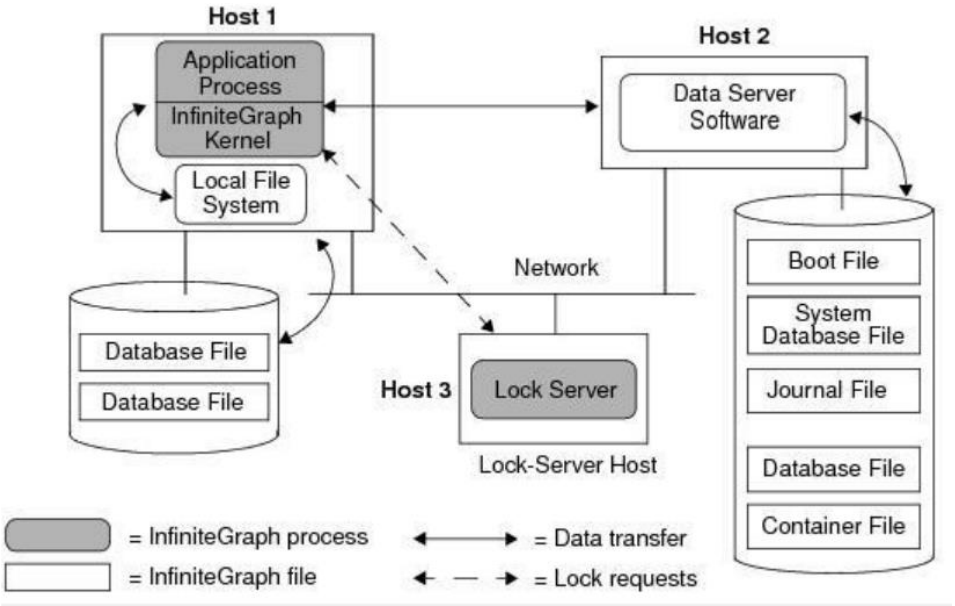

Image E: InfiniteGraph Systems (*InfiniteGraph Basics*, n.d.).

As mentioned with NoSQL databases, there are advantages and disadvantages attached with the graph databases. Some of those advantages may be similar, but different ones are:

- Speeds of query are only dependent on the number of concrete relationships—amount of data is not relevant.
- The results are given in real time.
- Relationships are represented clearly and manageably

- Structures are flexible and agile

On the opposite end, graph databases are not the best due to:

- The database can be difficult to scale because they are designed as one-tier architecture
- Among graph databases, there is no uniform query language

The above-mentioned advantages and disadvantages of the graph databases were found from *Graph Databases Explained* (n.d.).

References

- Bruggen, R. van, & Mohanta, P. (2014). Learning Neo4j. Packt Publishing.
- *Graph databases explained*. IONOS Digitalguide. (n.d.). Retrieved March 13, 2022, from https://www.ionos.com/digitalguide/hosting/technical-matters/graph-database/
- *InfiniteGraph Basics*. Reverb_infinitegraph. (n.d.). Retrieved March 13, 2022, from https://support.objectivity.com/sites/default/files/docs/ig/latest/index.html#page/topics%2Fcommon%2Fadministration%2FadmBasics.html%23ww1022469
- *Introduction to Apache couchdb.* GeeksforGeeks. (2020, July 6). Retrieved March 13, 2022, from https://www.geeksforgeeks.org/introduction-to-apache-couchdb/
- Loshin, D. (2013). *Big Data Analytics: From Strategic Planning to Enterprise Integration with Tools, Techniques, NoSQL, and Graph*. Morgan Kaufmann.
- Scazzoli, Davide & Mola, Andrea & Silverajan, Bilhanan & Magarini, Maurizio & Verticale, Giacomo. (2017). *A Redundant Gateway Prototype for Wireless Avionic Sensor Networks*. 10.1109/PIMRC.2017.8292683.
- Stein, Raffael & Zacharias, Valentin. (2010). RDF on Cloud Number Nine.
- Vaish, G. (2013). Getting Started with NoSQL: Your Guide to the World and Technology of NoSQL. Packt Publishing.
- yWorks, the diagramming company. (n.d.). Visualizing a neo4j graph database. yWorks, the diagramming experts. Retrieved March 13, 2022, from https://www.yworks.com/pages/visualizing-a-neo4j-graph-database