[DM-01-067] NoSQL Databases

Abstract

NoSQL databases are open-source, schema-less, horizontally scalable and high-performance databases. These characteristics make them very different from relational databases, the traditional choice for spatial data. The four types of data stores in NoSQL databases (key-value store, document store, column store, and graph store) contribute to significant flexibility for a range of applications. NoSQL databases are well suited to handle typical challenges of big data, including volume, variety, and velocity. For these reasons, they are increasingly adopted by private industries and used in research. They have gained tremendous popularity in the last decade due to their ability to manage unstructured data (e.g. social media data).

Keywords: database management systems, NoSQL databases, query operations, query languages, relational databases, spatial big data

Author & citation

Li, Z. (2018). NoSQL Databases. The Geographic Information Science & Technology Body of Knowledge (2nd Quarter 2018 Edition), John P. Wilson (Ed). DOI: 10.22224/gistbok/2018.2.10

Explanation

1. Overview of NoSQL Databases
2. NoSQL Databases vs. Relational Databases
3. Spatial Data in NoSQL Databases

1. Overview of NoSQL Databases

Relational databases, originally proposed by E. F. Codd (1970), have been the primary data storage solution for decades. In a relational database, Structured Query Language (SQL) is the standard language for storing, manipulating and retrieving data, thus they are also referred to as SQL databases. As an alternative, NoSQL databases store data in a non-tabular way, different from a relational database. Originally named with reference to Non-SQL/Non-relational databases, nowadays people consider NoSQL as “not only SQL”, emphasizing that a NoSQL database is able to perform as a SQL database but also supports advanced functionalities that a traditional SQL database does not.

NoSQL databases typically have four categories of model stores, namely **Key-Value, Document**, **Column**, and **Graph.**

1.1 Key-Value

Key-value is the most intuitive NoSQL data store. Every data item in the database is stored as a key-value pair, similar to a conventional dictionary. A key is typically a unique ID that points to the data with which it is associated. Possible operations to use with key-value

include getting the value for a key (get), putting or assigning a value for a key (put), and deleting a key (delete). The value can be any object such as string, number, date, array, JSON, etc., so the system is scheme-less. The application is responsible for understanding the type of object and parsing it accordingly. Basho's Riak and Amazon's Dynamo are the most well-known key-value store NoSQL databases.

An example of key-value store is shown in Table 1 below. Note that the address key is associated with a JSON value that contains three attributes: street, city, and state. In a relational database, such a store would be invalid.

Table 1. Example of a key-value store

Key	Value
ID	"a78d1238dfedhz"
Building	"COOR Hall"
Address	{ street: "976 S Forest Mall", city: "Tempe", state: "Arizona" }

1.2 Document store

Document store is similar to key-value store with one difference: it requires that its value stored (called a document) be structured and encoded by metadata. Common encodings include XML, JSON, BSON, and for spatial data, GeoJSON is well accepted by many document store NoSQL databases. MongoDB and Apache CouchDB are examples of this category.

The figure below is showing an earthquake event stored in a MongoDB as a document. The document is in GeoJSON format with earthquake properties and its geometry.

```
{
    "_id": {
        "$oid": "59d5495476bbdae070419021"
    },
    "type": "Feature",
    "properties": {
        "mag": 3.2,
        "place": "151km SSW of Chirikof Island, Alaska",
        "time": 1507149758758,
        "updated": 1507150112184,
        "tz": -600,
        "url": "https://earthquake.usgs.gov/earthquakes/eventpage/ak16989132",
        "type": "earthquake",
        "title": "M 3.2 - 151km SSW of Chirikof Island, Alaska"
    },
    "geometry": {
        "type": "Point",
        "coordinates": [
            -156.904,
            54.6732,
            0
        ]
    }
}
```


Figure 1. Example of an event stored in a MongoDB database.

1.3 Column Store

Column store supports data as columns instead of rows, an arrangement that is usually optimized for queries over large datasets. Querying over rows is memory intensive and requires huge disk access especially when each row contains many columns. With column store, columns are grouped into column families, and each column family can have an unlimited number of columns. In this way it is much easier to query the entire collection of columns for all the rows. Google's BigTable, HBase, and Cassandra are the most popular column store based databases.

In the example below, the different structures of row-based store (left) and column-based store (right) are shown (Figure 2).

Name	GPA	Year	Program
Tom	3.4	2	GEOG
Mary	4.0	1	CS
David	3.7	3	ENV

Name	GPA	Year	Program
Tom	3.4	2	GEOG
Mary	4.0	1	CS
David	3.7	3	ENV

Figure 2. Row-based store versus column-based store.

1.4 Graph Store

Graph store is used to store network data consisting of nodes (entities) and edges (relationships) and properties of nodes and edges. Example of network models includes social networks, life science networks, road networks, and IT/networks, where entities are highly connected through relationships.

In relational databases, entities are stored as tables. Querying relationships between entities is done by performing JOIN to those tables, an operation that is computing- and memory-intensive. In contrast, in a NoSQL database which enables Graph store, relationships are the priority. Each entity contains a list of relationship records linked to other nodes. When performing a query to find those relationships, the database will use this list to find connecting nodes, therefore reducing time from searching and matching. Available graph databases include Neo4J, InfiniteGraph, and OrientDB. On its website, Neo4J has published a helpful comparison of a Graph Store NoSQL database over a relational database.

2. NoSQL Databases vs. Relational Databases

Deciding whether to use a NoSQL database or a relational database depends on the type of applications. There is no “one fits all” solution in databases. Some of the major differences of NoSQL database and relational database are summarized in the table below.

Table 2. Relational and NoSQL Databases

	Relational Database	NoSQL Database
Development Model	Relational databases and their management systems are typically closed source.	NoSQL databases are open-source which boosts the agility of development.
Data Model	Relational databases store data in tables with rows and columns. A predefined schema is usually needed to strictly enforce the table structure and data types.	NoSQL databases typically do not have such pre-defined schema. The schema-less nature better adapts to changes in the data. A key is usually linked with values, columns, documents like JSON, XML, or other encodings.
Flexibility	Relational databases have clear and strict data structures.	NoSQL databases loosen the data structure constraints of relational databases and are considered better to store unstructured and semi-structured data such as text, voice, email, video, etc.
Scability	Scaling a relational database requires investment in hardware, which is considered vertical scaling. Building to distributed systems requires additional costs.	NoSQL databases are designed to horizontally scale across distributed systems at low-cost, which is a main advantage over relational databases.
Performance	Due to the nature of the single host storage, performance relies on the hardware and table structure as well as efficiency of algorithms for indexing and searching.	In distributed systems, the performance usually depends on hardware cluster sizes, network latency, and efficiency of calling application.
APIS	SQL (Structured Query Language) is the standard language for storing, manipulating and retrieving data. The database management system will parse and execute those queries.	NoSQL database supports Object-based APIs which enable applications to easily store and retrieve in-memory data structures. Different data models can be retrieved by their keys they are associated with.
ACID Properties	Relational databases support ACID (Atomicity, Consistency, Isolation and Durability). Atomicity means a change to the database either completely succeeds or completely fails. Consistency means the database should conform to its schema. Isolation requires that concurrent transactions execute separately from one another. Durability is the ability to recover from an unexpected system failure or power outage to the last known state.	While NoSQL does not completely contradict ACID properties of relational database management, it may compromise some ACID properties in return for scalability, flexibility, and performance.

3. Spatial Data in NoSQL Databases

Spatial data operations are commonly supported by many well-known NoSQL databases including MongoDB, Google's BigTable, Apache Casandra, and Apache CouchDB. For instance, MongoDB is used by Foursquare to store check-in and POI data.

In research, NoSQL databases are a popular data storage solution and there is an increasing trend in using NoSQL database in GIS research. The performance of NoSQL databases against relational ones is often tested, and new spatial architectures for NoSQL databases are developed. For example, Xiao and Liu (2011) evaluated NoSQL database in storing global remote sensing images and identified advantages over relational databases. Van der Veen et al. (2012) examined MongoDB's capability for storing sensor data. They compared MongoDB with Cassandra and Postgres (a relational database) and found that MongoDB has the best performance for single database operations. Zhang et al. (2014) designed HBase Spatial, a scalable spatial data storage solution built on HBase NoSQL database. Similarly, Han and Stroulia (2013) created HGrid, which is also a data model for large geospatial datasets in HBase. Padmanabhan et al. (2013) utilized the power of MongoDB to explore Twitter data for mapping diseases. What is more, for networks, Miler et al. (2014) tested Neo4J with PostgreGIS and claimed the shortest path calculation is around 30% faster in the Neo4J NoSQL database.

There are advantages of using NoSQL databases to manage the challenges typical of big data, whether it is spatial or not: volume, variety, and velocity. Volume refers to the huge amount of data being gathered every day every year, such as high spatial – temporal resolution satellite and airborne photos, high precision road network data, POI (Point of Interest) data, data collected from Volunteered Geographic Information (VGI), locations from GPS (Global Positioning System) and mobile devices, radiofrequency identification (RFID) geotags, and georeferenced social media posts (Miller, 2007, 2010; Sui and Goodchild, 2011; Townsend, 2013). Big volume data favors a highly scalable storage solution, which is challenging for relational databases due to their hardware constraints. However, NoSQL databases make it easy to scale horizontally across distributed clusters, and each cluster can host a subset of such data.

Variety means the structure of the data collected are in different structures and from different sources. Traditional relational spatial databases support basic spatial data types such as points, lines, polygons, and rasters. The schema-less nature of NoSQL databases can support a greater variety of spatial data formats from different sensors, sources, and platforms, making it much more flexible. This is important for more complicated types of data that contain spatial elements such as geotagged photos and videos, tweets, messages, etc. (Zhang et al., 2014). Another example would be storing timelines, which are associated with places, for individuals on social media sites. Different individuals have different timelines and it is hard to store such information in a structured way. Those types of data fall into unstructured or semi-structured categories, which relational database does not fully support while NoSQL databases do.

Certain types of data processing and analytics often require a short timeframe, which is referred to as a demand for Velocity. There is a desire for fast processing and querying of spatial data, especially for some predictive algorithms to be applied on real-time streaming data. Many GIS applications such as road navigation, emergency services, real-time remote sensing data processing, and face/vehicle-tracking from cameras require real- or near-real-time responses to be effective. The distributed nature of NoSQL in combination with distributed computing frameworks (e.g. Hadoop) make high performance data storage and

analysis possible. For example, Wachowicz et al (2015) developed a streaming data processing flow for geo-tagged Twitter data using Hadoop and MongoDB to study spatial-temporal activities. Hwang (2013) designed a real-time disease surveillance application using social media data stored on distributed NoSQL database.

Compared to a relational database, a NoSQL database is a better solution to big data challenges from a data storage and management perspective. The high scalability, high flexibility, and high-performance characteristics of NoSQL database can cope with Volume, Variety and Velocity respectively.

References

Codd, E. F. (1970). A Relational Data Model for Large Shared Data Banks. Communications of the ACM. 13(6), 377-387.

Han, D. & Stroulia, E. (2013). HGrid: A Data Model for Large Geospatial Data Sets in HBase. 2013 IEEE Sixth International Conference on Cloud Computing, Santa Clara, CA, USA. pp. 910-917

Hwang, M. H., Wang, S., Cao, G., Padmanabhan, A., & Zhang, Z. (2013, November). Spatiotemporal transformation of social media geostreams: a case study of twitter for flu risk analysis. In Proceedings of the 4th ACM SIGSPATIAL International Workshop on GeoStreaming (pp. 12-21). ACM.

Miler, M., Medak, D., & Odobašić, D. (2014). The shortest path algorithm performance comparison in graph and relational database on a transportation network. Promet - Traffic&Transportation, 26(1), 75-82.

Miller, H. J. (2010). The Data Avalanche is Here. Shouldn't We be Digging? Journal of Regional Science, 50(1), 181-201.

Padmanabhan, A., Wang, S., Cao, G., Hwang, M., Zhao, Y., Zhang, Z., & Gao, Y. (2013, July). FluMapper: an interactive CyberGIS environment for massive location-based social media data analysis. In Proceedings of XSEDE 13, the Conference on Extreme Science and Engineering Discovery Environment: Gateway to Discovery (p. 33). ACM.

Sui, D. Z., & Goodchild, M. (2011). The convergence of GIS and social media: Challenges for GIScience. International Journal of Geographical Information Science, 25(11), 1737–1748.

Townsend, A. M. (2013). Smart Cities Big Data, Civic Hackers, and the Quest for a New Utopia. WW Norton & Company.

Van der Veen, J. S., Van der Waaij, B., & Meijer, R. J. (2012). Sensor Data Storage Performance: SQL or NoSQL, Physical or Virtual," 2012 IEEE Fifth International Conference on Cloud Computing, Honolulu, HI, USA, 2012, pp. 431-438.

Wachowicz, M., Arteaga, M. D., Cha, S., & Bourgeois, Y. (2016). Developing a streaming data processing workflow for querying space–time activities from geotagged

tweets. Computers, Environment and Urban Systems, 59, 256-268.

Xiao, Z., & Liu, Y. (2011, June). Remote sensing image database based on NOSQL database. In 19th International Conference on Geoinformatics, Shanghai, China, 2011, pp. 1-5.

Zhang, X., Song, W., & Liu, L. (2014). An implementation approach to store GIS spatial data on NoSQL database. In Geoinformatics (GeoInformatics), 2014 22nd International Conference on Geoinformatics, Kaohsiung, Taiwan.