Annals of "Dunarea de Jos" University of Galati
Fascicle I. Economics and Applied Informatics
Years XXVI – nº3/2020
ISSN-L 1584-0409 ISSN-Online 2344-441X
www.eia.feaa.ugal.ro

DOI https://doi.org/10.35219/eai15840409134

Comparison between Relational and NoSQL Databases

Gianina Mihai*

ARTICLE INFO

Article history:
Accepted November 2020
Available online December 2020
JEL Classification
M15, M49

Keywords:
SQL, NoSQL, Database, Datastores, RDBMS, DBMS

ABSTRACT

Relational databases have not been the only solution for the applications that use them for years. New applications must deal with large volumes of complex, distributed data of different types, whose volume increases very rapidly. In these applications, relational databases are supplemented with several new generic data models referred to as NoSQL. In this article we shall present the main types and characteristics of NoSQL data models and we shall make, both a detailed and a synthetic, comparison between the relational model and the NoSQL data models.

1. Introduction

Since the advent of relational databases, in 1970 and, in 1974, of the SQL query language, they have been, for more than 30 years, the universal solutions used for data management in companies (with small exceptions, as was the case, for example, with the emergence and decline of object-oriented databases).

Relational databases began to reach their limits after the year 2000, with the massive development of the Internet and the emergence of widely used web applications, which generated very large volumes of varied data with huge speed (text, documents of various types, audio files, video files, etc.), that they needed to manage. In the early and mid-2000s, after the dotcom crisis, e-commerce companies like Amazon or Alibaba developed, social networks (Facebook, Twitter) and the YouTube platform appeared and, last but not least, the Google search engine emerged and evolved.

Relational databases, so efficient in managing structured data, with a slow growth rate and acceptable scaling needs, have been overtaken by the volume, speed and variety of data generated by these companies.

All these giant companies have made a major contribution to the creation and use of new database models and associated technologies, as they needed highly scalable database management systems that could manage data distributed worldwide and which, at the same time, could ensure performance and availability for tens and hundreds of millions of users.

NoSQL database models and NoSQL database management systems are the technologies that these companies created and that are widely used today.

In this article, the main differences between the relational model and the NoSQL data models are presented. Both for the general comparative analysis between the SQL and NoSQL data models, and for the detailed analysis between the relational model and the main types of NoSQL models, we accessed and analysed the domain-specific literature published in international scientific databases. The conclusions of the analyses were summarized and presented in tables, designed to be as suggestive as possible for the purpose of our analysis.

2. General comparison between SQL and NoSQL

The father of the term NoSQL is Carlo Strozzi, who first used it in 1998 to refer to the relational open-source database management system he had begun to develop and to point out the fact that he did not use the SQL language to query the database. (Gupta Edward & Sabharwal, 2015, p. 15). This term, meaning "No SQL database", is different from the generally accepted meaning it has today, namely that of "Not Only SQL", indicating all non-relational database systems, some of them being able to provide SQL-like query language for querying the database. The current meaning of the term NoSQL was given by Eric Evans, in June 2009, at a meeting in San Francisco, organized by Johan Oskarsson for discussions on new open-source distributed databases. In October of the same year, Emil Eifren, the founder and CEO of Neo4J, described, in a message on Twitter, the term NoSQL as "Not Only SQL" (Rocha França, 2015).

*Dunarea de Jos University of Galati, Romania. E-mail address: grizescu@ugal.ro (G. Mihai)

Today, NoSQL is used as an alias for "*Next Generation Database Management Systems mostly addressing some of the points: being non-relational, distributed, open-source and horizontally scalable*" (nosql-database.org, 2020) where:

- ✓ ***Non-relational*** means that the data are represented by principles other than those used by the relational model;
- ✓ ***Distributed*** refers to the fact that, unlike relational databases, NoSQL databases are used efficiently and flexibly in distributed systems with multiple servers and different environments;
- ✓ ***Open-source*** means that, unlike the vast majority of RDBMSs, NoSQL DBMSs are open-source in order to provide solutions for complex applications at lower costs, although many of these systems today also offer commercial variants;
- ✓ ***Horizontally scalable*** refers to the fact that, unlike relational databases that can be scaled mainly vertically, by increasing the capacities of highly specialized and very expensive servers, NoSQL databases can be distributed on multiple commodity servers, thus offering the possibility to add new servers in the system architecture, at any time, as needed;

In addition to the features mentioned in the definition of the term NoSQL, there are several other features that fundamentally differentiate relational databases from NoSQL. NoSQL databases can have all or only some of these features:

- ✓ They are not suitable for ACID transactions (Atomicity, Consistency, Isolation, and Durability), which observe the CAP theorem - "in a distributed database system, we can have at most only two of Consistency, Availability, and Partition tolerance" (Harrison, 2015) and the BASE model (Basically Available, Soft state, and Eventual consistency) whose main purpose is permanent availability (Vatika & Meenu, 2012; He, 2015);
- ✓ They are schema-free or have few schema restrictions, which gives them great flexibility in terms of data storage with different structures (Ruiz, Morales, & Molina, 2015);
- ✓ They are generally unsuitable for complex multi-entity queries because they do not support relationships, foreign keys and JOIN statements (Gupta Edward & Sabharwal, 2015, p. 20);
- ✓ They do not use the SQL language to query the database using object-oriented APIs and, more recently, SQL-like languages (Ularu & Puican, 2012).

In summary, NoSQL databases are intended for large, distributed web applications that allow the management and analysis of huge volumes of data with disparate data types and that require permanent availability at a lower cost. All these characteristics are summarized, compared to the relational databases, in table 1.

Table 1. General comparison between SQL and NoSQL

	Data Model	Transactions	Query Language	Complex query support	Data structure	Scaling	Distributed Database support	Cost	Support/ Expertise	Best suited for
SQL	Relational	ACID	SQL	High	Rigid	Vertical	Low	High	High	OLTP systems with structured data and low growth
NoSQL	Key value, Document, Graph, Column	CAP theorem compliance/ BASE	OO APIs, SQL-like	Low	Flexible	Horizontal	High	Low	Limited	Large scale web applications with un/semi structured data and fast growth

3. Comparison between relational database and types of NoSQL stores

In the domain-specific literature there is an almost general consensus when it comes to classifying NoSQL databases according to the model used for data representation and storage, namely: key-values stores, document stores, graph stores, and wide-column stores. (He, 2015; Harrison, 2015; Hecht & Jablonski, 2011; Davoudian, Chen, & Liu, 2018; Moniruzzaman & Hossain, 2013; Casado & Younas, 2014).

Key-value stores

They store data as alphanumeric identifier pairs - an associated value in standalone hash tables. Each key must be unique, and the values can be simple character strings, an integer or an array, or it can be an object, eliminating the need for fixed data models (Jatana, Puri, Ahuja, Kathuria, & Gosain, 2012). Data search is limited to exact searches and can only be done by key, not by value (Moniruzzaman & Hossain, 2013). Because key-value stores work with data loaded into memory, they are often used to increase the execution speed of complex SQL queries that take a long time to execute (Hecht & Jablonski, 2011). So, they are suitable for applications that use a single key to access data such as online shopping carts, user profile configuration, and web session information (Davoudian, Chen, & Liu, 2018).

Document stores

They are non-relational databases that encapsulate key-value pairs in XML or JSON (JavaScript Object Notation) documents in the form of attributes that have names and one or more values and provide limited support for ACID transactions (Harrison, 2015, p. 53). Like key-value stores, these documents have a flexible layout that allow the user to add or remove attributes at runtime without interrupting the operation of the application. Instead, unlike key-value stores, document stores allow you to search for data using attribute names and their values by value. Another important advantage of this type of data store is that it allows the use of data types. In addition, all the data of an object is stored in a single document and each object can be different from the others, which eliminates the need for object-relational mapping when loading data into the database and makes them suitable for data integration.

Some document stores offer an additional level of document aggregation, called a collection or bucket, that stores a set of documents that contain the same category of information. These collections are similar to tables in relational databases, in which each row is a document with a unique key, but which may have a different structure. In this case, resources, replication, persistence, security are managed much more efficiently at collection level and not at document level (Davoudian, Chen, & Liu, 2018).

XML document stores are primarily used in content management systems (e.g.: healthcare, digital libraries, electronic archives), while JSON stores are mainly used in much faster and interactive operational Web applications (e.g.: blogging platforms and social networks) (Harrison, 2015).

Graph stores

Like relational databases and in contrast to other NoSQL database types, graph stores have a strong theoretical foundation based on Graph Theory. Also, unlike relational databases and other NoSQL stores, graph stores are suitable for efficient data management with multiple relationships eliminating the need for recursive and expensive JOINs through more efficient graph traversals. (Hecht & Jablonski, 2011)

Graph databases store objects that can be represented as key-value pairs or as documents in nodes (ventrices), with relationships between them (edges or arcs), both nodes and relationships being able to have properties (Davoudian, Chen, & Liu, 2018). Nodes may represent entities like people, business, or any other item similar to objects from object-oriented programming. Properties designate any information related to nodes and the edges can relate a node to other node or a node to some property (Jatana, Puri, Ahuja, Kathuria, & Gosain, 2012).

Graph stores can use certain graph-specific algorithms and provide support for querying them. Thus, graph stores are an important alternative when the analysis of the relationships between objects is as important as that of the objects themselves. (Davoudian, Chen, & Liu, 2018)

Column stores

Column stores (wide-column stores, column family stores) store and process data at the column level. The basic concept used is that the column data is stored together on the disk (in the same disk blocks) and not the row data as in the case of the relational model (the data of all columns for the same row). Thus, this type of data store is much more suitable for queries that require data aggregation because all the values that need to be aggregated are stored in the same disk blocks (Harrison, 2015, p. 77). In this model, an arbitrary number of key-value pairs can be stored on lines. For a value, multiple versions are accepted, which are arranged chronologically by default, fact that allows to obtain consistency and an improved performance. Columns can be grouped into families, which improves data organization and the partitioning process. On the other hand, new rows and columns can be added on the fly at runtime but, in this case, column families must be predefined, which leads to less flexibility than that offered by key-value stores and document stores (Hecht & Jablonski, 2011).

Column stores are recommended for analytical and BI applications that require analysis and aggregation of a very large volume of data, often from few attributes (columns) and a huge number of rows (Matthew, 2010) and for applications that manage a very large volume of data stored in multiple clusters due to the efficient partitioning of this model (Hecht & Jablonski, 2011).

In table 2 we synthetically present a summary and comparison of the main characteristics of relational databases and NoSQL stores.

Table 2. Comparison between relational database and NoSQL stores

	Relational	Key-value	Document	Graph	Columns
Base data structure	Tables	Key – value pairs	XML or JSON Documents	Nodes	Tables and column families
Data Schema	Rigid	No schema	Flexible	Flexible	Flexible
Support for joins	Yes	No	Yes	Yes	No
Ad-hoc query	Yes	No	Yes	Yes	Yes
Aggregate across rows	Yes	No	No	No	Yes
Best suited when	complex queries and frequently CRUD operations on structured and slow growth data are needed	high speed and highly scalable caches for applications are needed	data can be easily interpreted as documents with rapidly and constantly growth	relationships between entities are more important than entities themselves	analytical needs on very high volume of data are necessary
Number of DBMSs on the market*	142	65	48	32	11
Representatives examples for DBMSs *	Oracle, MySQL, SQL Server, PostgreSQL, Db2, MariaDB etc.	Redis, Amazon Dynamo, Azure Cosmos, Memcached, Aerospike, Riak KV etc.	MongoDB, Amazon Dynamo, Azure Cosmos, CouchBase, CouchDB etc.	Neo4j, Azure Cosmos, ArangoDB, OrientDB, JanusGraph, Dgraph etc.	Cassandra, Hbase, Azure Cosmos, Datastax Enterprise, Azure Table Storage, Accumulo etc.

** according to the https://db-engines.com platform (multi-model systems, which support that model are also included)*

Most database management systems were organized around a single data model that determines how data could be organized, stored, and manipulated. Because, frequently, a single data model is not enough to cover all the needs of a company, multi-model databases have recently appeared (Mihai, 2020). A multi-model database allows a company to store data in different data models and allows for all these models to be managed with a single management system, which leads to a significant simplification of application development processes. Today, more and more relational database management systems or NoSQL provide support for other data models, becoming multi-model systems (Mihai, 2020).

5. Conclusions

The relational model and NoSQL models are not mutually exclusive, they are complementary, and each has advantages and disadvantages in a particular context. They represent solutions for different scenarios and coexist within companies.

The relational model is more appropriate in the context of centralized, monolithic applications that use structured data, with reduced scaling needs that can be solved by vertical scaling, with low data volume growth rate, but which require transactions and complex joins.

NoSQL models are more suitable for decentralized applications that use unstructured or semi-structured data, which require simple transactions and joins, with high scaling needs (horizontal scaling), with a rapidly increasing data volume, and that need to ensure permanent availability (zero-downtime).

Therefore, NoSQL tends to be a better option for modern web applications that need to store and process more complex, ever-changing data sets in real time and that require a flexible data model. The agility of NoSQL stores is another major advantage that these databases can bring to companies, an advantage that allows them both to enter the market faster and to make much faster updates to their applications in order to accommodate the changes.

Although relational databases still have their well-defined usage scenarios, NoSQL stores offer many features that the former cannot offer without sacrificing a number of parameters such as speed or agility and without greatly increasing costs.

References

1 *Casado, R., & Younas, M. (2014). Emerging trends and technologies in big data processing. Concurrency and Computation Practice and Experience. doi:10.1002/cpe.3398*

2 *Davoudian, A., Chen, L., & Liu, M. (2018). A Survey on NoSQL Stores. ACM Computing Surveys, 41(2). doi:https://doi.org/10.1145/3158661*

3 *Gupta Edward, S., & Sabharwal, N. (2015). Practical MongoDB - Architecting, Developing, and Administering MongoDB. Apress.*

4 *Harrison, G. (2015). Next Generation Databases - NoSQL, NewSQL, and BigData. Apress.*

5 *He, C. (2015). Survey on NoSQL Database Technology. Journal of Applied Science and Engineering Innovation, 2(2), 50-54.*

6 *Hecht, R., & Jablonski, S. (2011). NoSQL Evaluation: A Use Case Oriented Survey. 2011 International Conference on Cloud and Service Computing, (pp. 336-341). Hong Kong. doi:doi: 10.1109/CSC.2011.6138544*

7 *Jatana, N., Puri, S., Ahuja, M., Kathuria, I., & Gosain, D. (2012, August). A Survey and Comparison of Relational and Non-Relational Database. International Journal of Engineering Research & Technology, 1(6).*

8 *Matei, G. (2010). Column-Oriented Databases, an Alternative for Analytical Environment. (A. Bucharest, Ed.) Database Systems Journal, 1(2), 3-16.*

9 *Mihai, G. (2020). Multi-Model Database Systems: The State of Affairs. Annals of "Dunarea de Jos" University of Galati Fascicle I. Economics and Applied Informatics, XXVI (2), 211-2015. doi:https://doi.org/10.35219/eai15840409128*

10 *Moniruzzaman, A. B., & Hossain, S. A. (2013). NoSQL Database: New Era of Databases for Big data Analytics -. International Journal of Database Theory and Application, 6(4), 1-14.*
11 *nosql-database.org. (2020). What is NoSQL? Retrieved June 10, 2020, from nosql-database.org: https://hostingdata.co.uk/nosql-database-org-joins-hostingdata-co-uk/*
12 *Rocha França, W. (2015). MongoDB Data Modeling. Packt.*
13 *Ruiz, D. S., Morales, S. F., & Molina, J. G. (2015). Inferring Versioned Schemas from NoSQL Databases and its Applications. In Lecture Notes in Computer Science (Vol. 9381, pp. 467-480). Springer. doi:https://doi.org/10.1007/978-3-319-25264-3_35*
14 *Ularu, E.-G., & Puican, F. (2012). The new generation of NoSQl Database. Romanian Journal of Informatics and Automation, 4(22), 35-44.*
15 *Vatika, S., & Meenu, D. (2012). SQL and NoSQL Databases. International Journal of Advanced Research in Computer Science and Software Engineering, 2(8), 20-27.*
16 *https://db-engines.com/*