Overview and comparison of modern NoSQL databases

Slobodanka Stanojević Terzić, Fakultet organizacionih nauka, bobanas@gmail.com
Uroš Šošević, Fakultet organizacionih nauka, uros.sosevic@fon.bg.ac.rs
Velimir Štavljanin, Fakultet organizacionih nauka, velimir.stavljanin@fon.bg.ac.rs

Abstract: With every new day, data is getting bigger and more complex. The challenge had stepped out years ago and the industry has started exploring different ideas. At the beginning, the ideas were in the field of relational databases, handing out adjustments, new features, plays and tricks, which all in all did their job. For a while. With the expansion of web it became clear it was insufficient. Big data has stepped on the stage. And so did NoSQL databases. This paper presents different types of databases and shows differences between relational and NoSQL databases.

Keywords: Databases, NoSQL

1. INTRODUCTION

With every new day, data is getting bigger and more complex. The challenge had stepped out years ago and the industry has started exploring different ideas. At the beginning, the ideas were in the field of relational databases, handing out adjustments, new features, plays and tricks, which all in all did their job. For a while. With the expansion of web it became clear it was insufficient. Big data has stepped on the stage. And so did NoSQL databases.

It became necessary to deal with terabytes and petabytes of data, and massive read and write requests had to be handled immaculate. To answer these requirements, companies maintained clusters with thousands of commodity hardware machines. Due to their normalized data model and their full ACID support, joins and locks in relational databases had negative influence on performance in distributed systems. Besides high performance, another fundamental requirement has always been high availability. For that cause, databases had to be easily replicable and had to provide an integrated failover mechanism to deal with node or datacenter failures. Also, they had to balance read requests on multiple slaves to cope with access peaks which exceed the capacity of a single server. Replication techniques offered by relational databases were limited and these databases were typically based on consistency instead of availability, leading to have the requirements answered with additional effort and high expertise [1]. This all led to the trend in which many companies and organizations developed their own storage systems, which are now classified as NoSQL databases.

NoSQL databases have already been introduced, analysed and compared in the past [2] [3] [4] [5]. However, given that features of certain databases are continuously changing, an evaluation of the different features of those is outdated almost at the moment it is published. Hence, providing a durable overview is a challenging task. Instead, rather than being overambitious in delivering it, we shall guide the reader through some of the most important concepts.

Firstly, we will introduce ACID/BASE principles and CAP theorem. Afterwards provide a brief overview of different genres of database styles with corresponding examples for each of them. Finally, different implementations of NoSQL databases will be presented with comparison of their features.

2. ACID, BASE AND CAP THEOREME

ACID is a set of database transactions properties which by [6] exist from 1973. even though the term was defined
Atomicity defines that each transaction must be "all or nothing": if one part of the transaction fails, then the entire transaction fails, and the database state is left unchanged. Consistency ensures that any transaction will bring the database from one valid state to another. Isolation ensures that the concurrent execution of transactions results in a system state that would be obtained if transactions were executed serially. Durability ensures that once a transaction has been committed, it will remain so, even in the event of power loss, crashes, or errors. Even though it is one of the most important concepts of database theory it does become complicated when it comes to distributed transactions across a distributed database.

E. Brewer presented CAP principle in the year 2000. [8]. Two years later S. Gilbert and N. Lynch formalized it and defined as a theorem [9]. What the theorem says is that it is impossible for a distributed computer system to simultaneously provide all three of the following guarantees:

- **Consistency (C)** - equivalent to having a single up-to-date copy of the data,
- **Availability (A)** - the database will always return a value as long as a single server is running,
- **Partition tolerance (P)** - the system will still function even if server communication is temporarily lost—that is, a network partition.

It is important to understand CAP and ACID as two design philosophies which are on the opposite ends of the consistency-availability spectrum. Thorough information about CAP and ACID can be found in [10] and [11].

While relational databases have ACID compatibility (pessimistic), distributed databases have BASE compatibility (optimistic) instead. The acronym comes from the following:

- **Basic Availability (BA)** – the database appears to work most of the time (by using replication and sharding or by using partitioning, availability is guaranteed even if some portions of data become unavailable for short periods of time),
- **Soft state (S)** – stores do not have to be write-consistent; the system will change its state without user intervention due to the eventual consistency,
- **Eventually consistent (E)** – consistency is not guaranteed after every transaction but at some future time.

### 3. DATABASE TYPES AND CHARACTERISTICS

The database technologies are rapidly evolving. With each new version release, modern NoSQL databases incorporate more and more features in order to claim more of the market and remain competitive. In this chapter we will present several models of databases based on Redmond and Wilson and their very detailed research which resulted in [11]. Additionally, some modern open source implementations of those database types will be described.

There are several models of databases present:

- Relational
- Key-Value
- Columnar
- Document
- Graph

The relational model is the most common database model. Relational database management systems (RDBMSs) are implemented as two-dimensional tables with rows and columns. Interacting is achieved through Structured Query Language (SQL) queries. They are good when the structure of data is known in advance. However, relational databases are not so good if the data is highly variable or deeply hierarchical. Most popular open source relational databases include MySQL, MariaDB and PostgreSQL.

The key-value (KV) store is the simplest model of NoSQL databases. KV maps simple keys to more complex values and basically behaves like a huge hashtable. Databases of this type can have very good performance in a number of scenarios but generally are not useful in case of aggregated data needing complex queries. KV databases can be eventually consistent, ordered or implemented on RAM, SSD or rotating disk. Most popular open source KV stores are Dynamo, Redis, Riak, Voldemort, Cassandra and memcached.

Column-oriented databases got their name because data from a given column (in the two-dimensional table sense) is stored together, in contrary to a row-oriented database which keeps information about a row together. Structure
of columnar databases is between relational and key-value. Values are queried by matching keys. On the other side their values are groups of zero or more columns. By having column structure, query performance is increased and compression is more efficient. The most popular open source columnar stores are Apache HBase, MariaDB Columnstore, Apache Kudu and Hypertable.

Document databases allow for any number of fields per object and even allow objects to be nested to any depth as values of other fields. They exhibit a high degree of flexibility, allowing for variable domains. A common representation of these objects is as JavaScript Object Notation (JSON). Document based databases are good if data structure is not known in advance. Additionally, those databases map very well to object oriented-programming models. The two major open source players in the document database market are MongoDB and CouchDB.

Graph databases excel at dealing with highly interconnected data, focusing more on the free interrelation of data than the actual values. A graph database consists of nodes and relationships between nodes. Both nodes and relationships can have properties (key-value pairs) that store data. The real strength of graph databases is traversing through the nodes by following relationships. A graph database is a sort of free-form queries consisting of following edges shared by two nodes - traversing nodes. Good use case for graph based databases is a social network where each node is a user who can have various kinds of relationships. The most popular graph database today is Neo4J. Other than that, mentionable graph databases include ArangoDB and OrientDB.

Overview of most popular modern databases has been presented in Table 1.

4. CONCLUSION

Developing a software system which is able to handle large data requests in optimal time requires adopting big data as a core component of the information management and analytics infrastructure. Big data comes with a number of challenges relating to its complexity. One of them is to understand and use big data when it comes in an unstructured format. Another difficulty is capturing the most important data and delivering it to the right people in real time. And the final challenge is storing the data, analyzing it and understanding it given its size and our computational capacity. There are also numerous other challenges, like privacy, security, access, deployment, etc. [12]

Even though NoSQL concept has brought radical changes in how modern use cases can be dealt with, it is not likely that it will replace traditional approach in database

Table 1. Database characteristics overview
technologies. Successful players like the ones of Facebook, Amazon and Google combine the advantages of both concepts. Facebook’s primary data store is MySQL and they also use a variety of NoSQL stores: HBase for messaging and monitoring, Hadoop for big data analysis, RockesDB (key-value store), Parse (document store), Cassandra and others. [13]

When choosing the right database, aside from choosing a suitable data model, practitioners should also consider the impact of the underlying techniques and decide on the most important criteria for a database selection, having in mind querying possibilities, concurrency control, partitioning, replication and consistency.

REFERENCES