# Primjena i prednosti NoSQL baza podataka

Using the advantages of NoSQL databases

**Mario Novoselec, Denis Pavlović, Milan Pavlović**

## SAŽETAK

*Relacijske su baze danas temelj poslovanja velikog broja modernih organizacija. Težnja organizacija za skalabilnošću sustava i trend razvoja Web 2.0. aplikacija uvjetovali su razvoj NoSQL (Not only SQL) baza podataka. Drugi tip motivacije ogleda se u agilnom pristupu razvoju s naglaskom na smanjenje kompleksnosti i povećanje brzine razvoja. S obzirom na velike troškove u određenim domenama implementacije relacijskih baza podataka, NoSQL baze podataka nastoje smanjiti troškove održavanja skalabilnosti i pružiti jednostavna rješenja za distribuciju i particioniranje modela podataka. Ovaj će rad predstaviti danas najkorištenije tipove NoSQL baza podataka pozicionirajući ih direktno u okvire primjene s naglaskom na direktnu usporedbu u konkretnoj domeni s relacijskim bazama podataka.*

## ABSTRACT

*Relational databases are one of the underlying parts of modern organizations. Need for system scalability and development of Web 2.0. applications are one of the main drivers for NoSQL (Not only SQL) database rise. Another type of motivation is represented by trend of agile development with emphasis on reducing complexity requirements and increasing speed of application development. Considering higher costs in some areas of relational database system implementation, NoSQL databases are trying to directly reduce costs of maintaining scalability and offer solutions for effortless distribution and partitioning of data models. This paper will introduce the most used NoSQL database systems by positioning them directly into the scope of usage with emphasis on direct comparison with relational databases.*

## 1. INTRODUCTION

The NoSQL trend has appeared as a response to massive cost of storing and manipulating data in classical relational database systems. Another benefit of NoSQL movement was flexibility of data modeling and distribution. ''Relational databases provide a variety of features and strict data consistency. But this rich feature set and the ACID properties implemented by RDBMSs might be more than necessary for particular applications and use cases. As an example, Adobe's ConnectNow holds three copies of user session data; these replicas do not neither have to undergo all consistency checks of a relational database management systems nor do they have to be persisted. Hence, it is fully sufficient to hold them in memory.''[2] NoSQL systems share several key characteristics. ''When compared to relational databases, NoSQL systems are more scalable and provide superior performance.''[1]

With this approach, NoSQL databases are trying to resolve some of the most common relational database problems.

Since most of Web 2.0. applications are agile, NoSQL databases tend to be very flexible as opposite to relational databases. For instance, in most NoSQL systems you do not have fixed database schema structure and there is no need for forcing unique data model. These types of data modeling are applicable to applications that generate high amount of inconsistently structured data (e.g. Web blogs, etc…).

Jonathan Ellis from Rackspace defines three problems of relational databases: [3]

1. Data scalability
2. Single server performance
3. Strict schema design

This paper will describe main benefits and key concepts of NoSQL databases with concrete examples in areas of industry and science. Also, this paper will try to present some of the NoSQL database downsides and their affect to organizational and similar environments.

## 2. DATA MODELS

One of the main differentiation between relational and NoSQL databases is the data model. Modern NoSQL databases can be divided into three main categories.

**Document Model**

In this type of databases, data model is represented by documents. Documents have JSON (JavaScript Object Notation) like structure used for storing and traversing through data.

In relational databases one record is scattered through different columns, while in document data model one record is represented with single document (object). In that way, document data model provides an object-oriented approach to data representation.

Documents do not have strict schema structure and can contain different type of fields. Every field can contain different type of data such as date, binary, array or string. ‘’This flexibility can be particularly helpful for modelling unstructured and polymorphic data. It also makes it easier to evolve an application during development, such as adding new fields. Additionally, document databases generally provide the query robustness that developers have come to expect from relational databases. In particular, data can be queried based on any fields in a document."[1]

This paper will focus on one of the most popular document database used today, MongoDB. According to Mongo Inc. there is a wide variety of document database usage, especially in science.

The European Organisation for Nuclear Research, also known as CERN, is using MongoDB for solving problem of storing high amount of differently structured data. "At this scale, the information discovery within a heterogeneous, distributed environment becomes an important ingredient of successful data analysis. The data and associated meta-data are produced in variety of forms and digital formats. However, users want to be able to query different services and combine information from these varied sources. However, this vast and complex collection of data means they don’t necessarily know where to find the right information or have the domain knowledge to extract this data."[4]

The choice of NoSQL document model database was logical in this example mainly because there is no rigid data structure nor data persistency. Because data is not structured equally and there is a need for fast searching through big set of data, document model database can support that kind of requirements in fast and flexible way.

Main alternative to MongoDB as a leading document oriented database, is a CouchDB which is a database oriented towards Web applications. "CouchDB’s design borrows heavily from web architecture and the concepts of resources, methods, and representations. It augments this with powerful ways to query, map, combine, and filter data.’’[5]

**Key-Value Model**

The most basic type of NoSQL databases are key-value stores. Every data instance has its own unique key which is used to access associated value. These kind of data structure is very similar to dictionaries found in some higher programming languages.

This paper will briefly introduce one of the most advanced key-value databases on the market called Redis. ‘’Since most key value stores hold their dataset in memory, they are oftentimes used for caching of more time intensive SQL queries."[2]

This paper will briefly show some example of key-value model and some basic set of operations. In Redis, we can specify our database insert by using SET command:

```
SET user_role 'administrator'
```

Data insertion uses valid key-value syntax and allows ease of access to any value in a data set:

```
GET user_role
```

One of the most important use case of Redis database implementation is Pinterest, multinational social network. One of the main characteristics of social networks is ability to follow other users and their interest. Pinterest tried to implement graph structure as seen in Facebook or Twitter with millions of nodes representing users.

''For example, if Andrea follows Bob, she'll follow all of his boards, and if he creates a new board, she'll automatically follow that board. If Andrea follows Bob's *Recipes board*, she'll see all of his pins from that board in her home feed. Andrea will also be listed as a follower of that board. We term the board followers as *implicit* followers (while the previous type of user-to-user follower is an *explicit* follower).''[7]

That kind of in-depth analysis of users behavior had major requirements towards data caching for real-time user analysis. Relational databases quickly reached their limits because of graph caching specifics. ''Caching the graph data is hard because the cache is useful only if the entire subgraph of a user (vertex) is in cache, however this can quickly result in an attempt to cache the entire graph!'' [7]

Pinterest engineering team found solution in Redis SortedSet data structure. SortedSet is data structure very similar to standard set represented by binary-safe string but with addition of operation to return items in order. Pinterest used Redis for storing graphs which were sharded by ID of a single user. Major disadvantage was single threaded nature of Redis database which was overridden by running multiple instances of Redis on each CPU core.

By using Redis, Pinterest engineering team managed to implement advanced graph structure for user analysis:

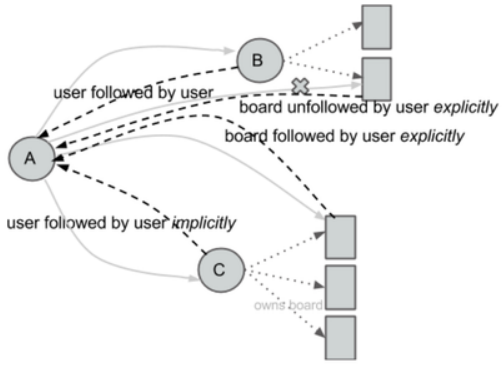


Figure 1: Pinterest graph structure for user analysis [7]

By moving away from relational databases, Pinterest gained some advantages in scalabilty and efficiency of existing infrastructure. ''In the end, when we migrated away from the existing sharded MySQL cluster, we saved about 30% IOps.''

**Column oriented databases**

Column oriented NoSQL databases use multidimensional sorted map as main structure for storing data. In this kind of structure there is random amount of key value pairs that can be stored in each record. ''Each record can vary in the number of columns that are stored, and columns can be nested inside other columns called super columns. Columns can be grouped together for access in column families, or columns can be spread across multiple column families.''[1]

Because of their column oriented structure, column databases are very similar to relational databases. Main advantage of NoSQL column databases are ability to store data without fixed schema and reducing amount of null values to minimum. If there is a data structure with many different types of attributes, relational database would have null value for every instance of data that is not known. In column oriented database data would simply be stored in one row if there is a need for it.

This paper will introduce Casandra as one of the most used NoSQL column oriented databases. In Casandra data structure can be easily represented by group of columns called column family:

CF= user_role

| | | | |
|---|---|---|---|
| rowKey1 | role | permission | dateCreated |
| | admin | All | 2012-1-11 |
| rowKey2 | role | permission | dateCreated |
| | user | Basic | 2013-11-13 |

Every row instance has unique identifying key. There is also column based structure for storing data which can be independent of fixed schema.

One of the most interesting case studies involving Casandra was Bazaarvoice. Bazaarvoice is a service for collecting user generated content and analyzing information gathered through different media. Bazaarvoice's case was oriented towards cloud friendly systems and ease of maintaining clustered database systems. ''Next, we needed a database that allowed for easy capacity expansion (especially write capacity) by simply adding new machines online. Having multiple data center support was also a very big deal, especially where we can write to multiple data centers at the same time.''[9]

Main disadvantage of MySQL as a classical relational database was impossibility to scale according to write capacity growth. Bazaarvoice uses Casandra to store all of customers metadata into single data catalog. That means that every customer becomes single key with different column structure depending on gathered information.

These kind of data structures can be optimized for quick data access because of easy to maintain data structure.

## 3. SCALING NOSQL DATABASES

Most of NoSQL databases are used across multiple systems to distribute large amount of data. NoSQL databases like MongoDB are using sharding to control process of partitioning data on multiple servers. A shard can be easily described as one or more instance of servers in a massive cluster used for distributing any subset of data. The goal is to distribute data evenly across multiple shards by redistributing them. ‘’Relational databases (traditionally) reside on one server, which can be scaled by adding more processors, more memory and external storage. Relational database residing on multiple servers usually uses replications to keep database synchronization.''[11] NoSQL databases are oriented towards cloud and multiple server scalability with focus on ease of maintaining partioned data. Data is transfered on multiple shards in range of key-value pairs. Each shard is only responsible for a specific range of data. Using this strategy, querying certain data range can be done in a fast and efficient way.

In order to maintan high performance of data access over multiple shards, NoSQL databases can compromise data integrity in a way that data can easily be lost or overwritten.

### SCALING ON CLOSED BENCHMARK

In work [12], some performance and scaling comparison between RDBMS MySQL and NoSQL system MongoDB was described. Authors of related work built the benchmarking harness using C programming language and latest stable drivers for each database system. As they describe on p. 12., their benchmarking harness measured the time required to complete a set number of transactions as each transaction on its own is negligible. For calculating the queries per second formulas on fig. 2. were used.

$$Queries\ per\ second = \frac{Total\ number\ of\ queries\ *\ Total\ number\ of\ threads}{Average\ query\ time}$$

$$Queries\ per\ second\ per\ thread = \frac{Total\ number\ of\ queries}{Average\ query\ time}$$

Figure 2. Metrics used in benchmark [12]

As they furthermore describe on p.12-13, database schema used in benchmark was designed and modeled to support a music application which would use different algorithms to suggest songs to users according to their tastes. The normalized schema was made for MySQL database implementation and shown on fig. 2 . Due to fact that MongoDB does not support complicated operations such as JOINs, some compromises were made. Final schema for MongoDB was showed on fig. 3. Details of queries (simple and complex), other statements and configuration of both databases can be found in [12], p. 15.-19. Every SQL statement had it’s equivalent for MongoDB.

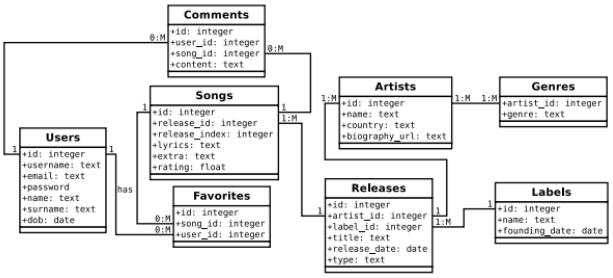


Figure 3. MySQL schema used in benchmark [12]

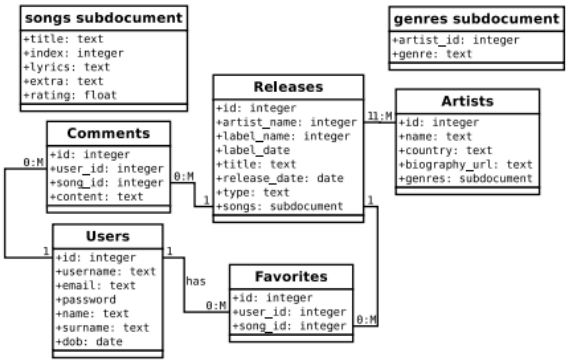


Figure 4. MongoDB schema used in benchmark [12]

Authors in [12] made several conclusions about implemented benchmark. MongoDB could handle more complex queries faster because it worked with simpler schema, but with the cost of data duplication. Despite observed performance gain in complex queries, when queries included nested SELECTs MySQL performed best. In last type of complex query which contained two JOINs and subquery, MongoDB had advantage over MySQL

due to Mongo's use of subdocuments. "This advantage comes at the cost of data duplication which causes an increase in the database size. If such queries are typical in an application then it is important to consider NoSQL databases as alternatives while taking in account the cost in storage and memory size resulting from the larger database size." [12], p. 35-36.

Write operations were also considered in above benchmark. MySQL performed better in data deletion and authors of [12] claim this is logical because MySQL performes better in simple search queries. Searching and deletion are connected because deletion requires finding the record to be deleted first. MongoDB performed better in insertions. Both databases had a linear trend in this test.

Mentioned authors also emphasized the use of different configurations for nodes and threads. "This part of the benchmark required running the benchmarking harness on 1, 2 and 3 nodes with multiple numbers of threads in order to test how the databases performed with multiple connections." [12], p 36. Although databases behaved differently depending on the query complexity, at higher numbers of connections the performance (queries per second) appeared to converge.

Finally, they concluded that two databases behave differently according to the type of queries, so the choice of which database to use lies on the type of application the system will be using. When using MongoDB as database system, it is important to have on mind that this database system results with increased database size. "Despite the indication that the performance penalty on both databases is small depending on the database size it is nonetheless an important factor when considering the type of queries which will be performed by applications" [12], p 36.

## 4. NOSQL POSSIBILITIES ON MOBILE PLATFORMS

Today, mobile applications have some specific and common requirements about data persistence and processing. Currently, mobile applications are one of the most dynamic areas of Information Technology. In similar way, demand for tablets and smartphones has created a huge market for mobile applications developed today. "Also many Business Information Systems/ Business Informatics undergraduate and master programs introduced in their curriculum courses related to mobile devices and applications"[13].

Most of mobile applications today require a persistent data layer, which is also one of the features of web applications. Currently, mobile applications share quite a few features of client - server web application architecture, but there is one striking difference between mobile and web application databases concerns. While on one side web applications have a larger scale and lots of resources on disposal, on the other hand mobile applications have lesser scale and lesser resources on their disposal (processing power) .

There had been huge amount of interest about NoSQL data stores in last couple of years. Primarily NoSQL data stores are used inside big web applications which have needs for storing huge amounts of information about user interactions and similar data. On the other side as mobile platforms and hardware are being developed, storage and performance of devices powered with same platforms and hardware is rapidly catching up desktop platforms. With that demand for solutions like NoSQL databases on mobile devices is increasing and currently used in some mobile applications. Community of developers is trying to embrace NoSQL on mobile platforms in a way of creating special libraries for mobile NoSQL databases. One of those is Android Couchbase library for using and storing data into popular document NoSQL database CouchDB. In that way, mobile application developers can embed CouchDB database into their Android application.

## 5. QUERYING NOSQL DATABASES

In classical SQL databases, schemas are composed of one or more tables where each of the tables is composed internally with fixed structure for table rows. Opposing to tables, MongoDB database is composed of collections in which each of the collections is composed of one or more documents. Also each of those documents that together create collections can have completely different data structure. On SQL side there are few options that can't be found inside MongoDB and vice versa. For example, users could have roles and roles could be referencing users while on MongoDB side there wouldn't be any referencing between collections. Both of databases support querying but in a bit different ways. SQL databases support them in terms of SQL queries. On the other side NoSQL

database or MongoDB in this case supports querying in terms of built in functions. Each of these functions can be used to manipulate data in different ways depending of the context of data. One important notice about MongoDB in this context is that it “permits finding documents with no value declared for an attribute”[13]. Example for this case is an region of one country, if it’s not declared it won’t be found, unless we explicitly specify the $exists attribute. SQL databases do not have equivalent for this query as such query couldn’t be possible because of sharing structure between all the rows in table. Problem on the other side is that as queries become more and more complicated MongoDB database shall extensively use variables inside of one query. Disadvantage of MongoDB query approach is that there are no subqueries, but that problem can be solved via *in* operator. Special operators for nesting queries differentiate NoSQL approach and guarantee most of SQL query possibilities.

```
db.roles.find({user:{$in:['admin','moderator']});
```

## 6. CONCLUSION

NoSQL trend is emerging as a valid relational database alternative for specific use. It is very important to analyse important aspects of different NoSQL data models and include them in system requirements. As mentioned, NoSQL databases behave different according to size of data set and amount of operations to execute. Flexible schema approach and functional query structure manage to increase performance and give effortless ways for data partitioning.

Relational SQL databases are focusing on rigid structure with defined data types for storing data. In that way, they are not suitable for use in environments that are generating massive amounts of differently structured data. With growth of Web 2.0. application usage there are a lot of requirements for NoSQL databases.

## REFERENCES


[1] MongoDB Inc., "Top 5 Considerations When Evaluating NoSQL Databases", June 2013.
[2] C. Strauch, "NoSQL Databases", Stuttgart Media University, 2004, pp. 1–45.
[3] J. McKEnna, "NoSQL Ecosystem" [Online]. Available: http://www.rackspace.com/blog/2009/11/09/nosql-ecosystem/. [Accessed: 16-May-2014].
[4] MongoDB Inc, "CERN CMS" [Online]. Available: http://www.mongodb.com/customers/cern-cms. [Accessed: 16-May-2014].
[5] The Apache Software Foundation, "Why CouchDB?" [Online]. Available: http://docs.couchdb.org/en/latest/intro/why.html. [Accessed: 14-May-2014].
[6] K.Seguin, "The Little Redis Book", January 2012.
[7] A. Khune, "Building a follower model from scratch" [Online]. Available: http://engineering.pinterest.com/post/55272557617/building-a-follower-model-from-scratch [Accessed: 14-May-2014].
[8] "NoSQL - Not only SQL" [Online]. Available: http://scriptandscroll.com/2011/08/21/nosql-not-only-sql-introduction-to-apache-cassandra/ [Accessed: 16-May-2014].
[9] Planet Casandra Inc., "Bazaarvoice Chooses Casandra over MySQL, HBasa and MongoDB to Power Content Analytics Platform" [Online]. Available: http://planetcassandra.org/blog/post/bazzarvoice-chooses-cassandra-over-mysql-hbase-and-mongodb-to-power-content-analytics-platform/ [Accessed: 16-May-2014].
[10] K.Chodorov, "Scaling MongoDB", O'Reilly Media New York, 2011.
[11] J. Pokorny, "NoSQL databases: a step to database scalability in web environment", Charles University, Praha, 2013.
[12] C. Hadjigeorgiou, "RDBMS vs NoSQL: Performance and Scaling Comparison", The University of Edinburgh, August 2013.
[13] M. Fotache, D. Cogean, "NoSQL and SQL Databases for Mobile Applications. Case Study: MongoDB versus PostgreSQL", University of Iasi, Romania, February 2013.


## INFORMATION ON AUTHORS:


**Mario Novoselec**
mnovosel2@foi.hr
Faculty of Organization and Informatics


Mario Novoselec is a third year full time undergraduate student at Faculty of Organization and Informatics, University of Zagreb. He is one of the initial members of FOI MT Lab. His main interests are focused towards developing Web applications with emphasis on user experience and modern design. He has also been involved in various non-commercial projects.


**Denis Pavlović**
dpavlovi@foi.hr
Faculty of Organization and Informatics


Denis Pavlović is a third year full time undergraduate student at Faculty of Organization and Informatics, University of Zagreb. He is also one of the initial members of FOI MT Lab. He is interested in modern web application design and development. He is involved in various non-commecial projects.

**Milan Pavlović**
mpavlovi2@foi.hr
Faculty of Organization and Informatics

Milan Pavlović is a third year full time undergraduate student at Faculty of Organization and Informatics, University of Zagreb. He is also one of initial member of FOI MT Lab. His main interests are Information Systems engineering and development. He is also interested in Java desktop, mobile and web application development and technology.