COMPARATIVE ANALYSIS OF RELATIONAL AND NON-RELATIONAL DATABASES

Pero Ranilović, Dražen Marinković, Nedeljko Šikanjić

Pan-European University "Apeiron". Banja Luka, RS, BiH

{pero.ranilovic, drazen.m.marinkovic, nedeljko.s.sikanjic}@apeiron-edu.eu

Contribution to the State of the Art

https://doi.org/10.7251/JIT2301020R UDC: 681.322.06:004.652.4

Abstract: This paper presents the results of research into the use of relational and non-relational databases, as well as their comparative analysis. A theoretical overview of the comparative analysis by different segments of relational and non-relational databases is presented. Comparative analysis through the practical application of databases is shown through the use of applications for measuring system performance.

Keywords: relational databases; non-relational databases; comparative analysis; MSSQL; MongoDB

INTRODUCTION

As the IT market develops rapidly, there is a need to check available database solutions, but also to adapt them to different conditions. The current situation in the IT world, as well as the great popularity of NoSQL databases, encourages more detailed research and analysis of why non-relational databases are being used more and more today, in addition to standard good relational databases. Relational and non-relational database models, as well as their comparative analysis, are presented in this paper.

COMPARATIVE ANALYSIS - THEORETICAL REVIEW

Many traditional relational databases have been used in a very large number of applications so far. New technologies have been developed with the aim of dealing with increasing amounts of complex data. Choosing the most suitable database can sometimes be tricky, and such comparisons make it easier for the user to choose the appropriate database that can meet all the needs of the application. The user uses SQL or another structured query language to manipulate and define the data, and then applies a predefined scheme to analyze the data. SQL is great for complex queries, however even the smallest change in the database can affect the functionality of the entire system. With NoSQL databases, dynamic schemes are used to manage data that does not have to be stored in tables, but can be displayed in documents, graphs, columns, and so on. For decades, the predominant data model used for application development was the relational data model used by relational databases such as Oracle, SQL Server, MySQL, PostgreSQL, and others. It was not until the mid-2000s that other data models began to be significantly adopted and used. The term NoSQL is used to distinguish and categorize these new classes of databases and data models. When choosing a modern database, one of the biggest decisions is whether to choose a relational (SQL) or non-relational (NoSQL) data structure. Although both relational and non-relational are viable options, there are key differences between these two types of databases that users must keep in mind when making a decision.

A. Database architecture and scheme

At the most basic architectural level, the biggest difference between the two technologies is that SQL databases are relational, while NoSQL databases are non-relational. SQL databases use a structured

query language and have a predefined scheme for handling data. SQL is one of the most versatile and widely used query languages, which makes it a common choice for many use cases. It is perfect for complex queries. However, SQL can be too restrictive. Predefined schemes must be used to determine the structure of the data in order to work with it. All data must have the same structure. This process requires significant preparation in advance. If there were ever a desire to change the data structure, it would be difficult and would disrupt the entire system. On the other hand, NoSQL databases have dynamic schemes, and data is stored in many ways. Column-oriented storage, documents, graphs, and similar can be used to store data. This flexibility means that documents can be created without first defining their structure, that each document can have a unique structure, that the syntax can differ from database to database, and that fields can be added at an indefinite time interval.

B. Database scaling

SQL databases are vertically scalable in most situations. It is possible to increase the load on a single server by adding more CPU, RAM or SSD capacity. NoSQL databases are horizontally scalable. More data can be managed which adds more servers to the NoSQL database. Horizontal scaling has greater overall capacity than vertical scaling, making NoSQL databases the preferred choice for large, frequently changing data sets.

C. Data structure

SQL databases are table-based, while NoSQL databases are document stores. SQL databases are better for multi-row transactions, while NoSQL is better for unstructured data such as documents or JSON. SQL databases are also often used for legacy systems that are built around a relational structure.

D. Optimal workload

Relational databases are designed for transactional and highly consistent online transaction processing (OLTP) applications and are good for online analytical processing (OLAP). On the other hand, non-relational databases are designed for many data access patterns involving low-latency applications as well as semi-structured data analytics.

Choosing or recommending a database is a key responsibility for most database professionals, and "SQL vs NoSQL" is a useful rubric for informed decision-making. When considering any database, it is also important to consider critical data needs and acceptable trade-offs to meet performance and uptime goals. Choosing the right database is not easy. An optimal but unknown database can negatively impact the entire project, while a suboptimal but known tool can be sufficient to get the job done.

Once a user has decided whether to use a SQL or NoSQL database, he must move his data into it. Data migration is a complex process that can present serious challenges. If there is a problem with that operation, Xplenty's Extract, Transform, Load (ETL) helps with automated functionality and a code-free visual interface to facilitate data transfer. Extensive support is available for all SQL databases from their vendors. There are also many independent consultants who can help with SQL database for very large applications, whereas some NoSQL databases still need to rely on community support. Only some external experts are available for NoSQL functionality.

SQL is usually a good choice and is fairly universal for most projects. However, for more specialized work, a NoSQL database can provide a much more efficient result. When you need to find a fast and scalable database, if you don't mind sacrificing some robustness, MongoDB might be just what you need. The use of both SQL and NoSQL databases has its place in modern software development. Each of them has its advantages and disadvantages. NoSQL databases can include SQL elements, while SQL databases can offer some of the advantages of NoSQL through new functions.

Comparative analysis – practical work

A. Installation

For the purposes of using databases, it is necessary to install servers and a tool for working with databases. MSSQL Express Server and MSSQL Server Management Studio 2018 were used in the project work to create relational databases. To create non-relational databases, we used MongoDB server v5.0 and MongoDB Compass. The procedure for installing and setting up both databases is very simple, but installing the MongoDB database requires addition-

al steps, such as setting folder permissions, adding system variables, and the like.

B. Syntax

The SQL language is used by relational databases. SQL is used to define data and to manipulate data. It represents a reliable and safe language for working with complex databases, as well as database queries. It has certain rules that must be followed when using it, and for that reason it is limited. An example of an SQL query is as follows:

```
INSERT INTO Izdavac (Naziv)
VALUES ('Rezim Beograd');
```

Non-relational databases, unlike relational databases, do not use the SQL language to define data, but store all data in JSON format. JSON (JavaScript Object Notation) is one of the standards for storing text designed for readable data exchange. In addition to the JSON format, BSON (Binary JSON) is also very often used, which enables the recording of additional data such as binary data and the like. An example of a NoSQL query is shown below:

```
db.izdavac.insert({"Naziv":"Rezim Beograd"});
```

Table I. *Display of Different Queries Over Databases*

MSSQL	MongoDB
SELECT * FROM Film	db.Film.find();
CREATE TABLE [dbo].[Zanr]([ZanrId] [int] IDENTITY(1,1) NOT NULL, [Naziv] [nvarchar](20) NULL, CONSTRAINT [PK_ZANR] PRIMARY KEY	db.createCollection("Film");
INSERT INTO Osoba VALUES ('Marko', 'Markovic', '1985-12-17', 'Vidovdanska bb')	db.Osoba.insert({"Ime":"Marko", "Prezime":"Markovic", "DatumRodjenja":'1985-12-17', "Adresa":"Vidovdanska bb"};
UPDATE Osoba SET Adresa = 'Banjalucka bb'	db.Osoba.updateMany({}, {$set:{"Adresa":"Banjalucka bb"}});
DELETE FROM Osoba WHERE Ime= 'Marko'	db.Osoba.deleteOne({"Ime":"Marko"});
DELETE FROM Osoba	db.Osoba.deleteMany ({});
SELECT F.Naziv, F.VrijemeTrajanja, Z.Naziv FROM Film as F JOIN FilmZanr as FZ ON FZ.FilmId = F.FilmId JOIN Zanr AS Z ON FZ.ZanrId = Z.ZanrId WHERE F.Naziv = 'Novi 2 film'	db.Film.find({Naziv:"Novi 2 film"});
SELECT COUNT(*) as Kolicina FROM Film	db.Film.count();

By using SQL and NoSQL databases, differences in the syntax of these databases can be observed. This paper will use the MSSQL database, which is a representative of SQL or relational databases, and the MongoDB database, which is a representative of NoSQL or non-relational databases. Given that the access and content of these databases is different and the syntax is significantly different. In order for one to better see the differences between the syntaxes, the table with basic queries in MSSQL and MongoDB is shown below.

C. Structure

Data within the database is logically organized according to the database model. The database model itself determines what the logical structure of the database may look like. The relational model is based on relations, and data is displayed in tables. Relational databases are based on tables. A table consists of columns and rows, and each column is defined as an attribute of the table. Rows within a table are defined as an "n-tuple" of the table. As mentioned earlier, MSSQL is limited and it is necessary to define precisely for each attribute which data type will be placed.

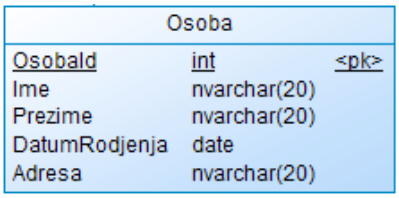


Figure 1. *View of the table Person with exactly defined attribute types*

Non-relational databases are dynamic, and the user determines which attributes and data to place within the collection. There is a great difference be-

tween relational databases that use tables and non-relational databases that use documents. Within the collection, data can be placed according to the user's wishes, and these data can also differ in each subsequent entry according to the number of parameters that will be passed. Using non-relational database models, there is a certain freedom and it is much simpler to add new types and new data. A view of one data set within the Person collection is shown in the image below. It can be concluded that during each data entry, the name of the attribute that is placed is also entered, and for this very reason, the possibility of entering a smaller number of attributes opens up.

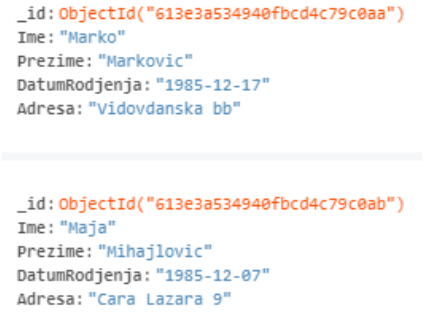


Figure 2. *View of the table Person with exactly defined attribute types*

TIME COMPARISON OF EXECUTION OF QUERIES USING MSSQL AND MONGODB DATABASES

In this part of the paper, the results obtained by executing different queries using different tools will be presented. After the obtained results, comparisons of the time needed to execute queries on the databases will be made. Client Statistics within MSSQL Management Studio was used to measure the time required to execute queries against the MSSQL database.

When executing a query against MongoDB, using the console, a function was used to display the necessary information after the query was executed:

db.Film.find().explain("executionStats");

After the function is successfully executed, an object containing the data "executionTimeMillisEstimate" is obtained, which returns the time required to execute the query. Using MongoDB Compass, the time required to execute a query can be found using the "Explain Plan" tab where you can get the exact performance and data about the executed query. All queries and performance tests were run on a spec laptop:

- Intel Core i7-7500U
- 16GB RAM DDR4
- Maxtor Z1 SSD 480GB, 6Gb/s
- Windows 10 Pro

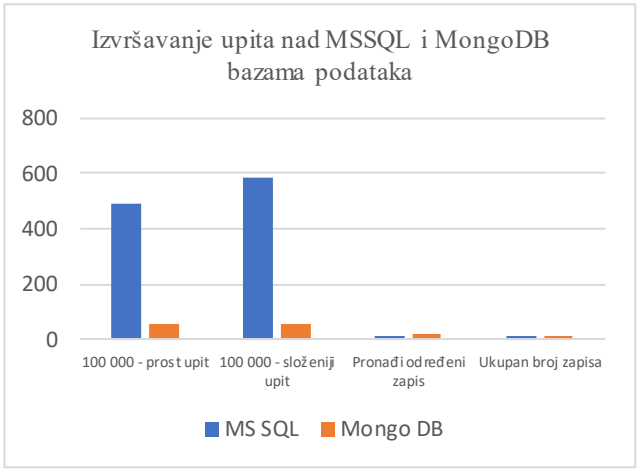
Table II. *Display of Executed Queries*

MSSQL	MongoDB
SELECT * FROM Film	db.Film.find();
SELECT F.Naziv, F.VrijemeTrajanja, Z.Naziv FROM Film as F JOIN FilmZanr as FZ ON FZ.FilmId = F.FilmId JOIN Zanr AS Z ON FZ.ZanrId = Z.ZanrId	db.Film.find();
SELECT F.Naziv, F.VrijemeTrajanja, Z.Naziv FROM Film as F JOIN FilmZanr as FZ ON FZ.FilmId = F.FilmId JOIN Zanr AS Z ON FZ.ZanrId = Z.ZanrId WHERE F.Naziv = 'Novi 2 film'	db.Film.find({Naziv:"Novi 2 film"});
SELECT COUNT(*) as Kolicina FROM Film	db.Film.count();

Table 3 shows the results of query execution times of different complexity on the MSSQL database and the MongoDB database. The data within both databases were identical. The number of records created in the movie table was 100,000. It should be emphasized that very often the results of executing a query on both databases are much slower the first time, when the query is executed again the execution is much faster. The results shown in the table are taken as the mean value of five consecutive measurements over the same number of records.

Table III. *Display of the Obtained Performance Times*

Execution time - MSSQL	Execution time - MongoDB	Successfully retrieved records
489 ms	55 ms	100 000
587 ms	54 ms	100 000
13 ms	19 ms	1
11 ms	16 ms	1



Graph 1. *Query execution times over MSSQL and MongoDB*

TIME COMPARISON OF EXECUTION OF QUERIES USING MSSQL AND MONGODB INSIDE THE APPLICATION

In order to be able to measure and then compare the execution time of basic CRUD operations on MSSQL databases and MongoDB databases, a project was created in which connections were made to one and the other database. The application was developed in the programming language C#. Database access is enabled using the NuGet packages MongoDB. Driver and MongoDB.BSON. Given that two databases were used, in order to enable fast and simple database changes, the connection strings containing the paths to the databases were stored inside the web.config file. Within the code itself, there are no major differences between the ways of using both databases. The steps in creating the project were creating a connection, creating methods for adding data within databases, reading, modifying and deleting (so-called CRUD operations). Created multiple methods that generated datasets depending on the passed noElements parameter. A test data set was created for testing purposes. CRUD operations on the data were performed on samples of 1, 10, 100, 1000, 10,000, 100,000 records, in order to observe the time dependence with increasing number of records. The test data set was created with the help of the createList method, which receives the variable noElements as a parameter. The method code is shown in the image below.

```
7 references
public static List<BsonDocument> kreirajListu(int brElemenata)
{
    List<BsonDocument> listaElemenata = new List<BsonDocument>();
    for (var i = 0; i < brElemenata; i++)
    {
        var naziv = "Novi " + (i + 1) + " film";
        BsonDocument document = new BsonDocument {
            { "Naziv", naziv },
            { "Orginalni naziv", naziv },
            { "Distributer", new BsonDocument { { "Naziv", "Cinema City" }}},
            { "Jezik", new BsonDocument { { "Naziv", "Srpski" }}},
            { "VrijemeTrajanja", (i+1) },
            { "DatumPremijere", DateTime.Now },
            {"Sinhronizovan", true }
        };
        listaElemenata.Add(document);
    }

    return listaElemenata;
}
```

Figure 3. *CreateList method code*

A. Data addition operation

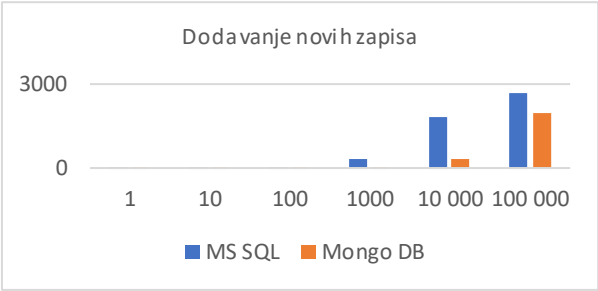
The operations of adding records to the MSSQL database do not require much effort to implement, because both MSSQL and Visual Studio are created by Microsoft, which leads to more efficient collaboration and interaction between the two platforms. The process of adding records to a MongoDB database requires almost the same amount of effort as implementing it with an MSSQL database. The only difference is that more pre-installation and preparation is required at the very beginning. Figure 3 shows the code for adding records to the MongoDB database. To add records to the database, the InsertManyAsync method is used, which is taken from the MongoDB.Driver package.

```
public static async Task<string> InsertFilm(List<BsonDocument> pplList)
{
    Stopwatch timer = new Stopwatch();
    var collection = OstvariKonekcijuNoSQL();
    timer.Start();
    var result = await collection.InsertManyAsync(pplList).ContinueWith(x => x.ToJson());
    timer.Stop();
    return timer.Elapsed.TotalMilliseconds.ToString();
}
```

Figure 4. *The InsertFilm method used to add records to a MongoDB database*

Table IV. *Display of the Obtained Times Required for Performing the Addition Operation*

Execution time - MSSQL	Execution time - MongoDB	Successfully created records
33.19 ms	7.65 ms	1
18.65 ms	19.95 ms	10
56.33 ms	11.82 ms	100
353.98 ms	51.75 ms	1000
1845.08 ms	338.91 ms	10000
2695.66 ms	1955.36 ms	100000



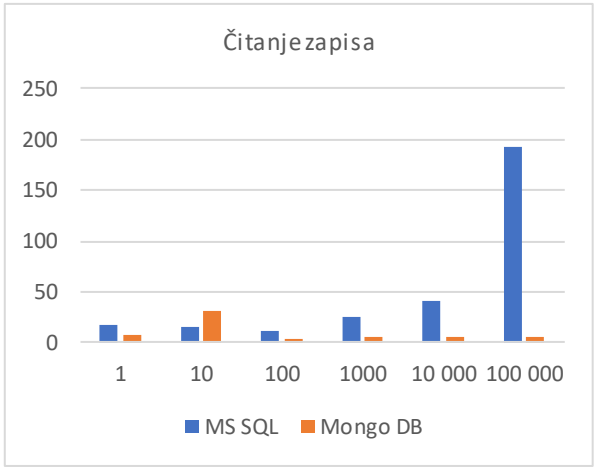
Graph 2. *Adding new records in MSSQL and MongoDB*

B. Data reading operation

Database reading operation depends on many factors. The very structure of the data and the method of saving it in the database is an important factor. Due to its unstructured nature, the MongoDB way of reading data can be quite complex when finding and searching for information in the database. Read operations mostly depend on the complexity of the data structure and how it is stored. The data used when executing the query and measuring the execution time does not have complex complexity, which results that for reading certain simple records the performance of the MongoDB database is better compared to MSSQL. By executing queries on the database and reading the data, results were obtained that show that temporal MongoDB queries are executed faster, especially in the case of a large number of simple records. The results are based on the average time it takes to perform a read operation over a different number of records.

Table V. *Display of the Obtained Times Required for Performing the Reading Operation*

Execution time - MSSQL	Execution time - MongoDB	Successfully created records
17.15 ms	8.06 ms	1
14.69 ms	31.34 ms	10
12.40 ms	3.45 ms	100
24.96 ms	5.28 ms	1000
40.36 ms	6.06 ms	10000
192.76 ms	5.90 ms	100000



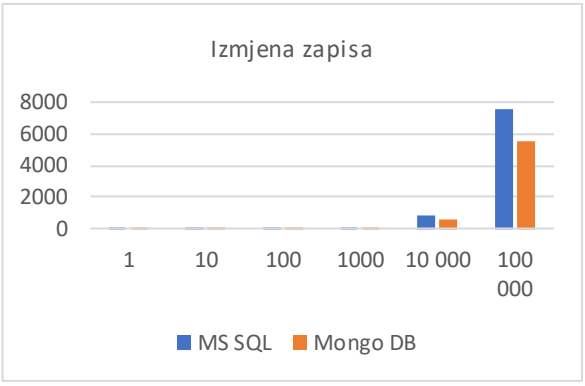
Graph 3. *Reading records in MSSQL and MongoDB*

C. Data change operation

Changes to database data can be made using different criteria. When executing the query in the application, the update of all records and the modification of two attributes within the record were used. Execution performance depends on the criteria used when finding a particular record. MongoDB achieves better results, while in situations with complex criteria, MSSQL wins. For example, changing all records that contain data of the string type "Banja Luka" in a populated place. Record change operations and their time comparison are shown in Table 6.

Table VI. *Display of the Obtained Times Required for Performing the Change Operation*

Execution time - MSSQL	Execution time - MongoDB	Successfully created records
17.27 ms	25.05 ms	1
15.61 ms	10.48 ms	10
15.52 ms	6.52 ms	100
118.85 ms	64.41 ms	1000
850.57 ms	618.78 ms	10000
7487.32 ms	5511.06 ms	100000



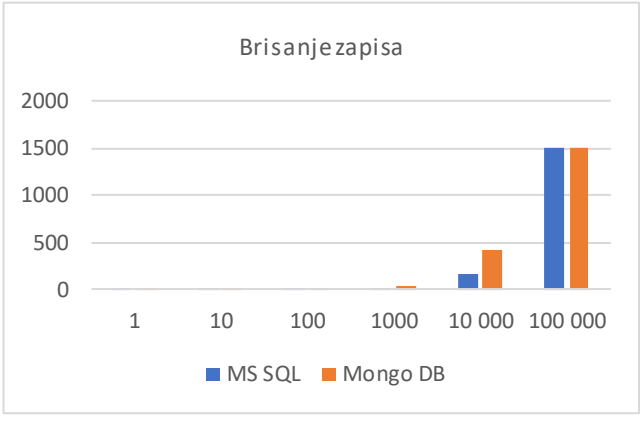
Graph 4. *Change of records in MSSQL and MongoDB*

D. Data deletion operation

The performance of deletion operations on both databases gives similar execution times. When deleting 10,000 records, a better average time is obtained for the MSSQL server, while in other cases the execution times are approx.

Table VII. *Display of the Obtained Times Required for Performing the Deletion Operation*

Execution time - MSSQL	Execution time - MongoDB	Successfully created records
19.35 ms	9.42 ms	1
16.62 ms	8.58 ms	10
12.92 ms	5.23 ms	100
22.09 ms	46.38 ms	1000
157.64 ms	429.36 ms	10000
1508.67 ms	1495.60 ms	100000

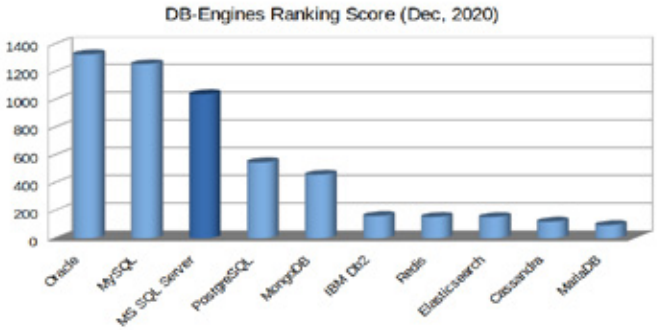


Graph 5. *Deletion of records in MSSQL and MongoDB*

E. Description of the database management systems used

The databases that were used during the development of the application for comparative performance analysis are among the 10 most famous databases in the world. According to data from database ranking site DB-Engines, the MSSQL database is in third place in terms of usage, while the MongoDB database is in fifth place. Microsoft SQL Database Management System is Microsoft's database storage tool. The first version of this tool appeared in 1989 in cooperation between Microsoft and Sybase. After breaking up with Sybase, Microsoft made significant progress in the development of its DBMS. In 1998, there were the first possibilities of using relational databases with personal computers. Its base language is Transact-SQL, which is an implementation of the ANSI/ISO SQL standard. Microsoft SQL Server Express v15.0 was used during the creation of the project, which is otherwise a free version for smaller applications and learning.

MongoDB is a representative of non-relational databases, it is a database management system that uses a document-oriented database model. It is one of the numerous representatives of NoSQL databases. It was created by Dwight Merriman and Elliott Horwitz. It uses a document-based data model because it claims to be a better and more natural way to display data. It stores data as JSON or BSON. MongoDB v5.0 was used in the project development.



Graph 6. *Top 10 databases used in 2021 according to DB-Engines results*

I.

CONCLUSION

Research in the field of databases carried out during the preparation of the paper, and the analysis of all obtained results shows that MongoDB generates better performance on a larger umber of records and on larger amounts of data. Up to 1000 records, the execution times have approximate values, while above 1000 records, much better times are observed when using the MondoDB database. By reviewing individual cases where the results are better with MSSQL databases, it can be concluded that by directly applying MSSQL Management Studio over the database, better execution times are obtained. Therefore, it can be said that MSSQL databases are suitable for small and medium applications. They are suitable when performance is not a priority. Relational databases are widely used in most applications and they perform well when manipulating a limited amount of data. One should be careful when choosing a database. Major factors such as data volume, flexibility, scheme, budget, server type, amount of transactions to be executed, and frequency should be considered. Of course, these are not the only criteria for choosing a database, as it also depends on the company, as well as the purpose for which the application is being developed.

References

[1] Neeraj Sharma, "*Database Fundamentals*", IBM Corporation, 2010.

[2] S. Marić, D. Brđanin, „*Relacione baze podataka*", Banja Luka, 2012.

[3] Aya Al-Sakran, Musbah Jumah Aqel, "*A Comparative study of NoSQL Databases"*, 2017.

[4] G. Pavlović-Lažetić, "*Osnove relacionih baza podataka*", Matematički fakultet, 1999.

[5] S. Alagić, "*Relacione baze podataka"*, Svjetlost, Sarajevo, 1984.

[6] Xiaochuan He, "NoSQL analysis and case study of MongoDB", 2017.

[7] Nela Tomić , "NoSQL tehnologije i primjene",2016.

[8] "*MongoDB Manual*", (03.09.2021.), https://docs.mongodb.com/manual/

[9] "Differece between MSSQL server and MongoDB", (08.09.2021.), https://www.geeksforgeeks.org/difference-between-ms-sql-server-and-mongodb/

[10] M. Kamaruzzaman, "*Top 10 databases to use in 2021*", (22.09.2021.), https://towardsdatascience.com/top-10-databases-to-use-in-2021-d7e6a85402ba

[11] J. Rasanayagam, "*NoSQL*", (22.09.2021.), https://medium.com/@R.Sumangala/nosql-ab44a979a831

[12] Oracle, "*What is a relation database"*, (15.09.2021.), https://www.oracle.com/database/what-is-a-relational-database/

[13] IBM, "*NoSQL Databases*" , (12.09.2021.), https://www.ibm.com/cloud/learn/nosql-databases

[14] S. Knight, "*MongoDB vs SQL*" , (12.09.2021.), https://www.knowi.com/blog/mongodb-vs-sql/

[15] Nedeljko Šikanjić, Zoran Ž. Avramović, Esad F. Jakupović, Implementation of the Neural Network Algorithm in Advanced Databases, *JITA – Journal of Information Technology and Applications,* PanEuropien University APEIRON, Banja Luka, Republika Srpska, Bosna i Hercegovina, JITA 8(2018) 2:54-63, (UDC: 004.738.5:551.588:551.506), (DOI: 10.7251/JIT1802054S), Volume 8, Number 2, Banja Luka, december 2018 (45-96), ISSN 2232-9625 (print), ISSN 2233-0194 (online), UDC 004

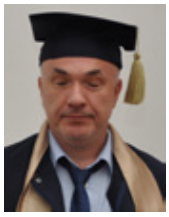
[16] Adrijan Božinovski, George Tanev, Biljana Stojčevska, Veno Pačovski, Nevena Ackovska,Time Complexity Analysis of the Binary Tree Roll Algorithm, *JITA – Journal of Information Technology and Applications*, PanEuropien University APEIRON, Banja Luka, Republika Srpska, Bosna i Hercegovina, JITA 6(2016) 2:53-62, (UDC: 519.857:004.021), (DOI: 10.7251/JIT1602053B), Volume 6, Number 2, Banja Luka, december 2016, ISSN 2232-9625 (Print), ISSN 2233-0194 (Online), UDC 004

Reacived: March 27, 2023
Accepted: May 15, 2023

About the authors

Pero Ranilović, born in the city of Prijedor (Republic of Srpska, BiH). was born on March 11, 1995. Graduated from high school in Novi Grad. Bachelor's degree in Programming and Software Engineering earned at the Faculty of Information Technology at Pan-European University "APEIRON" in Banja Luka. Currently studiyng for a master's degree under the mentorship of Assistant professor Dražen Marinković, PhD. Employed as a software developer since 2018. In addition, hired as a teaching assistant at Pan-European University "APEIRON".

Dražen Marinković was born on 1978 in Banjaluka. He finished elementary school and high school in electrical engineering. The Faculty of Business Informatics at the Pan-European University APEIRON Banjaluka enrolled in the academic year 2009/2010. years and graduated with an average grade of 9.81, defending a master's thesis entitled: "NoSQL databases in theory and practice" with a grade of 10. Academic 2015/2016 enrolls in doctoral academic studies of the third cycle at the Pan-European University APEIRON in Banjaluka, at the Faculty of Information Technology, and in the doctoral program "Information Systems in Communications and Logistics" where he passed all exams with excellent grades - excellent, 10. During his doctoral academic studies, as an author and co-author, he published 15 scientific and professional papers, two of which were published in international peer-reviewed journals.



Nedeljko Šikanjić obtained the degree of Doctor of Science in Informatics and Computer Science and has worked for more than 17 years as a Software and Database Architect/Engineer. His main fields of studies are in the area of advanced Databases and Software Architectures. He has been a holder of an active Microsoft Certified Trainer Certificate since 2012 and has been teaching courses on various topics in Information Technologies. Doctoral studies of the third degree enrolled in the academic 2017/2018.

FOR CITATION
Pero Ranilović, Dražen Marinković, Nedeljko Šikanjić, Comparative Analysis of Relational and Non-Relational Databases, *JITA – Journal of Information Technology and Applications, Banja Luka*, Pan-Europien University APEIRON, Banja Luka, Republika Srpska, Bosna i Hercegovina, JITA 13(2023) 1:20-28, (UDC: 681.322.06:004.652.4), (DOI: 10.7251/JIT2301020R, Volume 13, Number 1, Banja Luka, June (1-56), ISSN 2232-9625 (print), ISSN 2233-0194 (online), UDC 004