



Vojnotehnicki glasnik/Military Technical  
Courier

ISSN: 0042-8469

vojnotehnicki.glasnik@mod.gov.rs

University of Defence  
Serbia

Krsti, Lazar J.; Krsti, Marija S.

TESTING THE PERFORMANCE OF NoSQL DATABASES VIA THE DATABASE  
BENCHMARK TOOL

Vojnotehnicki glasnik/Military Technical Courier, vol. 66, núm. 3, 2018, pp. 614-639  
University of Defence

Available in: <https://www.redalyc.org/articulo.oa?id=661770389008>

- How to cite
- Complete issue
- More information about this article
- Journal's homepage in redalyc.org

redalyc.org

Scientific Information System

Network of Scientific Journals from Latin America, the Caribbean, Spain and Portugal

Non-profit academic project, developed under the open access initiative

СТРУЧНИ ЧЛАНЦИ  
 ПРОФЕССИОНАЛЬНЫЕ СТАТЬИ  
 PROFESSIONAL PAPERS

## TESTING THE PERFORMANCE OF NoSQL DATABASES VIA THE DATABASE BENCHMARK TOOL

Lazar J. Krstić<sup>a</sup>, Marija S. Krstić<sup>b</sup>

<sup>a</sup> Higher Business School of Applied Studies,  
 Leskovac, Republic of Serbia,  
 e-mail: krstic.lazar@vpsle.edu.rs,  
 ORCID iD: <https://orcid.org/0000-0001-9131-6876>

<sup>b</sup> Technical School „Nikola Tesla“,  
 Medvedja, Republic of Serbia,  
 e-mail: krsticmarija1989@gmail.com,  
 ORCID iD: <https://orcid.org/0000-0003-3009-8400>

DOI: 10.5937/vojtehg66-15928; <https://doi.org/10.5937/vojtehg66-15928>

FIELD: Informatics, Databases

ARTICLE TYPE: Professional Paper

ARTICLE LANGUAGE: English

### Summary:

*NoSQL is often used as a successful alternative to relational databases, especially when it is necessary to provide adequate system dimensioning, usage of a variety of data types and high efficiency at a low cost for maintaining consistency. The work is conceived in a manner that covers the general concept of a database, i.e. the concept of relational and non-relational databases, which are substantiated by all important aspects and in an appropriate context. After analyzing the types of NoSQL databases, emphasis is placed on explaining their advantages and disadvantages, as well as on an overview of the NoSQL and SQL database comparison. The final part of the paper presents the results of testing the performance of NoSQL databases, obtained through the Database Benchmark tool. The aim of the paper is to highlight all the details of NoSQL databases in order to establish the justification of their application in practice.*

**Key words:** Databases, Relational databases, NoSQL databases, Performance testing.

## Introduction

Nowadays, data and information are accumulated at a rapid pace from a variety of sources, and it is not easy to understand what they

mean. To cope with the growing amounts of digital data of varied nature, organizations must use sophisticated techniques for information management. Viewed from this angle, databases are essential in all areas of operation of one organization. Relational databases support most of business systems of today's organizations and for a good reason. Functionality and reliability of such databases have been proven in many systems through many years. They have been supported by a number of tools, documented in detail, and there are a large number of people qualified for the implementation and maintenance of these systems. However, both analytical and operational organizations now increasingly take into account different solutions for their business problems, which results in relational databases not always being appropriate for storing and processing data. That is why NoSQL (Not only SQL) databases have been created.

NoSQL databases apply different mechanisms for data storing and establishing relationships across data from relational databases. If data does not need to be stored in tables or there are relations that cannot be presented by classic SQL relationships, and data has to be accessed fast, then NoSQL databases are applied.

### The term database

A database is a collection of interconnected logical related data stored in computer external memory, simultaneously available to users and application programs. It is organized in such a way that a set of computer programs - system for managing databases (Database Management System DBMS) - allows all users to access all data.

A database management system is a set of programs which provide the user with tools to add, delete, access and analyze the data stored in one location. Data can be accessed using queries or reporting tools (which are an integral part of the DBMS) or using application programs written specifically for the purpose of accessing the information. The DBMS also provides mechanisms for the preservation of data integrity, for the management of security and for user's access to information database in case of a crash.

This system minimizes the following issues:

- data redundancy - when the same data is stored in more than one place;
- isolation of data - when an application cannot access the data that is associated with other applications;

- data inconsistency - when different copies of the same data do not match.

On the other hand, the system maximizes:

- data security;
- data integrity - data must meet certain criteria, e.g. there must not be any letters in the field reserved for a personal identification number;
- data independence - application software and data are independent from each other (i.e. applications and data are not related to each other, which means that different applications can access the same data).

Defining a database requires not only a specification of types and data structures that need to be memorised into the base, but also the limitations of the data. In this regard, Data Definition Language - DDL is used for the execution of these activities. Data definitions are located in the system directory and are called metadata.

Add, delete, modify and search data represent the activities of manipulating databases by the help of Data Manipulation Language DML. A DML part that searches a database is called a query language. Figure 1 represents a simplified database system (<http://sakshieducation.com>).

Searching data in the database is probably the most frequently executed process. Structured Query Language (SQL) is the most popular language for data search. It allows the execution of complex search using relatively simple phrases or keywords. Typical keywords are SELECT (to determine the desired attribute), FROM (to determine the tables that will be used) and WHERE (to determine the conditions which will apply to search). For example: SELECT the name of the student, FROM a database of students, WHERE the average score > 9.00.

Another way to find data in the database is by using search examples (Query By Example - QBE). In the QBE search system, users fill out a template (also called a form) to define an example or a description of the desired information. Users can quickly and easily construct a questionnaire using the techniques of drag-and-drop in the DBMS. Such a search is easier than typing SQL commands.

There are many different models, such as the database Entity-Relationship model (ER model), hierarchical models, network models, relational models and object model. The most popular model is undoubtedly the Entity-Relationship model (ER model).

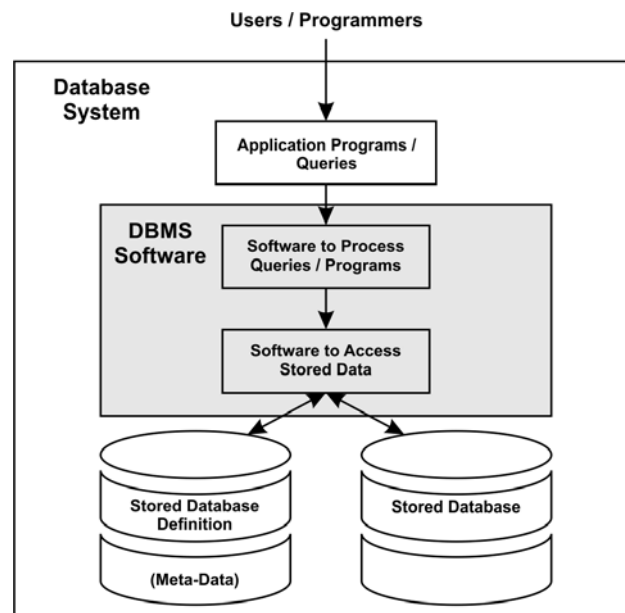


Figure 1 – Simplified database system  
 Рисунок 1 – Упрощенная система баз данных  
 Слика 1 – Поједностављен систем базе података


## Relational databases

Relational databases contain and manage relational structured data, and have a system for manipulating them. The data is stored in a two-dimensional table consisting of rows and columns. The interaction with the relational database management system, in most cases, is implemented by using SQL query language. Types of data that could be stored are determined by the system while information is organized using a clearly determined scheme, typically based on the Entity Relationship (ER) model. The relational database model and SQL as a standard query language are widely accepted and were considered for long to be the only alternative to storing and organizing data which can be accessed by more than one user in a consistent manner.

Since there is a data structure, it is necessary to follow it, including the classification of data into the table by rows and columns (Figure 2). The intersection of a column and a row has only one value, and each variable has a data type, each of which has a limited domain of allowed values. Primary keys are placed in order to be able to identify each row in the table, while the foreign keys are used to interconnect multiple tables.

Because of this structure type, relational databases are vertically scalable, which means that they work best on a single node, that is, on a single computer. If expansion is necessary due to a growing amount of data, the best option is to increase the resource of that node.

Table Name - Student			
#IndexNumber	Name	Age	\$DirectionCode
1234	Petar	22	F1
2345	Marija	21	F1
3456	Milena	21	M2

 Primary Key

 Foreign Key

 Record (Row)

 Attribute Names (Column)


 Individual Value of Attribute

Figure 2 – Table (basic terms and components)

Рисунок 2 – Таблица (основные термины и компоненты)

Слика 2 – Табела (основни појмови и саставни елементи)

### Non-relational database

As their name suggests, non-relational databases are not based on the relational model, or at least do not stick to it firmly, and include data without schemes. They are widely distributed systems that enable rapid organization and analysis of large quantities of various types of data. They are simple and horizontally scalable, meaning you can easily add a new node (computer) which does not affect the operation of the system (Strauch, 2011). Unlike relational databases, where the ACID (Atomicity, Consistency, Isolation, Durability) properties are applied, non-relational databases are based on the BASE (small plates, Soft state, are/is possible consistency) properties. The BASE properties certainly do not mean that NoSQL databases are unreliable and inconsistent, but eventual consistency means that the system will become consistent once all data is propagated into all nodes in the cluster.

### ACID and BASE models

The ACID and BASE are models of control over transactions in order to ensure consistency. The difference between them depends on a layer in which they are: the ACID model is in the base layer, while the BASE model is in the application layer. Also, unlike the ACID model which is oriented to consistency, systems that use the BASE model focus on accessibility.

Relational databases support the ACID properties of transactions:

1. Atomicity - transactions must be done fully or not at all, which means that systems that provide atomicity must be prepared for all possible problems (hardware and software problems, network failures, disk problems or full system crashes);
2. Consistency - data must be entirely consistent;
3. Isolation - operations are mutually isolated (no part of the transaction is aware of the others);
4. Durability - after all transaction elements are complete, the transaction becomes permanent.

NoSQL databases follow the BASE properties:

1. Basic Availability - the system is allowed to be temporarily inconsistent in order to carry out transactions;
2. Soft state - some inaccuracies are allowed for a short time and the data can be changed while being used, in order to reduce the consumption of resources;
3. Eventual consistency - after all operations are performed, the system will become consistent.

### *The CAP theorem*

The ACID and BASE concepts should not be seen as mutually exclusive options, but as a spectrum. The BASE concept is often mentioned along with another important concept, the CAP (Consistency, Availability, and Partition Tolerance) theorem. This proposition can clarify why it is sometimes necessary to sacrifice consistent work for other properties. For NoSQL databases, according to the CAP theorem, only two of the next three properties can be guaranteed at the same time (Kumar, nd):

1. Consistency - all nodes have identical copies of the data available for the transaction;
2. Availability - every request for information will be processed successfully or there will be a message that the request cannot be processed;
3. Partition Tolerance - the system will continue to operate in the case of breaking the link between the nodes, which would create partitions in which nodes can communicate with each other only within their own partition.

### NoSQL - „Not Only SQL“

The term NoSQL ("Not only SQL") is the term that describes the entire class of databases which do not have the characteristics of

traditional relational databases and for which standard query SQL language is not generally used. NoSQL databases are considered to be the next generation databases and may be defined as non-relational, distributed databases, open source, and horizontally scalable. They are characterized by a less strict static data structure, simple support to replication and simple application programming interface (API). They are often related to large data sets that need to be quickly and efficiently accessed and changed on the Web.

### *Basic characteristics of NoSQL databases*

The basic characteristics of NoSQL databases are:

1. Distributed computing (Scalability, Reliability, Resource Sharing, Performance) - NoSQL databases are distributed, provide horizontal scalability, and handle large amounts of data e.g. several of terabytes or petabytes with small time delay. A parallel increase in the number of users and amount of data requests relevant data from the Web and mobile applications, as well as from accompanying databases, which can be achieved by using the following two methods:
  - a. Scale up - this involves adding resources to a single node by setting an additional processor or by increasing storage memory.
    - Vertical scalability with relational databases: to support a large number of simultaneous users and/or store more data, there is a need for big servers with more processors that will handle that workload, more memory and more storage space for all tables. Installing and maintaining large servers is a complex task that requires large investments (costs).
  - b. Scale out - refers to adding new nodes to the system (for example, adding new computers to the distributed software application).
    - Horizontal scalability with NoSQL databases: with this method, the resource is added to a cluster of servers for data storage and supporting operations with databases. The cluster is expanded by adding additional servers to spread database processing to a larger cluster. As servers are vulnerable to failure, NoSQL databases are designed to withstand and recover from constant failures, which makes them very resistant. NoSQL databases enable a simpler, straight-line approach to scaling databases. They are best at dealing with sudden jumps in activities of new users. In order to cope with sudden jumps in volume, a new database server needs to be added to expand the cluster.



2. More flexible data model - NoSQL databases appear in one of the following models:
  - key-value databases,
  - document-oriented databases,
  - wide column databases,
  - graph databases.
3. Asynchronous insertions and updates / low transactional - Complete transactional guarantees and simultaneous execution of transactions in all nodes in a distributed environment are not provided by NoSQL databases. Instead, they guarantee the availability of data at the distributed level (through an internal synchronization). This is why NoSQL is a perfect model for applications such as social networking, where simultaneous transactions are not a limitation.
4. Follow BASE/CAP instead of ACID - Instead on ACID, NoSQL databases work more or less on the BASE principle. All NoSQL databases rely more or less on the ACID properties (CAP theorem). For example, when there are no updates for a while (this can mean a few seconds), all updates can propagate through the system at the very end, which depends on the loads, the size of the cluster and the network traffic, and it will make all nodes consistent.
5. Query language - These databases do not support SQL, unlike relational databases.
6. NoJoins - NoSQL databases do not use the concept of linking.
7. Low cost – They use clusters of low-cost, already available equipment (servers) instead of own servers for managing large amounts of data and transactions.
8. Easy implementation – They provide flexibility and simpler schemes, unlike the Relational Database Management System (RDBMS).

### *Types of NoSQL databases*

There are four basic types of databases that are classified under the category of NoSQL (Sadalage, 2014):

1. key-value databases,
2. document-oriented databases,
3. wide column databases,
4. graph databases.

### *Key-value databases*

Key-value databases can be compared to a table in a relational model that has two columns, the key and the value. The data is stored in

distributed hash maps, where the key is most often a string, while the value can be one of the types supported by all programming languages, such as strings, numbers, arrays, or objects. These databases store a variety of data, but do not perform any additional data search by multiple criteria.

- Advantages:
  - working with large amounts of data,
  - very fast,
  - usually support automatic replication and horizontal partition of collections.
- Disadvantages: (Graovac, 2016):
  - a high level of redundancy,
  - complex structures are implemented by a large number of collections
  - if the data is "densely" linked, the efficiency drops dramatically,
  - they have no mechanisms for the preservation of the integrity - often do not provide even transaction atomicity,
  - search condition is exclusively a fixed key value or a range of key values.

The most popular key-value databases are: DynamoDB (Amazon), Riak, Redis, Voldemort, and Oracle NoSQL.

### *Document-oriented databases*

With document-oriented databases, data is organized as a collection of documents with possibly different structures, which supports simple adding and dropping of attributes. These bases store XML, JSON, BSON formats of documents (tagged format) or, for example, PDF format documents (unstructured format). Data is not normalized.

- Advantages:
  - very simple and efficient operation,
  - usually support at least semi-automatic replication and horizontal collection partition - often just a replication of the main-subordinate type.
- Disadvantages:
  - relatively limited domain of application,
  - many implementations do not allow ad hoc querying and changing data,
  - some implementations do not stand high frequency data changes.

Some of the most popular document-oriented databases are: CouchDB, MongoDB, RavenDB, Couchbase, Azure DocumentDB.

### *Wide column databases*

In wide column databases, data is grouped in columns, which gives a better performance when there is a need for queries that should restore only certain attributes, and not full entities. They operate with the terms *column* and *super column*. The columns have a name, a value, and a timestamp, which indicates that there is no need to define a scheme.

- Advantages:
  - support very large amounts of data,
  - they are very fast, except in some cases of values with very complex structures
  - most support automatic replication and horizontal collection partition.
- Disadvantages:
  - a high level of redundancy,
  - complex structures are implemented by a large number of collections,
  - if the data is "densely" linked, the efficiency drops dramatically,
  - they have no mechanisms for the preservation of integrity - often do not provide even transaction atomicity,
  - search condition is exclusively a fixed key value or a range of key values.

Some of the most popular wide column databases are: Cassandra (Facebook, now part of the Apache Software), Hadoop / Hbase, Accumulo, Hypertable, Amazon SimpleDB.

### *Graph databases*

In graph-oriented databases, data is represented in the form of graphs, with entities being represented by nodes, and their relations by the edges of the graph. Each link and node carry certain information on the basis of which quick inquiries can be made. Searching data by links shows great performance advantages.

- Advantages:
  - in contrast to other non-relational databases, they are very effective when it comes to common operations with graphs,
  - some support transactions and the ACID mode of conditions,
  - usually only the replication of the main-subordinate type.
- Disadvantages:
  - relatively restricted application domain,
  - not suitable out of their domain.

Some of the most popular databases oriented towards graphs are: Neo4J, AllegroGraph, OrientDB, ArangoDB, Infinite Graph.

### *The general criteria for the selection of NoSQL databases*

General criteria that need to be taken into consideration before making a decision about which NoSQL database to use are as follows:

1. Storage type - one of important criteria that should be considered when choosing an NoSQL database.
  - For example, get, put and delete functions are best supported by key-value systems.
  - Aggregation becomes much simpler when using column-oriented systems. They use tables, but without joining.
  - Data mapping becomes easier with object-oriented software using NoSQL databases based on documents, such as XML or JSON.
  - A tabular format is changing, and data is saved in the graphical format.
2. Control of parallel execution - specifies how two users may simultaneously modify the same information. It often happens that one of the users loses access, so he/she cannot change or perform actions, while an active user ends editing (Kumar, nd).
  - Lock - prevents more than one active user from changing an entity such as a document, a line or an object.
  - Multiversion concurrency control (MVCC) - provides a readable overview of the base, but leads to conflicting versions of an entity, if more than one user perform changes at the same time. MVCC enables seemingly smooth processing of transactions by creating multiple versions of a site. This means that the consistency of transactions is retained, although different users at any given moment are shown various displays. All changes to the database will be displayed to all users depending on which view they are viewing.
  - In some systems there is a lack of atomicity due to which all users who modify the database do not have the same base overview.
  - ACID should be chosen if reliable transactions are needed.
3. Replication - enables continuous synchronization of backups.
  - Synchronous mode - this approach (although expensive due to dependence on another server to respond) always provides consistency.
  - Asynchronous mode - with this approach, the database update is done without waiting for a response from another database. There is a small scale of inconsistencies that can last a few milliseconds.

4. Language for implementation - assists in determining how fast the database processing can be performed. NoSQL databases written in low-level programming languages are most often the fastest. On the other hand, those written in higher-level programming languages are easier to modify.

### *Examples of applying the four basic types of NoSQL databases*

The next part of the paper provides some examples of applying the four basic types of NoSQL databases, as well as some of the specific criteria for determining which type meets specific requirements.

#### *Examples of application of key-value databases*

The databases of type key-value are suitable for applications that have frequent short readings and typing with simple data models. The values stored in this database type can be a simple scalar value such as an integer or a boolean, but they can also be structured data such as lists and JSON structures. Generally, have the option of a simple query that allows identifying a value over its key. Some key-value databases support search functions and are therefore more flexible.

The database of the key-value type is used in a large number of applications, such as (Sullivan, nd):

- caching data from relational databases to boost performance,
- monitoring temporary attributes in a Web application, such as a shopping cart,
- storing configurations and user information for mobile applications,
- storing large objects, such as images and audio files.

#### *Examples of the application of document-oriented databases*

Databases oriented towards documents are designed to be flexible. If an application requires the ability to put different attributes together with large amounts of data, then this database is the right solution. For example, in order to present products in a relational database, the model maker can use a table for common features and additional tables for each product subtype in order to store the attributes that occur only in product subtypes. Databases oriented towards documents are the best choice for these situations.

Databases oriented towards documents are used for embedded documents which are good for denormalization. Instead of storing data in different tables, data which is often searched together is placed in the same document. In addition, these databases improve the ability of key-

value databases by indexing and by the ability to filter documents based on attributes in the document itself.

Databases oriented towards documents are perhaps the most popular NoSQL databases because of their flexibility, performance, and ease of use. They can be applied in the following situations:

- back-end support for sites with large numbers of hits and data,
- management of data with variable attributes, such as products,
- tracking changeable metadata types,
- applications that use the JSON data structure,
- applications that take advantage of denormalization by embedding structures into the existing structures.

Databases oriented towards documents are also available as cloud service (Microsoft Azure Document and Cloudant's database).

#### *Examples of the application of wide column databases*

Wide column databases are designed for large amounts of data, for reading and registration performances and they are extremely affordable. Google's Bigtable was designed for its needs, while Facebook developed Cassandra to enable the Inbox search service to its customers. These database management systems function on the principle of a cluster of multiple servers.

Wide column databases can be used in the following cases:

- applications that require a competence of permanent entries in the database,
- applications that are geographically distributed across multiple data centers,
- applications that tolerate some short-term inconsistencies in responses,
- applications with dynamic fields,
- applications with potentially vast amounts of data, for example, thousands of terabytes.

#### *Examples of the application of graph databases*

Databases oriented towards graphs are suitable for application in problems that can be solved using graphical structures for data saving and storage. One way to evaluate the utility of databases oriented towards graphs is to check if each element contains a direct pointer to the next element (if one element is connected with other elements).

For example, two orders in electronic trade probably are not linked to each other. They may have been sent by the same customer but that is a common attribute, not a link. Similarly, the configuration and progress in a game of one gamer probably has nothing to do with the configuration of another game of some other gamer. In these cases, the most commonly used are key-value databases, document oriented or relational databases.

On the other hand, examples such as railway tracks that connect cities, proteins that are intertwined with other proteins and workers who collaborate with other workers are cases where there is some kind of connection or a direct relationship between the two instances of the entities. These are the types of problems where database oriented towards graphs can be used.

Other examples include:

- online management and IT infrastructure management,
- identity and access management,
- business process management,
- recommending products and services,
- social networks.

The above examples clearly show that, when it is necessary to make a model of explicit and fast connections between the entities, databases oriented towards graphs should be used.

### *NoSQL advantages and disadvantages*

Table 1 presents the pros and cons of NoSQL databases.

*Table 1 – NoSQL advantages and disadvantages*  
*Таблица 1 – NoSQL преимущества и недостатки*  
*Табела 1 – NoSQL предности и mane*

Advantages of NoSQL	Disadvantages of NoSQL
High scalability	Too many options (over 150), so it is hard to make a decision
Flexibility of the scheme	Limits query (for now)
Distributed computing (scalability, reliability, resource sharing, speed)	Eventual consistency is not intuitive to programs such as banking applications
There are no complicated connections	Lack of Joins, Group by, Order by features
Lower costs	ACID transactions
Open Source - all NoSQL options with the exception of Amazon S3 (Amazon Dynamo) are open source solutions	Limited support - open source

### Comparison between NoSQL and SQL

Table 2 presents the difference between NoSQL and SQL (<https://azure.microsoft.com>).

Table 2 – Comparison between NoSQL and SQL

Таблица 2 – Сравнение между NoSQL и SQL

Tabela 2 – Поређење између NoSQL и SQL

	NoSQL	SQL
Model	Non Relational	Relational
	Storing data in JSON documents, key-value pairs, column-oriented warehouses or graphs	Storing data in tables
Data	Flexibility due to the fact that all records do not have to store the same characteristics	Excellent for solutions where each record has the same characteristics
	Quickly add new features	Adding a new feature requires modifying the scheme
	Links are graded by denormalizing data and presenting all data for one entity in a single record	Links are built in normalized models by joining tables
	Suitable for semi-structured, complex or nested data	Suitable for structured data
Scheme	Dynamic or flexible schemes	Clearly defined schemes
	Database does not accept the scheme, and it is specified by the application. This results in agility and highly interactive development	Scheme must be maintained and be synchronized between applications and databases
Transactions	ACID transaction support varies depending on the solutions	Supports ACID transactions
Consistency and availability	Supports eventual consistency to strong consistency, depending on the solutions	Strong consistency
	Consistency, availability, and performances may vary depending on the application requirements (CAP theorem)	Consistency has priority over availability and performance
Performances	Performances can be maximised on account of consistency (if necessary)	Insertion and update performances depend on the speed of record creation, with strong consistency Performances can be maximized by scaling available resources
	All information about an entity are generally in a single record, so an update can be performed in a single operation	Information about the entity can be in many tables and rows, so it requires joining to perform updates or queries
Scalability	Scalability is mainly horizontal and data is distributed on multiple servers	Scalability is mainly vertical with more server resources



## Testing the performances of NoSQL databases

Database performance testing is one of the areas with few open source tools. This may be due to the fact that most relational databases are commercial tools that come with the existing infrastructure provided by the seller. However, the current rise in NoSQL databases may still change this situation in the future.

One of the most popular, and by many the most popular open source tool for testing database performance is Yahoo! Cloud Service Benchmark (YCSB). This tool enables testing of different systems and their comparison; for example, on the same hardware configuration, multiple systems can be installed over which an identical load scenario will be run, so that eventually their characteristics can be compared. It consists of a YCSB client, a load generator, and a set of basic load scenarios (Cooper et al, 2017). The YCSB client is a Java program for generating data that will be added to the database and generate operations, which are the workload scenario. The Load Scheduler manages multiple threads of the client. Each thread executes a sequential series of operations by calling the database interface layer to load it (load phase) and to execute the load scenario (transaction phase). Threads also regulate the rate at which requirements are regulated, so it is possible to directly control the offered load on the database. Threads measure both latency and the achieved bandwidth of their operations and pass the measurements to the statistical module. Finally, the statistical module collects the measurements and reports on the average values.

Performance testing using the YCSB tool requires first a correct installation and configuration of the database management system. The configuration refers to the creation of known "template tables" (depending on the data model, as well as on the load scenario), within which the tool itself will generate data. The next step involves selecting a correct database interface layer generated by the Java class that will perform reading, writing, updating, deleting, and searching. Then a load scenario is selected from the basic set or an arbitrary one is created, to be followed by the selection of the appropriate execution parameters (number of client threads, targeted bandwidth, etc.). The last step is to load the desired amount of data into the database, and start the simulation.

Analyzing a large number of scientific and professional papers whose content is focused on testing the performance of NoSQL databases has led to two important conclusions:

- YCSB tool was used for performance testing;
- the emphasis is on testing the performance of some of the most popular NoSQL databases, such as: MongoDB, Redis, CassandraDB, Couchbase, Hbase.

Given the above facts, this part of the paper will present the results of NoSQL databases performance tests obtained using Database Benchmark. This is still one of the few tools for this purpose that has built-in support for some of the most popular NoSQL databases. The computer configuration presented in Table 3 test was used to test five less popular (but not of a lesser value) NoSQL databases: HamsterDB, LevelDB, STSdb 4.0 (database type key-value), RavenDB (document oriented database) and BrightstarDB (graphs oriented database).

Database Benchmark is one of the most powerful open-source tools for testing the performance of bases with large amounts of data. The application has two main test scenarios:

- inserting large amounts of randomly generated records with sequential or random keys and
- reading of the inserted records according to their keys.

### *Testing performance of NoSQL databases via the Database Benchmark tool*

The limited-resource computer system whose specifications are presented in Table 3 was used for testing the performance of five NoSQL databases: HamsterDB, LevelDB, STSdb 4.0, RavenDB and BrightstarDB. Database Benchmark, a tool that was used to perform the measurement itself, was selected to keep the NoSQL databases running in different ways at approximately the same level, so that the obtained measurement results could be relatively realistically compared. The following parameters were measured:

- speed insertion of all records generated with random keys, and
- speed reading of all inserted records according to their keys.

*Table 3 – Computer configuration (Acer Aspire 5750G)*  
*Таблица 3 – Конфигурација компјутера (Acer Aspire 5750G)*  
*Табела 3 – Рачунарска конфигурација (Acer Aspire 5750G)*

	Acer Aspire 5750G
Processor	Intel® Core™ i3-2310M CPU @ 2.10 GHz
Random access memory	6GB DDR3 @ 1333 MHz
Storage	640GB SATA 5400rpm
Operating system	Microsoft Windows 10 Pro - 64 bit

Testing the performance of databases via the Database Benchmark tool includes the following steps:

- select the databases to be tested,
- select the number of data streams to be inserted (tasks),
- select the number of records to be generated for each stream of data (records),
- Select the type of generated keys for all streams (keys). Keys can be sequential or random.

### *Speed of insertion of all records generated with random keys*

Speed of insertion of all records generated with random keys was measured in three different circumstances:

1. Number of data streams = 1; The number of records for each data stream = 50,000; Key type = random;
2. Number of data streams = 2; The number of records for each data stream = 50,000; Key type = random;
3. Number of data streams = 5; The number of records for each data stream = 50,000; Key type = random.

The obtained measurement results are presented in Figures 3 to 6.

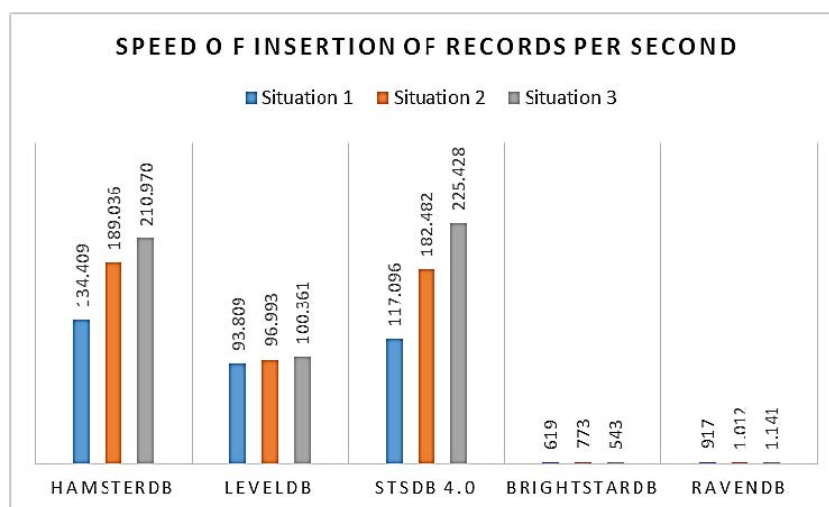


Figure 3 – Speed of insertion of records per second  
Рисунок 3 – Скорость вставки записей в секунду  
Слика 3 – Брзина уметања записа по секунди

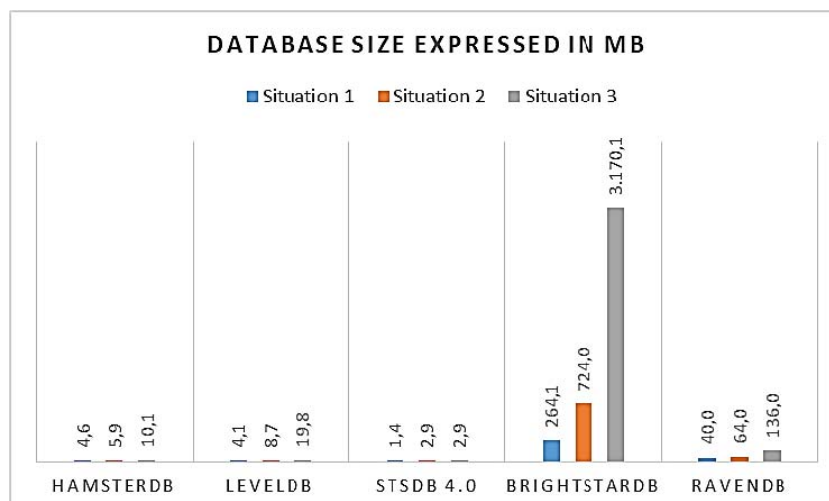


Figure 4 – Database size expressed in MB  
 Рисунок 4 – Размер базы данных, выраженный в MB  
 Слика 4 – Величина база података изражена у MB

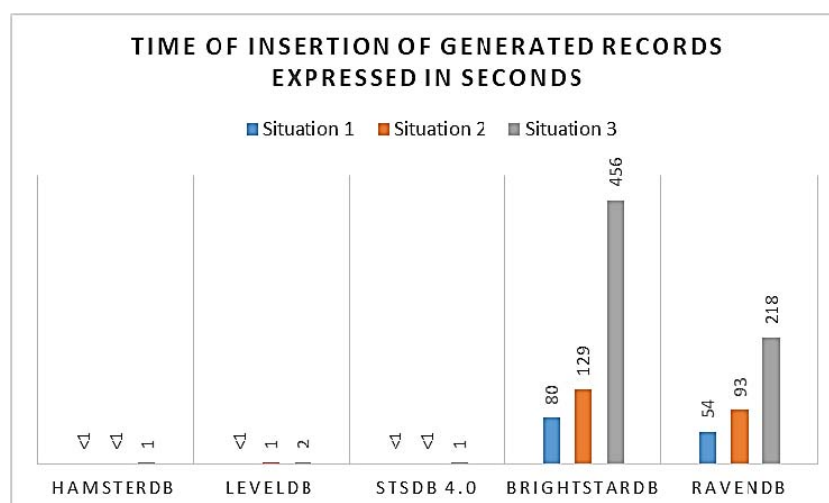


Figure 5 – Time of insertion of generated records expressed in seconds  
 Рисунок 5 – Время вставки сгенерированных записей, выраженное в секундах  
 Слика 5 – Време уметања генерисаних записа изражено у секундама

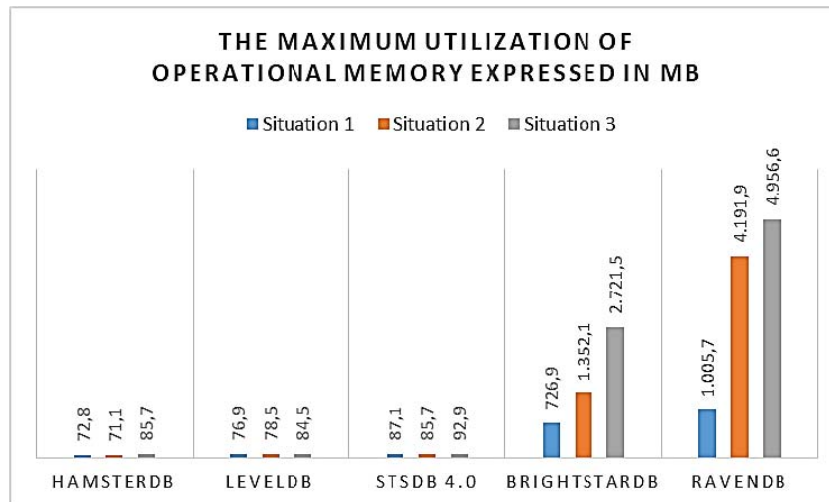


Figure 6 – Maximum utilization of operational memory expressed in MB  
 Рисунок 6 – Максимальное использование оперативной памяти, выраженное в MB  
 Слика 6 – Максимум искоришћења оперативне меморије изражен у MB

### Speed of reading of all inserted records according to their keys

Speed of reading of all inserted records by their keys was measured in three different circumstances:

1. Number of data streams = 1; The number of records for each data stream = 50,000; Key type = random;
2. Number of data streams = 2; The number of records for each data stream = 50,000; Key type = random;
3. Number of data streams = 5; The number of records for each data stream = 50,000; Key type = random.

The obtained measurement results are presented in Figures 7 to 10.

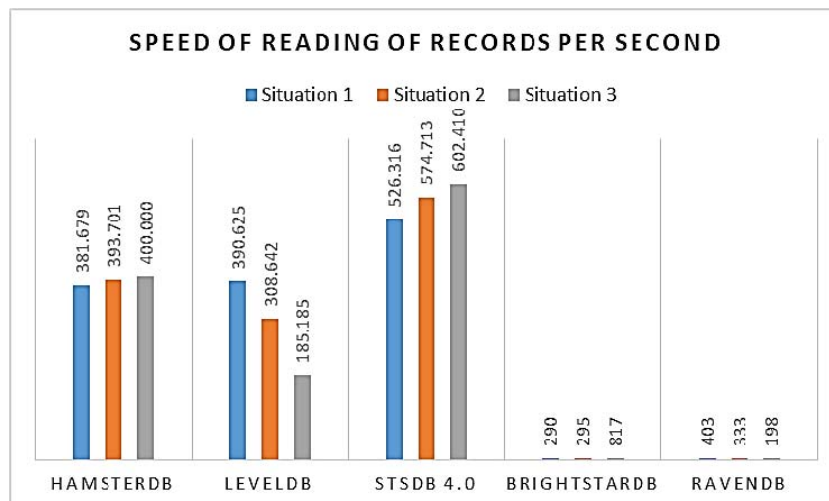


Figure 7 – Speed of reading of records per second  
 Рисунок 7 – Скорость чтения записей в секунду  
 Слика 7 – Брзина читања записа по секунди

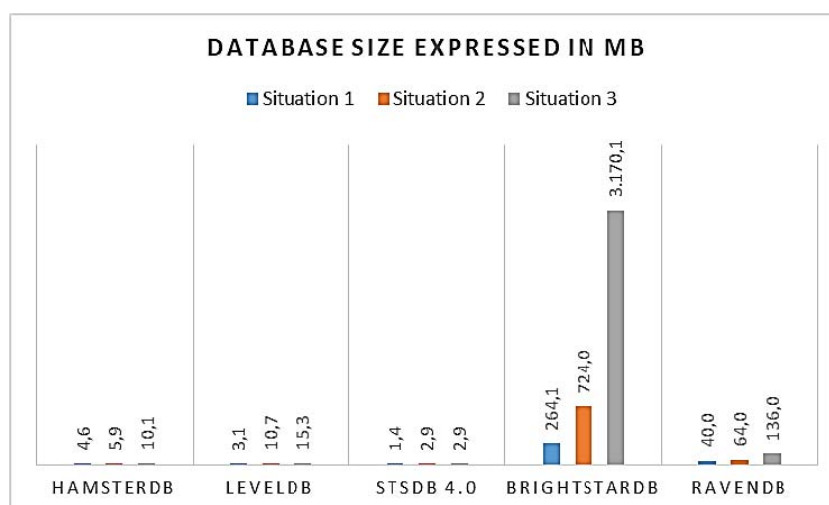


Figure 8 – Database size expressed in MB  
 Рисунок 8 – Размер базы данных, выраженный в MB  
 Слика 8 – Величина база података изражена у MB

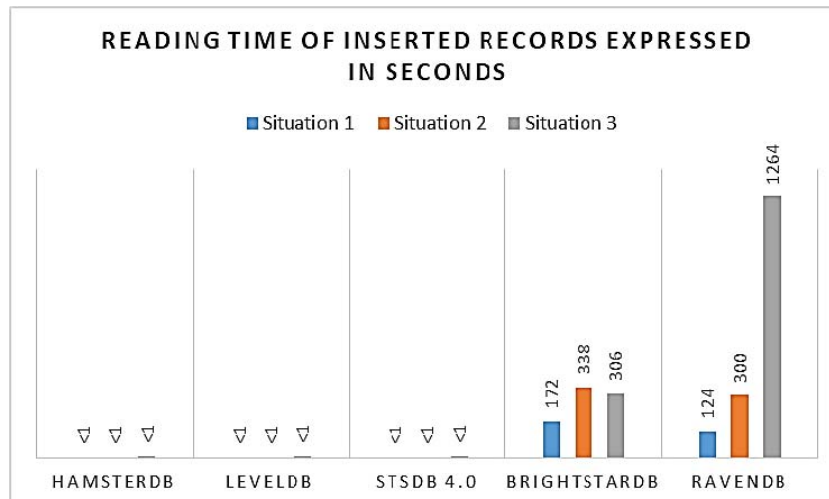


Figure 9 – Reading time of inserted records expressed in seconds  
 Рисунок 9 – Время чтения вставленных записей, выраженное в секундах  
 Слика 9 – Време читања уметнутих записа изражено у секундама

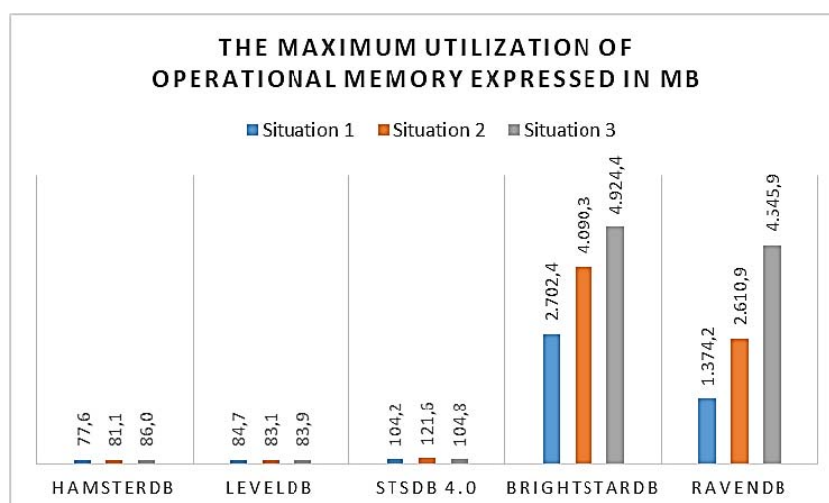


Figure 10 – Maximum utilization of operational memory expressed in MB  
 Рисунок 10 – Максимальное использование оперативной памяти, выраженное в MB  
 Слика 10 – Максимум искоришћења оперативне меморије изражен у MB

### *Analysis of the obtained measurement results*

The analysis of the speed of inserting all records generated with random keys in situations where the number of data streams is 1, 2 or 5, the number of records for each data stream is 50,000, and the key is random can be presented as follows:

- in all three situations, HamsterDB has the highest speed of insertion of records per second, while the worst results are with BrightstarDB,
- in all three measurements, STSdb has the smallest value, while BrightstarDB has the largest value,
- the time it takes to insert all records generated with random keys in all three situations is almost similar when it comes to HamsterDB, LevelDB and STSdb, while the lowest time is achieved by BrightstarDB,
- the maximum utilization of the operating memory varies depending on the particular database, but the best result in two of the three cases was shown by HamsterDB, while the Raven showed the worst.

The analysis of the speed of reading all the inserted records according to their keys in situations where the number of data streams is 1, 2 or 5, the number of records for each data stream is 50,000, and the key type is random can be presented as follows:

- STSdb has the highest reading speed of inserted records per second in all three cases of measurement, while the smallest read speed in two of three cases has been shown by BrightstarDB,
- in all three cases, the STSdb measurement has the smallest value, while BrightstarDB has the largest value,
- the time it takes to read all of the inserted records according to their keys in all three situations is almost similar when it comes to HamsterDB, LevelDB and STSdb, while much worse time is achieved by BrightstarDB and RavenDB,
- with the maximum use of operational memory, the best result in two of the three cases was shown by LevelDB, while BrightstarDB proved to be the worst.

### **Conclusion**

Information technologies and information systems support organizations in data management, from collecting, organizing, storing, accessing, to analyzing and interpreting data. For enormous amounts of



available data that are growing every day, relational databases are not always the best solution for managing and storing them. In this sense, big companies like Google, Facebook and Amazon have played a significant role in reviving NoSQL technology.

In order to select the best database, it is necessary to see the advantages and disadvantages of both relational and NoSQL databases. Thus, for example, if structured data are used where the consistency of data in transaction systems is very important, the right solution is a relational database. If, on the other hand, it is necessary to process unstructured data where speed and availability are important but consistency is not to such an extent, the advantage is with NoSQL databases. NoSQL databases are much cheaper than the known relational databases and do not require expensive licenses and hardware. However, this should in no way be taken as an excuse for selecting any of NoSQL databases before considering their intended purpose.

NoSQL databases have their drawbacks, but as relatively new technology, they have space for improvement. Although it can only be guessed what will happen in the future with their development, it is certain that the NoSQL market will grow significantly.

Considering the performance analysis of some less popular NoSQL databases, a concrete conclusion can be drawn that HamsterDB has the best performance, while the worst is BrightstarDB. This conclusion was expected before the start of the actual performance measurement.

## References

Cooper, B., Silberstein, A., Tam, E., Ramakrishnan, R., & Sears, R. 2017. *Benchmarking Cloud Serving Systems with YCSB*. [Internet]. Available at: <https://www.cs.duke.edu/courses/fall13/cps296.4/838-CloudPapers/ycsb.pdf>. Accessed: 10 Oct 2017.

Graovac, J. 2016. *Projektovanje baza podataka*. [Internet]. Available at: [http://poincare.matf.bg.ac.rs/~jgraovac/courses/projbp/2016\\_2017/projbp\\_skripta.pdf](http://poincare.matf.bg.ac.rs/~jgraovac/courses/projbp/2016_2017/projbp_skripta.pdf) (in Serbian). Accessed: 9 Nov 2017.

Kumar, K. 2017. *Selection criteria for NoSQL database part III*. [Internet]. Available at: <https://www.3pillarglobal.com/insights/selection-criteria-for-nosql-database>. Accessed: 20 Sep 2017.

Kumar, K. *Just say yes to NoSQL part I*. [Internet]. Available at: <https://www.3pillarglobal.com/insights/just-say-yes-to-nosql>. Accessed: 20 Sep 2017.

Sadalage, P. 2014. *NoSQL Databases: An Overview*. [Internet]. Available at: <http://www.informit.com/articles/article.aspx?p=2266741>. Accessed: 9 Nov 2017.

Strauch, C. 2011. *NoSQL Databases*. [Internet]. Available at: <http://www.christof-strauch.de/nosql dbs.pdf>. Accessed: 30 Sep 2017.

Sullivan, D. *Types of NoSQL databases and key criteria for choosing them*. [Internet]. Available at: <http://searchdatamanagement.techtarget.com/feature/Key-criteria-for-choosing-different-types-of-NoSQL-databases>. Accessed: 9 Oct 2017.

<https://azure.microsoft.com/en-us/services/cosmos-db/?v=17.45b>. Accessed: 20 Oct 2017.

<http://sakshieducation.com/Engineering/StudyStory.aspx?nid=100042&cid=11&sid=666&chid=1107&tid=664>. Accessed: 2 Nov 2017.

## ТЕСТИРОВАНИЕ ВОЗМОЖНОСТЕЙ БАЗЫ ДАННЫХ NoSQL С ПОМОЩЬЮ DATABASE BENCHMARK ИНСТРУМЕНТА

Лазар Й. Крстич<sup>а</sup>, Мария С. Крстич<sup>б</sup>

<sup>а</sup> Высшая школа профессионального обучения,  
г. Лесковац, Республика Сербия

<sup>б</sup> Техническая школа «Никола Тесла»,  
г. Медвежья, Республика Сербия

ОБЛАСТЬ: информатика, базы данных  
ВИД СТАТЬИ: профессиональная статья  
ЯЗЫК СТАТЬИ: английский

### Резюме:

База данных NoSQL часто используется в качестве удачной альтернативы реляционным базам данных, особенно в тех случаях, когда необходимо обеспечить соответствующие размеры системы, применение данных различных типов и высокую эффективность по низкой стоимости хранения и поддержки консистентности данных. В данной работе представлено общее определение базы данных, описаны различные виды баз данных как реляционных, так и нереляционных, выявлены их преимущества и недостатки в соответствующем контексте. Выявленные преимущества и недостатки баз данных NoSQL и SQL проанализированы, и результаты сравнительного анализа представлены в таблице. В заключительной части статьи представлены результаты тестирования возможностей базы данных NoSQL, полученные с помощью инструмента Database Benchmark. Цель данной работы заключается в выявлении всех характеристик базы данных NoSQL, ради лучшего понимания обоснованности их использования на практике.

Ключевые слова: базы данных, реляционные базы данных, NoSQL базы данных, тестирование возможностей.

## ТЕСТИРАЊЕ ПЕРФОРМАНСИ NoSQL БАЗА ПОДАТАКА ПОМОЋУ DATABASE BENCHMARK АЛАТА

Лазар Ј. Крстић<sup>а</sup>, Марија С. Крстић<sup>б</sup>

<sup>а</sup> Висока пословна школа струковних студија, Лесковац, Република Србија

<sup>б</sup> Техничка школа "Никола Тесла", Медвеђа, Република Србија

ОБЛАСТ: информатика, базе података

ВРСТА ЧЛАНКА: стручни чланак

ЈЕЗИК ЧЛАНКА: енглески

### Сажетак:

*NoSQL база података често је успешна алтернатива релационим базама података, посебно када је потребно обезбедити адекватно димензионирање система, коришћење разноврсних типова података и високу ефикасност уз ниске трошкове одржавања конзистентности. У раду је наведен општи појам базе података, односно појам релационих и нерелационих база података, које су објашњене са свих значајнијих аспеката и у одговарајућем контексту. Након анализирања врста NoSQL база података, тежиште је на образлагању њихових предности и недостатака и упоредном прегледу анализе поређења NoSQL и SQL база података. У последњем делу рада представљени су резултати тестирања перформанси NoSQL база података добијених применом Database Benchmark алата. Циљ рада јесте истраживање свих појединости NoSQL база података ради утврђивања оправданости њихове примене у пракси.*

*Кључне речи: базе података, релационе базе података, NoSQL базе података, тестирање перформанси.*

Paper received on / Дата получения работы / Датум пријема чланка: 06.12.2017.

Manuscript corrections submitted on / Дата получения исправленной версии работы / Датум достављања исправки рукописа: 09.01.2018.

Paper accepted for publishing on / Дата окончательного согласования работы / Датум коначног прихватања чланка за објављивање: 11.01.2018.

© 2018 The Authors. Published by Vojnotehnički glasnik / Military Technical Courier (www.vtg.mod.gov.rs, втг.мо.унр.срб). This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution license (<http://creativecommons.org/licenses/by/3.0/rs/>).

© 2018 Авторы. Опубликовано в «Военно-технический вестник / Vojnotehnički glasnik / Military Technical Courier» (www.vtg.mod.gov.rs, втг.мо.унр.срб). Данная статья в открытом доступе и распространяется в соответствии с лицензией «Creative Commons» (<http://creativecommons.org/licenses/by/3.0/rs/>).

© 2018 Аутори. Објавио Војнотехнички гласник / Vojnotehnički glasnik / Military Technical Courier (www.vtg.mod.gov.rs, втг.мо.унр.срб). Ово је чланак отвореног приступа и дистрибуира се у складу са Creative Commons licencom (<http://creativecommons.org/licenses/by/3.0/rs/>).

