



Performance comparison of the most popular relational and non-relational database management systems

Kamil Kolonko

This thesis is submitted to the Faculty of Computing at Blekinge Institute of Technology in partial fulfillment of the requirements for the degree of Master of Science in Software Engineering. The thesis is equivalent to 20 weeks of full time studies.

Contact Information:

Author(s):

Kamil Kolonko

E-mail: kakd15@student.bth.se, kamil.kolonko@outlook.com

University advisor:

Javier González-Huerta

Department of Software Engineering

Faculty of Computing
Blekinge Institute of Technology
SE-371 79 Karlskrona, Sweden

Internet : www.bth.se
Phone : +46 455 38 50 00
Fax : +46 455 38 50 57

ABSTRACT

Context. Database is an essential part of any software product. With an emphasis on application performance, database efficiency becomes one of the key factors to analyze in the process of technology selection. With a development of new data models and storage technologies, the necessity for a comparison between relational and non-relational database engines is especially evident in the software engineering domain.

Objectives. This thesis investigates current knowledge on database performance measurement methods, popularity of relational and non-relational database engines, defines characteristics of databases, approximates their average values and compares the performance of two selected database engines.

Methods. In this study a number of research methods are used, including literature review, a review of Internet sources, and an experiment. Literature datasets used in the research incorporate over 100 sources including IEEE Xplore and ACM Digital Library. YCSB Benchmark was used as a direct performance comparison method in an experiment to compare OracleDB's and MongoDB's performance.

Results. A list of database performance measurement methods has been defined as a result of the literature review. Two most popular database management engines, one relational and one non-relational has been identified. A set of database characteristics and a database performance comparison methodology has been identified. Performance of two selected database engines has been measured and compared.

Conclusions. Performance comparison between two selected database engines indicated superior results for MongoDB under the experimental conditions. This database proved to be more efficient in terms of average operation latency and throughput for each of the measured workloads. OracleDB however, presented stable results in each of the category leaving the final choice of database to the specifics of a software engineering project. Activities required for the definition of database performance comparison methodology proved to be challenging and require study extension.

Keywords: database, performance comparison, MongoDB, OracleDB

CONTENTS

ABSTRACT	I
CONTENTS	II
1 INTRODUCTION	1
1.1 DEFINITIONS	1
1.2 HISTORY	1
1.3 PERFORMANCE.....	2
1.4 TAXONOMY	2
1.5 RELATIONAL VS NON-RELATIONAL	2
2 RELATED WORK	4
3 METHODOLOGY	6
4 DATABASE PERFORMANCE MEASUREMENT METHODS	9
4.1 RESEARCH METHODOLOGY	9
4.2 RESEARCH QUESTION	10
4.3 THE NEED FOR A REVIEW	10
4.4 REVIEW	10
4.4.1 Search methodology.....	10
4.4.2 Selection of articles.....	11
4.4.3 Results.....	12
4.5 CONCLUSIONS.....	14
4.5.1 Method choice.....	14
5 CHOICE OF COMPARED DBMSS.....	16
5.1 CHOICE CRITERIA.....	16
5.2 INITIAL STUDY	17
5.3 RESEARCH METHODOLOGY	17
5.4 RESEARCH QUESTION	18
5.5 RESEARCH	18
5.5.1 Search procedure.....	18
5.5.2 Result	19
5.6 CONCLUSIONS.....	20
6 DATABASE CHARACTERISTICS DEFINITION	22
6.1 DATA SOURCES	24
6.2 DATA ACQUISITION	24
6.3 MEASURED DATABASE.....	26
7 PERFORMANCE COMPARISON	27
7.1 SCHEMA.....	27
7.1.1 Schema translation	27
7.2 MEASUREMENT METHOD	28
7.3 WORKLOAD DEFINITION	29
7.4 HARDWARE OUTLINE	31
8 RESULTS	32
8.1 MONGODB	32
8.2 ORACLE	33
9 ANALYSIS AND DISCUSSION	35
9.1 WORKLOAD A.....	35
9.2 WORKLOAD B.....	38
9.3 WORKLOAD C.....	41

9.4	WORKLOAD D.....	42
9.5	WORKLOAD E.....	45
9.6	WORKLOAD F.....	48
9.7	CROSS-WORKLOAD ANALYSIS.....	52
10	THREATS TO VALIDITY.....	54
10.1	THREATS TO INTERNAL VALIDITY	54
10.2	THREATS TO EXTERNAL VALIDITY	55
10.3	THREATS TO CONSTRUCT VALIDITY	55
11	CONCLUSION AND FUTURE WORK	56
	REFERENCES	58
	ATTACHMENT 1. LIST OF DATABASE SEARCH QUERIES AND RESULTS.....	63

1 INTRODUCTION

Databases and database management systems (DBMS) are nowadays a common element of everyday work, dependent less of the area where they are applied [1]. The extent of database popularity is overwhelming as it is hard to imagine a business domain that does not utilize any of the benefits that come with database application. Proof of database acclaim and adoption can be found not only by browsing basic and complex information systems but also in our private lives in activities that require data management e.g. home budget tracking or cooking recipes administration. Such a widespread utilization of database systems allows for a statement that they are ubiquitous in our lives and surroundings [2].

1.1 Definitions

Studies in the area of databases should be started with a clear identification of all respective definitions. Concepts that are important to determine are “database” and “database management system”. In regards to this matter, a valuable knowledge source is [3] that defines the following:

“Database system is [...] a computerized record-keeping system”

while:

“A database is a collection of persistent data that is used by the application systems of some given enterprise.”

Abovementioned definitions, however short and concise, timelessly identify both terms and a distinction between them. To paraphrase C.J. Date’s words [3], DBMS is a system that defines a schema for persistent data preservation and manipulation. A database is thus an instance of database management system that is filled with records. Database is a set of data collected and managed in a manner defined by DBMS.

1.2 History

Relational Database Management Systems (RDBMS) are now long present in the computer science domain [4]. The history of databases begins with the hierarchical and network database management systems which were created and developed in the late 1960s and early 1970s [5]. An interesting story in the history of software products is how the new relational model, published in 1970 by E.F. Codd, revolutionized the market and many companies [5]. The analysis of abovementioned revolution in information technology and RDBMS adoption is utterly interesting.

In the beginning of database evolution, computer systems did multiprocessing meaning that the tasks were completed in a sequence from start to finish. Later on to improve performance, multiprocessing was introduced where the jobs could share the machine, however each operation was still executed independently and had to wait in queue for available hardware resources [6]. Such system transformed in time into transactional model and formed a well-known client-server architecture we use in SQL databases [6].

Among many qualities of RDBMS we can easily identify: simplicity, robustness, flexibility, and compatibility [4]. Emergence of new generation of applications like Social networking, Business Intelligence and Web 2.0 defined and prioritized new qualities of databases [7]. These are best described in CAP Theorem stated by Prof. Eric Brewer and mention consistency, availability and partition tolerance [8]. Traditional

database systems are not able to serve all the modern applications' needs such as handling large sets of unstructured data or providing scalability [4]. This is where Non-Relational Database Management Systems (NoSQL) comes to rescue [9].

1.3 Performance

As mentioned above, current advancements in technology create an increase in data generation. That creates a demand for fast processing of big data loads which leads to the observation that not only CAP qualities but also performance of database systems becomes a crucial factor for its quality assessment. Keeping the performance of DBMS is a predominant priority of today's data driven organizations [10]. It was estimated that by the year 2017 "...data volumes will grow exponentially, while CPU capacity will increase only geometrically" [11] therefore the need for database performance improvement solutions arises. To satisfy the requirements listed above, software engineering specialists seek for a database system that delivers best performance. The need for application of a most performant database system in computer software is evident. That creates a demand for efficiency comparison between various database management engines. With new technologies emerging in database domain, such as non-relational databases, the necessity to compare those is clear. Below study will focus on the quantitative efficiency measurement of two most popular relational and non-relational database engines which will allow to satisfy the above-described need for database performance comparison.

1.4 Taxonomy

Conducting studies in database management area requires a proper categorization of different DBMSs. A basic DBMS type division could be based on data relation model which groups investigated objects into relational and non-relational sets. Within these two batches, because of its big diversity in data preservation structures and algorithms, taxonomy-wise more interesting is the NoSQL collection. Although there are a couple of approaches on non-relational database engine classification, a standard taxonomy is yet to be discussed by experts [12]. Documents [13, 14, 15] suggest a categorization of NoSQL databases based on similarities in data model. These classifications slightly differ between each other nonetheless a big picture can be outlined and few groups defined i.e. object, key-value, document, column and graph databases. Such defined taxonomy will be used in the following research.

1.5 Relational vs non-relational

Relational and non-relational database management systems are designed with different qualities in mind. Relational databases can be characterized by an ACID principle [16] where Atomicity, Consistency, Isolation and Durability are the qualities that should be implemented by any database engine of the family. On the other side of the domain, there are non-relational databases with BASE [17] semantics as a baseline that describes the characteristics of the family representatives. Qualities such as Availability, Soft State and Eventual Consistency are the key aspects of NoSQL representatives. In addition, non-relational databases' values are defined in a CAP theorem [18] which is closely related to the BASE rule mentioned above. These values are Consistency, Availability and Partition Tolerance.

An interesting observation is that none of the principles listed above mentions performance quality. This is because the rules above apply to a database model and functionality while performance is a result of a specific implementation of these.

Performance comparison of relational and non-relational database representatives is therefore sensible and seems justified. Additional arguments that prove the reasoning behind comparison of above-mentioned families are common use cases where the application of either of database engines is possible and connected with various benefits. One example use case can be a large retail website where multiple database qualities are factors driving for the engine choice. Examples are atomicity – justified by the need to store orders as atomic values in order to prevent multiple charges for the same order, availability – as costs of periods when website is non-operational can be critical or performance – when time required to display an offering description is critical for customer satisfaction. Such defined requirements can be satisfied by application of both, relational and non-relational databases while performance might be considered as a factor that colors the decision. A multitude of possible use cases where application of relational and non-relational database systems is equally considered is overwhelming and, as such, proves the concept of comparisons between these families.

2 RELATED WORK

A brief literature review on related work in the database area resulted in identification of several studies that undertook a topic of database performance. BTH library Summon@BTH search system [19] and Google Scholar engine [20] were used in order to determine research that raised a subject of database efficiency comparison. Papers [21], [22], [23], [24], [25], [26], [27], [28], [29], [30], [31], [32] and [33] all make an attempt to compare database performance. There are however some drawbacks in each of the studies that lower value of the research.

Study [21] compares Cassandra, Hive and MySQL databases. Each database is shortly described although a reason for such a database choice is not justified. Paper also lacks a precise description of data and data selection method that was inserted into each evaluated object. Arguments on the choice of performance comparison method are also lacking. Research [22] takes Cassandra, HBase, Yahoo!'s PNUTS, and a simple MySQL database implementations under examination. Reasons for such dataset choice and description of inserted data are not mentioned. Database management engines such as HBase, Cassandra and MySQL are compared in [23]. An immediately observed drawback of this research is the imprecise description of rationale behind the choice of compared objects. Authors only vaguely state that "larger use at the present moment" was a factor that allowed for restriction of compared systems and that thought is not extended anywhere later nor proved by any source. Performance measurements in that research are taken from [22] thus faults of that research apply likewise. Hive's, MongoDB's and SQL Server PDW's performance was measured in [24]. Authors are aware of study drawbacks as they reference lack of justification for choices of measurement method and studied database systems as a possible threats. Furthermore data and data structure of investigated objects are not described. Another performance measurements are performed in [27] where MongoDB and MySQL databases are investigated. This study however omits the area of measurement method choice and does not describe database contents during experiment. Research [28] takes on comparison of MongoDB, RavenDB, CouchDB, Cassandra, Hypertable, CouchBase and MS SQL Express. The paper unfortunately lacks description of the process in some areas such as identification of a way to conduct measurements, justification of database choice and description of database data model. In [29] MongoDB and Oracle DB are investigated. Again, the study does not identify a procedure of choosing a performance measurement method. Data inserted into examined objects is vaguely characterized as its structure is not based on any previous research. Studies [31] and [32] both compare performance of MongoDB and PostgreSQL. Some drawbacks can be identified in both research first of which could be addressing only specific domain areas (sensor data in [31] and Web Map Service in [32]). Data measured in databases is specific to one field and not described precisely enough to prove that it represents a common example. Furthermore [32] does not justify choices for measurement method and compared objects. An approach to calculate efficiency in [31] is not mentioned. Another identified paper in this research area is [33] that compares MongoDB and SQL Server Express. A deficiency of aforementioned study is again lack of method description and database contents justification.

Furthermore, studies that compare performance of only non-relational database systems were identified. Research [25] contributes to CouchDB and RavenDB testing. The methods used for efficiency comparison and database choice are not mentioned in the study. Additionally, only general characteristic of data size is mentioned in the description of verified dataset and the choice of such values is not justified. Comparison of only NoSQL engines is also present in [26], [30], [34], [35] and [36] while as such it does not correspond directly to this thesis topic.

Among the reports, there are numerous cases of neglecting proper descriptions of used measurement method as well as database contents during the measurement execution. Studies are hardly repeatable and results are often hard or impossible to verify. Some additional research on database comparison is also worth mentioning, ([6], [24], [25], [26], [27], [28], [29]) although it does not incorporate quantitative performance measurements. None of the aforementioned published studies, investigates the topic of defining characteristics of the database that is being measured. That identifies a research gap in the area. Although big effort has been put on multiple database management systems performance comparison, there is no example of attempts to measure an average case implementation. This paper is to fill the discovered research gap.

3 METHODOLOGY

The aim of this research is to measure and compare performance of two database management systems. Two representatives will be chosen, each from different database family (relational and non-relational) and compared. Big scope of research imposes a requirement to divide the study into several sections. A factor that allows for an identification of research steps could be definition of research questions. List of research questions that will be addressed in this thesis is presented below.

RQ 1. What are the reported methods of measuring database performance?

This question is aimed to identify methods for measuring DBMS performance reported in published scientific sources. Based on the results, proper method will be chosen for future use in this research.

RQ 2. What are the two most popular non-relational and relational database management systems?

This question is stated to determine the popularities of non-relational and relational database management systems in sources published on the Internet. The intermediate answer to this question is expected to deliver a list of NoSQL and relational database management systems and a number of their references in Internet search engines. The final solution for the stated query will be the names of two DBMS's (one relational and one non-relational) with the highest number of identified references.

RQ 3. What are the characteristics of an average database and their respective values?

The answer to this question provides a list of characteristics of a database instance and their average values. This defines both, properties and values that a "standard" database represents.

RQ 4. What is the performance comparison of the most popular relational and non-relational database management systems in an average case implementation?

The answer to this question provides a quantitative measure of selected databases performance in specified average case implementation. Conducted comparison relates data between two measured objects.

Clear distinction of research stages can be observed in Figure 1. Such a research plan with a distinction of several different phases requires definition of multiple study methodologies. Each research question needs to be addressed thus a research method should be identified for every candidate.

Taken into consideration that RQ 1 is aimed to identify a list of database performance measurement methods, a natural conclusion is that only methods, validated in research should be included in the study. Therefore, undertaking a literature review (LR) becomes an intuitive choice for a research method. LR is an approach that applies to paper study and provides a set of guidelines that can improve the overall quality of research.

For similar reasons, a review will also be performed to answer RQ 2. The method will be applied to Internet sources. The choice of DBMS's compared in this paper will be based on popularity of each system on the Internet. For this sake, an Internet search engine will be used. An example could be Google [37]. According to [38], Google is

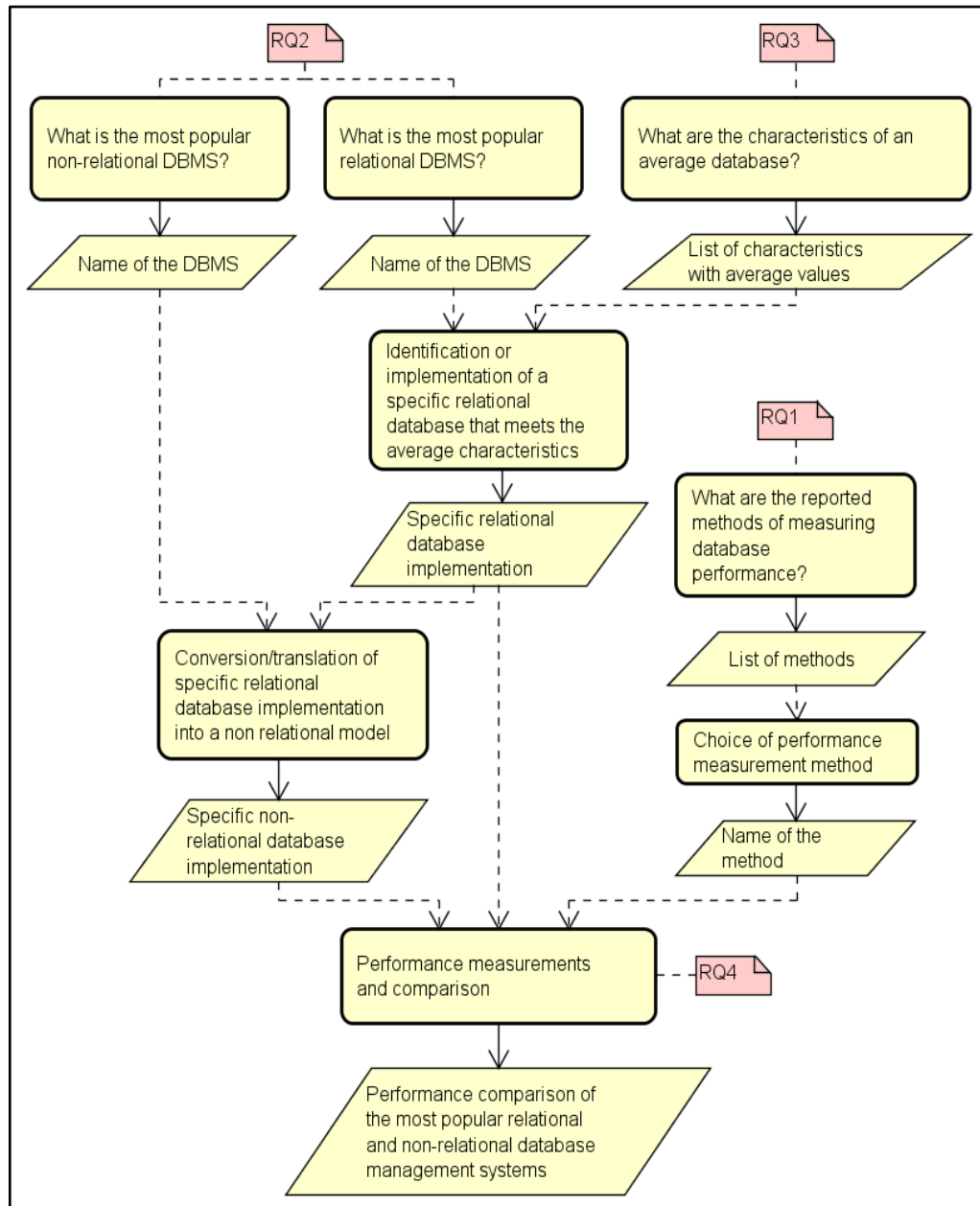

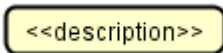
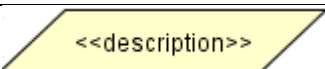


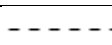


Figure 1 Overview of research process

Table 1 Figure 1 legend

	Note		Process
	Data		outputs
	inputs		relates to

currently the biggest search engine in the world with over 66% market share. It also supports Boolean commands “AND”, “OR” and additional search operators which are necessary for scientific queries [39]. Other research methods were also considered to answer RQ 2. Using published scientific research as a source of information on DBMS popularity was rejected due to the fact that popularity of DBMS as a research object

might not indicate the real state of the market. Performing a survey in the business area, however interesting and possibly solid, was also excluded as a research method as it required an overwhelming amount of responses to state grounds for further research. Internet search engines data is indexed in all possible domains, including scientific and business area, therefore, results obtained that way, serve as a good measure for DBMS popularity.

RQ 3 aims to identify the characteristics of an average database and their values. The question will be responded with two research methods. The first step, identification of database model characteristics, will be answered in a consultancy with database experts from author's home university and with study of Internet sources. Identification of the average values for defined characteristics will be performed with the usage of Internet open dataset indexes e.g. Open Government Data Catalog [40] or Data Portals [41]. An automated script will be developed to gather the metadata for available datasets and the average value for each characteristic will be calculated.

To measure performance of the most popular relational and non-relational database management systems, and thereby answer RQ 4, an experiment will be performed. Two database instances, one representing relational DBMS and the second, demonstrating NoSQL, will be compared in terms of efficiency. Both implementations will represent the same characteristics of the average database. A measurement will be performed, using one of the methods, identified in RQ 1.

Additional research methods were considered to be used in the study. Interviews and conducting a survey were examined as possible approaches to answer RQ 1, RQ 2 and RQ 3 however they were ruled out due to various limitations. Both methods require collecting information from respondents although with a limited time that was planned for master's thesis course, the number of collectable responses could be a threat to research validity. With potentially low amount of collected feedback the possibility of missing identification of database performance measurement methods, in case of RQ 1, or database characteristics, in case of RQ 3, was high. Additional drawback of reasoning over data collected by surveys or interviews is the high amount of responses that is necessary to form a statistically significant sample which is an enormous effort with resources limited in university course. With the consideration of outlined methods' limitations the selection of the study approaches that will be used in this research is described in the paragraphs above.

4 DATABASE PERFORMANCE MEASUREMENT METHODS

An important activity in database performance comparison is the identification of method used to measure systems' efficiency. Developing a procedure that will allow for identification of quantitative assessment of chosen database performance is crucial for study correctness. Uniform measurement methodology, in case if necessary, will enable for multiple study repetitions and at the same time will prove reliability of received outcomes and open an area for future analysis.

To achieve the described goal a study will be undertaken that will result in a list of methods that allow database performance assessment. Interpretation of received outcomes will enable for a choice of a method that is best suited for the aim of this thesis i.e. Performance comparison of the most popular relational and non-relational database management systems.

4.1 Research methodology

There are numerous methods for gathering research data from existing internet, library or expert sources. These methods can vary within the steps taken to acquire information and therefore lead to completely different extraction results. In below study, when the aim of the research is to get performance measurement methods, the choice of research methodology is limited to a few possibilities.

First of the considered methods is the analysis of business domain, particularly performing a survey in the area of practical database system usage. Such an activity is problematic because of a need to gather a vast number of responses from multiple and diverse, possibly international, environments. Unquestionable advantage of surveying is the delivery of a list of database measurement methods that are accepted and used in practice thus their reliability is proved.

Another idea of a research technique is a literature review in a domain of interest. Access to databases that gather published scientific literature grants a possibility to analyze all available sources and collect necessary information. One recognized and widely researched approach to such data mining activity is Literature Review (SLR). Literature review is a means of evaluating and interpreting available research relevant to a particular research question, topic area, or phenomenon of interest. Literature reviews aim to present a fair evaluation of a research topic by using a trustworthy, rigorous, and auditable methodology. Such described advantages of literature reviews present the method as one of the most valuable tools for research review. It is also evident that if applying a literature review to below study will result in database performance measurement methods that are scientifically documented and proven techniques thus their correctness is guaranteed.

To perform below study, a literature review was used. Such a choice of a research technique was justified by a guarantee of result correctness and possibility to reproduce the research if required in the future with similar results. Selection of literature review over surveying is rationalized by a huge availability of research in database domain and limited possibility of performing survey that would result in measurable and representative data. Time and resources required for such an activity are not sufficient in a thesis project.

Researching a domain of a study is a crucial activity in any expert academic work. This chapter is a review of the advancements in relational and non-relational database performance measurement methods. The study presented here intends to identify reported ways of qualifying the database efficiency and is an initial work essential for further analysis of performance of relational and non-relational databases.

4.2 Research question

An activity which is crucial for proper literature review is stating the research question. A source that helps in that activity is the definition of Systematic Literature Review and five exemplary types of questions that can be found there, with “Assessing the effect of a software engineering technology” as one of the examples [42]:

Based on the abovementioned, the research question is stated as follows:

RQ1. *What are the reported database performance measurement methods published in scientific sources?*

Such a practice leads to the increased confidence in research question statement and therefore the study will be continued with this particular inquiry in mind.

4.3 The need for a review

A brief research on the database performance measurement methods didn't identify any articles answering similar research questions, therefore a literature review will be performed. Chapter 1.3 described the importance of performance evaluation on database engine choice. As the usage of database technologies in software projects nowadays is evident and their popularity increases, new technologies such as non-relational databases emerge [9]. The need for the up-to-date review in performance measurement methods is thus clear.

4.4 Review

4.4.1 Search methodology

Essential step in review planning is development of the review protocol. A review protocol specifies the methods that will be used to undertake a specific literature review. Early definition of the protocol is necessary to reduce the possibility of researcher bias. The components of a protocol consist of all elements of the literature review with extra planning information. Methodology of the search consists of the strategy that will be used to search for primary studies including search terms and resources to be searched, resources include databases, specific journals, and conference proceedings.

To maximize the review efficiency, multiple data sources have been selected for the search. In the following literature review, over 110 databases have been searched with the use of Summon@BTH search engine available at Blekinge Institute of Technology Library [19].

Proper search requires detailed specification of query terms. As [42] suggests, most common approach in defining the search strategy is extraction of the search keywords from the research question. Thesaurus and abbreviation dictionary might be of help. Combining the keywords and using Boolean AND's and OR's creates a sophisticated search strings. Based on the research questions, the following terms has been identified:

- Database, performance, efficiency, benchmark, measure, evaluation, assessment, method, analysis, manner, approach, technique, way, pattern

In result, the following search string has been defined:

- database AND (performance OR efficiency OR benchmark) AND (measure OR evaluation OR assessment OR analysis) AND (method OR manner OR approach OR technique OR way OR pattern)

Executing above query resulted in around 2 900 000 results. The number of results had to be therefore limited as the limited time in the course did not enable full and professional research. For these means, the data inclusion and exclusion criteria have been specified. Next, the strategy has been designed and the steps undertaken to identify most relevant studies were defined as follows:

1. Executing search query
2. Applying inclusion/exclusion criteria
3. Sorting the results by relevance to the query
4. Manual selection of articles from first 20 results

The inclusion and exclusion criteria are listed below:

- Publication date > 1986
- Discipline = computer science OR engineering OR sciences
- Study type = article OR book chapter OR patent OR conference proceedings OR bachelor thesis OR master thesis
- Language = English OR Polish
- Full Text Online available

4.4.2 Selection of articles

Analysis of query results led to identification of most significant articles. Narrowing down the number of discovered studies was executed with the help of search engine sorting functionality. The “sort by relevance” criterion was used to arrange the results and top twenty studies were investigated in the further research. Summon@BTH engine uses the relevance ranking system developed by ProQuest. This system utilizes a technology that allows to store a single index of the data collected from all supported research publication services in order to apply the algorithms equally across the entire body of indexed content, and return the results in a single, unbiased, relevance-ranked results list [43]. The Summon service relevance ranking consists of three criteria sets that allow to score indexed items. These are Dynamic Rank, Static Rank and Recommendations which group the total of 19 distinct industry standard concepts for relevance rating. Some of the examples of criteria applied in the above-mentioned functionality are query term proximity, term stemming, synonyms, scholarly/peer reviews, citation count and related search suggestions with the full list available at [43]. Each of the 20 results from the query obtained by application of the above-mentioned functionality, was browsed, read and analyzed, what resulted in identification of 8 reports for further data extraction. Data about author, date of publication, title and article type has been registered for each study and presented in Table 2.

Table 2 List of articles selected for data extraction

ID	Authors	Date	Title	Article Type
S1 [30]	Gandini A., Gribaudo M., Knottenbelt W., Osman R., Piazzolla P.	2014	Performance Evaluation of NoSQL Databases	Book Chapter
S2 [25]	Nepaliya P., Gupta P.	2015	Performance Analysis of NoSQL Databases	Journal Article

S3 [44]	Bornhoevd C., Guerrero P.	2005	Systems and methods for repeatable database performance testing	US Patent
S4 [45]	Shee D.	2002	Database performance monitoring method and tool	US Patent
S5 [46]	Teorey T.J.	1998	Dependability and performance measures for the database practitioner	Journal Article
S6 [47]	Gray J.	1987	A view of database system performance measures	Conference Proceedings
S7 [26]	Henricsson R.	2011	Document Oriented NoSQL Databases: A comparison of performance in MongoDB and CouchDB using a Python interface	Bachelor Thesis
S8 [48]	Dwivedi A. K., Lamba C.S., Shukla S	2012	Performance Analysis of Column Oriented Database Vs Row Oriented Database	Journal Article

4.4.3 Results

The steps taken in data extraction process were based on Kitchenham's description in [1]. For each study selected, the analysis of contents have been performed and the following data has been listed:

- The source
- Identified database performance measurement methods
- Tools – software used to perform measurement
- Description of identified methods

Table 3 Database performance measurement methods

Src	Measurement method	Tool	Description
S1	Latency (microseconds per operation), Throughput (operations per second)	Yahoo! Cloud Serving Benchmark (YCSB)	Random data is loaded into the database. YCSB executes randomly mixed requests with 50% reads and 50% writes for each client database. The number of querying threads is varying. Latency and throughput are measured for each execution.
S2	Latency (microseconds per query)		Measured time for query execution on various JSON files of different size (20k-75k lines).
S3	Latency or throughput		Running performance critical database transaction statements in the application corresponding to one or more tables and separately executing each performance critical database transaction statement against the database a predetermined number of times, Wherein the database is in a predetermined initial state at the beginning of each execution.
S4	Latency (per SQL statement)		Monitoring the process performance of a database that accepts and records SQL

			statements and that records the status of a session of use of the database. The method obtains the SQL address and hash value for each SQL statement, the current session status corresponding to each SQL statement and the previous session status corresponding to each SQL statement. The method also records a time stamp at the time the session status information is obtained. The information gathering steps are repeated at a predetermined interval of time. Using the information gathered, the run time for each SQL statement is calculated.
S5	Mean transaction time (milliseconds per operation)		The mean transaction time in the system is a function of the mean service delay time for the transaction over all components, plus restart delays, plus queuing delays for contention. Once the mean service delay time is known, we can then compute the mean completion time, taking into account the restart delays. Finally, we compute the mean transaction time in the system from known queuing formulas.
S6	Wisconsin benchmark (latency, throughput)		A set of 32 retrieval and update statements and a script for multi-user tests. They give two performance metrics: the elapsed time for each statement and the throughput of the system when running sixteen simultaneous scripts. No response time requirement or cost measure is included in the definition.
	Bitton and DeWitt (SCAN – latency, DebitCredit – transaction-per-second, price per transaction)		SCAN - A mini-batch operation to sequentially copy 1000 records. SORT - A batch operation to sort one million records. DebitCredit - A short transaction with terminal input and output via X.25, presentation services, and a mix of five database accesses.
S7	Latency (milliseconds per operation)	Linux time	A Python interface for selected databases has been used, and the time for defined transactions has been measured using the Linux Bash utility “time”.
S8	Latency (microseconds per operation)	MATLAB 7.6.0	A MATLAB interface has been designed to run defined queries and measure the time of each task execution.

As data extraction showed, there are multiple methods to assess database performance used in the domain. As can be observed, there are three basic scales for measuring this quality which are: time – latency for transactions or single operations, throughput – number of operations done in a given period of time, cost – expressing the overall price per transaction defined as money spent on database management for the time when transaction is performed. Such measures may seem similar at least throughout one given scale, however there are big differences between single methods.

An example could be the difference between a method defined in [30] that randomizes the queries and communicates with the database through Java interface and Wisconsin benchmark presented in study [47] that runs the test on a set of predefined queries. A decision on which method is the best is subject for another literature review or should be decided individually.

4.5 Conclusions

Comparing various database systems in terms of performance requires a lot of effort and initial research. One question that arises before the study can be started is the right choice of the method to assess database efficiency. This literature review was to list possible ways of database performance measurement. Definition of search methodology allowed for a clear identification of steps in the process. Proper statement of research question indicated phrases connected to the field of interest and enabled the creation of search queries. Additional inclusion and exclusion criteria as well as quality assessment, limited the number of potential studies for further analysis. Eight articles have been selected for data extraction. During the process, nine different database performance measurement methods have been discovered. As there are some clear similarities between these methods, each one is unique.

4.5.1 Method choice

To analyze performance of database systems it is necessary to identify one method that will be used to collect necessary measurements. Abovementioned research delivered a list of candidates from which a procedure that is best suited for below thesis must be identified. To achieve such a goal it is necessary to define criteria that will allow for a clear selection of a method that satisfies all requirements. Factors that will affect further analysis are outlined as follows:

- Precision – a precise measurement of database performance is a key factor. Inaccurate results could affect the quality of research and potentially lower its value.
- Cost – this research is carried out as a part of Master’s thesis program thus no budget exists that could cover the expenses connected with a choice of database performance measurement method. Chosen technique should be free to use and should not have any license limitations that could disallow or pose a threat to future research usage or extension.
- Availability of the source code– usage of any kind of software tool to measure database performance is necessary. A scenario when such a tool needs to be modified or adjusted to work with multiple database systems is probable and should be taken into account. Example software change that could require implementation is improvement of measurement precision. That creates a requirement for source code availability.
- Metric diversity – a choice of a method that allows for performance measurement based on multiple metrics is desired. Ideally, more aspects taken into consideration translates into more profound method.

Based on the criteria defined above, the Yahoo Cloud Serving Benchmark was selected as a method to be used in the future performance comparison in this research. This method satisfies all of the requirements listed above by offering performance measurements up to a microsecond precision and multitude of metrics with examples of throughput and latency. Furthermore, the authors of the research that describes the method developed a tool named YCSB Client that automates the process of measurements and is available online for free and with open sourced codebase that

allows for any necessary modifications. YCSB is commonly used in related research works in the area of database performance measurements, examples of which are [21], [22], [23], [34], [35] or [36] discussed in related work section 2.

5 CHOICE OF COMPARED DBMSs

An activity essential in comparing databases' performance is undoubtedly a proper choice of investigated objects. Inaccurate and erroneous choice of Database Management Systems might lead to a judgmental assessment and invalid comparison results and in consequence result in a low value research. Therefore, to minimize risk of low quality study it is necessary to precisely identify a procedure of choosing investigated DBMS and to perform an analysis that will result in a set of candidates for future experiment.

5.1 Choice criteria

A choice of Database Management Systems used in this research depends on a number of factors. The most important of these factors is a statement of a set of criteria that will be a basis for potential candidate analysis. A good example of such criteria is the popularity in Internet sources and popularity amongst users in professional domain. A principle of popularity amongst users has a clear advantage here above popularity in the Internet as it describes current and exact state of business domain. In other words, there is a risk that discussions and database acknowledgement on the Internet does not describe the real state of the market. Popularity of DBMSs is a basis for narrowing down the number of investigated objects as it might be perceived that more intensively discussed and utilized objects tend to present some advantages over competitors creating a connection between popularity and overall quality. Popularity of a given DBMS must have a reason in its characteristics or environment. The term popularity is considered in the further research in a cumulative, historical and present context. Above reasoning justifies a choice of this criteria for selection of a proper DBMS for this study.

Examples of another DBMS choice conditions might also be: popularity in published literature, reference in published scientific sources that take on the topic of database performance and popularity as wells as business criteria such as price in commercial use, documentation availability etc. As mentioned above, criteria of popularity in Internet sources and popularity amongst users have an advantage over other conditions as it might be observed that they partially or entirely contain other suggested criteria. An example could be DBMS's price in commercial use that most probably closely correlates with its popularity [49] thus is already included in this criteria.

It is worth noting that the difference between DBMS popularity in Internet and popularity in business domain is substantial. Especially in case of the Internet sources popularity describes the number of references of a certain phrase dropping the context which might be relevant. Investigating whether given database name was used in a positive or negative context is rather difficult if not impossible. However, it should be underlined that popularity counts for both, beneficial and unfavorable mentions. Investigating popularity of DBMSs in business domain would neutralize the problem of context although such an activity is complex to perform given the necessity to examine a big and diverse research group.

Aforementioned analysis proves a point of choosing popularity as a driving factor for identifying databases used in this study.

5.2 Initial study

A brief review was performed in order to verify publicly available data sources that take on the subject of databases' popularity. This activity was performed manually using Google search engine [37]. Query used in the search was defined as "database popularity" and resulted in over 20 million results. The list of outcomes was sorted by relevance criteria using the default Google's search filter and limited to first 10 picks due to limited time for results investigation. The returned list included [50], [51] and [52] as the most interesting data sources. A comprehensive sorted list of DBMSs was discovered under DB-Engines [50] that describes itself as a Knowledge Base of Relational and NoSQL Database Management Systems. DB-Engines measures popularity with multiple factors in mind a list of which can be found at [53] and is shortly listed below:

- Number of mentions of the system on websites
- General interest in the system
- Frequency of technical discussions about the system
- Number of job offers, in which the system is mentioned
- Number of profiles in professional networks, in which the system is mentioned.
- Relevance in social networks

Top ten results acquired by DB-Engines and sorted by popularity are listed in Table 4

Table 4 List of ten most popular DBMSs by DB-Engines

Rank	DBMS name	DBMS type
1	Oracle	Relational DBMS
2	MySQL	Relational DBMS
3	Microsoft SQL	Relational DBMS
4	PostgreSQL	Relational DBMS
5	MongoDB	Document store
6	DB2	Relational DBMS
7	Cassandra	Wide column store
8	Microsoft Access	Relational DBMS
9	SQLite	Relational DBMS
10	Redis	Key-value store

Most popular databases identified by sources [51] and [52] are in contrast to the results presented above. Article [51] however defines the data based on a single survey performed amongst www.stackoverflow.com website users while [52] does not publish any information on statistics' source. The discrepancy between the database popularity analyses in above sources is a factor that confirms the necessity for an additional study in this domain.

5.3 Research methodology

In order to continue analysis on choosing investigated database systems it is necessary to define a research methodology that will outline procedure for further study. Early definition of research protocol is crucial for limiting possible author bias. An appropriate approach, similarly as when researching proper methodology for database performance measurement, seems to be following the guidelines of Kitchenham's Systematic Literature Review [42]. In the following case it is however necessary to adjust some of the described practices to more closely match and adopt these methods to a research in Internet, rather than literature, sources. Thereby this study will be performed with the following steps, defined below.

1. Definition of research question.
2. Definition of data sources used in the review.
3. Definition of search queries and data filters.
4. Analysis of acquired review results.

A clear difference between above definition of research process and guidelines defined in [42] is a lack of definition of data extraction from gathered results. This step is not required in below study and will be omitted in further research. In case of an analysis of data from Internet sources and in particular in an analysis of query popularity, individual investigation and breakdown of results is not necessary. Searched information is already represented in a fact that a given query returns any results.

Browsing Internet in search of an object's popularity might lead to different results over time as multiple factors influence potential number of returned outcomes. An example could be a change in a chosen search engine's implementation which might impact quantity of indexed websites. Another illustration of a problem could be modification of a search algorithm. That identifies a minimal risk which should be addressed when discussing research methodology. It must however be pointed out that if any of previously listed examples occurs, meaning modification of search engine algorithm or its implementation, it is generally intended to improve accuracy of presented results. Therefore a response to such a jeopardy is not necessary and this risk will be ignored. A problem of slightly higher importance and impact on research value might however be an activity of proper research question statement. As author's intention is to achieve most precise results, some additional study should be involved to verify outcomes obtained in abovementioned review. This step will be performed in a manner of additional brief informal literature study on the topic of database systems' popularity. Internet and literature sources will be queried against research already performed in this area and results will be merged.

5.4 Research question

Choice of literature review as a research method requires a definition of a proper research question. As the aim of this study is to identify popularity of Database Management Systems, research questions was defined as follows:

What is the popularity of chosen Database Management Systems in Internet Sources?

An outcome expected as of answering above query is a list of DBMS engines with its popularity expressed as a number of results per query. That will allow for easy list sorting and filtering out two most popular objects, one relational and one non-relational, database systems.

5.5 Research

5.5.1 Search procedure

As it was mentioned in the proceeding research, where database performance measurement method was evaluated, proper definition of search protocol largely contributes to research quality. That leads to an observation that a strategy should be defined to maximize process effectiveness.

Important part of protocol statement is the choice of utilized search engine. It is natural that in case of querying Internet sources the aim should be to maximally widen the search data set. An engine that indexes possibly highest number of websites should

be thus used in further research. Study [54] describes changes in search engine sizes over nine years up to 2015. Authors did not observe a constant variation tendency in the number of indexed sources of each investigated engine. Although the number of results per investigated query differs throughout years it is clear that Google search engine [37] has a clear advantage over competitors with highest number of returned pages. For such a reason, this engine provider was chosen as a tool for this study. Additionally, Alexa Internet [55], one of the biggest Internet analytics companies, lists Google as a most popular search engine up to date [56] with over 80% global market share [57]. Engine supports queries with “AND” and “OR” operators which simplifies further research.

Correct search activity requires statement of phrases that must be included in search queries. Systematic Literature Review practice suggests here extraction of keywords from the research question nonetheless such an action would not produce relevant results in current example. That identifies a need for an additional short study that will allow for recognition of names required for further research steps. Helpful here are studies that were already performed in this area that group lists of names of relational and non-relational database management systems. One example could be [14] that was mentioned earlier when discussing NoSQL taxonomy. Additionally in between sources that list database names, [58], [59] and [60] should be mentioned. Above research allow for enumeration of over 300 DBMS that will be used in this study.

Identification of DBMS names which popularity will be measured allows for statement of search phrases for chosen search engine. It is necessary to pinpoint some of the predicted problems that, if not addressed, might cause improper measurements.

1. Ambiguity of DBMS names e.g. U2 is used both as a name for popular music band and a database system. Sophia is a female name and at the same time a label of non-relational DBMS.

A solution for such a defined problem is construction of search phrases with a use of quotation mark and “database” word for all investigated objects. Google engine offers a mechanism that allows for a phrase search excluding analysis of its permutations i.e. an exemplary search for “Sophia database” (without quotes) will result in a combination of searches for “Sophia”, “database” and other permutations of this phrase. That is obviously incorrect and misleading. Surrounding search query with quotations returns websites that contain exact usage of searched phrase, in this example only text that contains “Sophia database” will be listed with results.

- 1.1. Only DBMS names that does not contain “database” keyword already included in the name will be extended with such.
2. Multiplicity of names for a single DBMS e.g. Apache Flink database system was named Stratosphere in legacy versions. Both keywords point to the same engine thus a search will be performed for all of them and search results will be summed.
3. Usage of acronyms in DBMS names e.g. “EventSourcing for Java” database is also often addressed as “es4j”. A solution similar to the one described in point 2 will be applied here.

Full list of searched phrases can be found in Attachment 1.

5.5.2 Result

Execution of the study required running 310 queries in Google search engine. The number of queries is a result of collecting all non-relational and relational database names and resolving conflicting situations listed above. The activity of executing the queries was performed by implementation of a simple script in Python language utilizing

“google 1.9.1” library [61]. Source code of the developed script can be found in [62]. Twenty queries that resulted in highest number of results obtained by executing above script are presented in Table 5. All performed queries are listed in Attachment 1. Relational database engines are marked with gray background.

Table 5 List of searched phrases with number of results per phrase.

Query	Number of results
Oracle database	9430000
MySQL database	643000
SQL Server database	535000
Microsoft Access database	480000
SQLite database	459000
PostgreSQL database	452000
MongoDB database	167000
IBM DB2 database	156000
Teradata database	140000
H2 database	127000
SAP HANA database	115000
Greenplum database	110000
Informix database	97800
Firebird database	94500
SQL Azure database	74800
Oracle NOSQL Database	67500
FileMaker Pro database	60700
MariaDB database	57700
Cassandra database	54700
mSQL database	48200

5.6 Conclusions

As it might be observed, search results presented in Table 5 have high discrepancy. Some of the investigated phrases did not return any results while the most popular one had over nine million mentions. Lack of results does not however mean that a given DBMS is not used or discussed in business domain. As it was mentioned above, search phrases were constructed by merging a name of a given DBMS and “database” keyword to identify exact occurrences of such permutation. Absence of search results in such example only means that no given occurrence is indexed by chosen search engine. Nevertheless, such a phenomenon brings a substantial information. For all identified relational and non-relational database management systems same search procedure was applied. Extracted results present a clear difference in chosen database engines’ popularity thus a choice of objects used in further research is possible.

Above research resulted in a list of DBMS names and their popularity in Internet sources. Sorting all obtained data based on number of query results allows for choosing objects that will be used in performance measurement research. The aim of this thesis is to compare performance of two databases, single representatives of relational and non-relational types. As it is presented in Table 5, most popular DBMSs of each kind are

respectively: Oracle as a relational DBMS representative and MongoDB, which represents NoSQL family.

Additional literature review was performed to validate results obtained from original study. A source that independently investigates DBMSs' popularity was identified and results were displayed in Table 4. Two of the most popular database systems that fulfil requirements of this thesis are Oracle and MongoDB. Such a choice exactly matches results from above study thus no specification of merging procedure is necessary. This confirms the results that were obtained in the above research. Furthermore, the analysis of studies that undertake the area of database performance comparison, which were identified in the related work, was performed. MongoDB was identified as the database system that was frequently included in referenced studies, in particular [24], [26], [27], [28], [29], [31], [32] or [33]. Additionally OracleDB's performance was compared against NoSQL for example in research [29]. Summary result of above literature and Internet source analysis proves the selection of Oracle RDBMS and MongoDB for the further research.

6 DATABASE CHARACTERISTICS DEFINITION

The aim of this research is to compare performance of two chosen database management systems. In order to complete such an operation, specific instances of database engines must be created and filled with data. Differences in data models and structures might affect performance measurements and therefore cause research bias. It is easy to imagine that a database filled with several data structures with certain size will perform different than a system of multiple small data sets. Study [63] clearly describes relation between database file size, expressed in data compression, and its performance. Partitioning of data and its influence on systems efficiency is also described in [64] where author suggests measures that should be taken to optimize database performance. It is thus clear that data structure and data entered into database itself should be investigated and precisely described.

In an activity of data description, various approaches might be taken. The most important question that should be asked here is what kind of data shall be described and what are the characteristics of this data. It is necessary to understand differences in validity of various data models as it might represent a threat to the legitimacy of the research. Additional is the impact caused by a choice of database domain as it is clear that attributes of data differ across disciplines. It would be desired to query all possible areas to get most accurate characteristics and value definition. Complete study of all database usage domains is however impossible due to obvious reasons of data unavailability. As much as it is desired to cover possibly highest number of sources it must be limited to several areas and possibly extended in future. It is predictable that only several database domains will be reachable and available for analysis in this study while it is easy to broaden the set of available sources if any new is available.

Variety of data structure models is one of the numerous differences between relational and non-relational database management. Diversity of these is best visible in NoSQL engine family where the same data can be stored using multiple methods. Clear examples might be column-data and key-value stores that are designed for different usage thus store data with distinct models. Same information there can be preserved in many ways. Such a phenomenon carries significant effects as not only the data model across database engines differs but also characteristics for these patterns are diverse. A task of mapping relational and various non-relational engine data models is very complex if not impossible. That results in an inability to precisely and meticulously describe attributes of data stored with distinct models. Therefore this research will focus on description of data characteristics stored in relational DBMSs when an extension of such is possible and subject to future research.

Natural approach for data description is collection of its characteristics and values. It would be desired to list all possible attributes that could describe a data set and then perform appropriate measurements for their values. The searched attributes are in fact database metadata, the description of data stored in database. As [65] describes, there are numerous approaches to metadata preservation and modelling. Each DBMS defines its own metadata schema and access methods. Lists of stored metadata attributes are usually present in system documentation. Several examples of metadata description for most popular relational database management engines can be found at [66] for Oracle DB, [67] for MySQL DB, [68] and its subpages for Microsoft SQL Server. It is easily observable that abovementioned lists of attributes mostly differ between each other. A common set of characteristics is hard to acquire however some general features can be easily found. Some of these that will be used in the future study are defined and described below.

Data size – referenced in study [21]

One of the most obvious common characteristics for all databases is the size of all information that they preserve. Estimation of this value is quite easy with a few observations in mind. All database engines, regardless of schema type, must store data in the manner defined by computer filesystem. In all modern operating systems information is stored in structures called “files”. It often happens that one application and its data is preserved in multiple file instances, which is also applicable to DBMSs. That leads to an observation that database size could be calculated by gathering all files that are used to make it function. However, it must be pointed that it is possible that such files might contain additional information over raw preserved data thus a distinction between database engine size and size of data inserted into the system must be made. “Data size” characteristic should represent only the size of data inserted into the system by the user while database engine size is the volume of the engine itself with no user-inserted data inside. If a situation happens that DBMS engine files and actual data files cannot be separated, a method for inserted file extraction must be individually investigated. With above remark in mind it seems that calculation of database size from filesystem size seems reasonable and could be used in this research.

Number of data tables – referenced in studies [27], [29]

Relational databases store data in a schema that is constructed from tables and relations between them. An activity of database design requires a good knowledge of relational model and often results in a definition of multiple tables for a single problem solution. Number of tables stored in a database is a good characteristic that describes the dataset. Although it has been decided that only attributes of relational DBMSs will be described here, it is worth mentioning that currently described characteristic could be present in non-relational engines as well. A good example here are column-based storage systems that preserve data in structures called “column families” Such a construction is much alike to a relational table.

Number of table columns – referenced in studies [27], [26], [29]

As it is described above a feature characteristic to relational database model is a schema based on table data management. Each table can be defined with a various number of columns describing information that is contained in one. This is a commonly used metadata attribute and closely correlates to actual data structure. An interesting value in terms of this research is a total number of columns of all tables in a given database.

Number of records – referenced in studies [25], [29]

An important attribute that describes any relational database is a number of records that is preserved in one. It can be easily calculated as a sum of a number of rows of all tables defined in a database.

With a list of metadata attributes defined, an interesting task is a search for their values. Type of gathered values and a method for their calculation should be described before a research is continued. A natural approach for value computation is metadata analysis and averaging acquired data. An average of a set of values is representative for the whole set. Another methods of calculation were also analyzed such as median however in this example there is a danger of inaccurate outcome if input data is only slightly miscellaneous which is probable in cases of low number of input datasets. Calculating an average value for all acquired data is therefore a safe method of combining metadata characteristics.

An interesting result of the above specification of research protocol is creation of an “average” database image. Possible high number of investigated characteristics in a connection with multiple input data sources (analyzed database instances) across

numerous domains might conclude in an impression of a standard database. As much as such a vision is very promising it is worth noting that is difficult to achieve and requires enormous effort. This study aims to be a method description and an initial point for further discussion.

6.1 Data sources

In order to calculate values for above-defined database characteristics it is necessary to collect as many specimen as possible. A task of metadata analysis is not easy, particularly in this research where a high number of input objects is required to create a proper model database. That creates a requirement for a definition of data sources that will be used in this research. It is natural and necessary to underline that the increase of input metadata will lead to higher quality study thus all researchers should aim to extend it as much as it is possible however an activity of analyzing all available sources is practically impossible due to a constant data generation growth over the world [69].

Manual selection of database groups required for characteristics calculation is a laborious activity that requires contacting a huge number of subjects in database business domain. For obvious reasons such a task is hard to automate and would take an enormous amount of time to complete which in result makes it impossible to perform during a Master's thesis project. This problem can be solved by a usage of open datasets that are publicly available in Internet sources. Gathering required data however still remains troublesome due to a necessity of visiting and querying multiple websites that collect such data. Open data search engines come to rescue here. Various Web pages were created in order to facilitate open data browsing and mining some of which are: DataPortals.org [41], Datahub.io [70], Quandl [71], Socrata Open Data Network [72], or International Open Government Dataset Search [73].

When investigating big loads of data available online, an important aspect that greatly influences the ease of work is an availability of Application Programming Interface. Activities such as data collection and analysis are largely repetitive and similar between different datasets. Well defined API, together with programming knowledge, facilitate automation of many tasks that would require big effort otherwise. Among above defined open data engines not all examples present public API or do not document it appropriately. One, particularly interesting service is Quandl that specializes in gathering financial and economic data from all over the world. Extensively documented API and a big size of collected data sources [74] serve as a good reason for choosing this engine for future database characteristics analysis. The services provides free and premium access to its resources however it is necessary to outline that only the data that is available for free will be analyzed in this study.

6.2 Data acquisition

With an open data search engine defined, the next step of the research was to collect and measure all required characteristics. It has been decided that values for data size, number of data tables, number of table columns and number of rows will be calculated to describe a database image. Investigation of Quandl API documentation directly returns a couple of possible methods for data acquisition. First and most obvious is downloading the metadata pre-calculated by Quandl for each database. This approach has to be however crossed out as it does not provide all searched attributes. Quandl also blocks a possibility to download the whole database through the appropriate calls for databases accessible in free mode. After a detailed investigation of available API methods an approach was formulated to:

1. Download metadata information on names of all free databases accessible in Quandl.
 - a. This metadata contains a number of datasets (Quandl representative of a database table) in a given database
2. Choose databases that will be furtherly investigated.
3. Based on a list of gathered names, download identifications of all tables that are contained within given database.
4. Download each and every table for a given database and calculate values for searched characteristics.
 - a. The remaining characteristics such as database size, total number of rows in a database and total number of columns are obtained here.
5. Process results

Starting from the first step, the results were obtained by running single API call where a list of 400 databases was returned. The data contained information that allowed for a quick filtering out objects that were only accessible with premium privileges. That created a list of 314 databases with a number of tables per each. As the total number of tables accounted to over nine million, it is apparent that following the next procedure steps would create a big computation load. When assuming that a download of each data set and a calculation of the pursued values would take one second, the activity repeated for all data sets would take more than one hundred days. Such a time span is not acceptable therefore it has been decided that a number of investigated databases will be limited to a group of representatives. Additionally, increasing the performance of the script to run multiple requests per second might have overloaded Quandl servers resulting in an activity similar to DDOS attack. Furthermore simultaneous API calls from a single free user are prohibited by Quandl [75]. Systematic sampling was chosen as a method for evaluating most representative set of investigated objects. A situation described here, analysis of a certain group of example databases from the whole population, as described in [76, 77] is well suited for using systematic sampling. For these measures all gathered database names were sorted in a random order and then sixteen databases were chosen taking every 20th object starting from a randomly generated position. The distribution factor was chosen to limit number of investigated databases to a quantity that would facilitate analysis in a reasonable time span. A list of investigated databases along with their descriptions is presented in [78].

As a result, over 73 000 datasets were evaluated and analyzed. That allowed for a calculation of average values for characteristics defined above. Results are presented below while full list of obtained and raw data results can be accessed online at [79].

The summary of results is presented in Table 6.

Table 6 Summary of characteristics and their values for chosen databases in Quandl

Database code	Number of tables	Total number of rows	Total number of columns	Total physical size [bytes]
BATS	55169	41269724	110396	990473376
BOF	2621	335456	2621	5367296
DME	53	2504	424	180288
FMAC	466	260402	953	6546168
INDIA_LAB	5	1674	5	26784
ITU	1690	22727	1855	374440
LBMA	4	33608	28	2011912
MCX	970	88696	5820	4966976
NAHB	482	25963	2924	1397880

NYUSTERN	2	63	154	16000
ODA	7380	285056	7380	4560896
RFSS	180	1306	1546	117032
TUNISSE	2	6080	6	194560
WIKI	3186	14668665	38232	1525541160
WSE	800	1623180	6400	116868960
ZCE	675	145011	6075	11600880

Values presented above can be easily transformed into an overall image of a database on Quandl:

Table 7 Quandl database characteristics computations

Value	Number of tables	Total number of rows	Total number of columns	Total physical size [bytes]
Average	4605.31	3673132.19	11551.19	166890288.00
St. deviation	13616.50	10662760.82	27931.64	437918694.27
Average min¹	149.25	11740.63	621.38	539862.00
Average max²	9061.38	7334523.75	22481.00	333240714.00

6.3 Measured database

Research presented in the previous chapter resulted in a description of certain database characteristics and their average values measured from a number of chosen representatives. A final step in order to proceed to performance measurements is defining a database instance to be measured. Two approaches were considered here one of which was generating a database that would precisely map the average values from Table 7. That, however, could put a hazard onto validity of future performance measures due to artificial values of entities. To eliminate that risk, the measured database was chosen from all resources available on Quandl. All values from Table 7 were specified as requirements for the selection. Querying available open datasets resulted in choosing one representative for the future research that most precisely met the search criteria. Values of chosen characteristics for the selected database are presented below:

Table 8 Characteristics of database chosen for performance measurements.

Dataset name	Number of tables	Total number of rows	Total number of columns	Total physical size [bytes]
TSE	4039	4130774	20193	198276112

¹ Calculated as an average value of a given characteristic taking into account only 8 smallest representatives. Example: average min. number of tables is calculated as an average number of tables of 8 databases where these numbers were smallest amongst all representatives (DME, FMAC, INDIA_LAB, LBMA, NAHB, NYUSTERN, RFSS, and TUNISSE).

² Calculated as an average value of a given characteristic taking into account only 8 biggest representatives. Example: average min. number of tables is calculated as an average number of tables of 8 databases where these numbers were highest amongst all representatives (BATS, BOF, ITU, MCX, ODA, WIKI, WSE, and ZCE).

7 PERFORMANCE COMPARISON

Having collected the necessary information on average database structure, it is important to define the process of performance comparison.

7.1 Schema

An activity of creating an image of an average database is complicated as described in above chapters. One of the numerous challenges in this process is reflecting the relations in database model. The reason of such problem is the lack of availability of relationship data in publicly available open datasets. All of the sources identified in this research were lacking relationship data describing database model. It is necessary to remember however, that non-relational databases are by design modelled with no relations in mind. Connections between entities are characterized there by data redundancy.

7.1.1 Schema translation

The above study resulted in a relational-typed database model and the specific data that was preserved there. That constituted for a sufficient and measurable relational database instance. An activity necessary to achieve the aim of the study was the translation of the obtained relational schema to non-relational model. MongoDB offers a set of instructions related to RDBMS and NoSQL transformation, specification and maintenance. A document that was used in the process of schema translation of the database chosen in this research was the official RDBMS to MongoDB Migration Guide [80].

Following the guidelines stated in the above document a translation method was formulated to obtain properly formatted MongoDB data structures. An example of translated entity is presented below:

Table 9 Example relational data structure

Date	Open	High	Low	Close	Volume
30/03/2017	3005	3010	2936	2942	104000

Table 10 Example translated MongoDB data structure.

{ Date: 2017 - 03 - 30, Open: 3005.0, High: 3010.0, Low: 2936.0, Close: 2942.0, Volume: 104000.0 }

Transformation of big loads of data could be problematic if done manually. In order to automate the task and therefore minimize the number of errors a Python script has been developed.

7.2 Measurement method

Previous study on database performance measurement methodology presented in Chapter 4 resulted in a list of methods. It is necessary to choose the metric that will be applied in the future in order to proceed with the research. To ensure the reliability of the study and decrease research bias a metric that is precise in measurements, can be applied to any database type and defines a repeatable workload scenarios needs to be identified. Of all the conclusions listed in Table 3 only one method meets all the criteria described above and in subsection 4.5.1 which is Yahoo Cloud Serving Benchmark (YCSB). This tool offers an operation time measurement with up to microsecond precision, allows for source code modification and is free for use. An additional advantage of YCSB over competitive database performance measurement methods is the variety of metrics with the offer of latency, throughput and total runtime calculations.

It is necessary to highlight that total runtime metric indicates the time spent on executing the workload as well on extra activities connected with database operations. Examples of such extra activities could be garbage collection necessary for all Java applications (since YCSB benchmark used in the experiment was developed in Java programming language) or cleanup operations (necessary to shutdown application threads that handled the database connection). Therefore the value of that characteristic does not necessarily correlate with throughput which should be equal to total runtime when multiplied by the number of operations. The values for total runtime is intentionally included in the below analysis however it does not play a leading role in the comparison.

The design of the above application allows performance measurements for predefined workloads with various CRUD operations. The standard implementation however, performs the operations on a clean database instance. That was not acceptable for this research thus some modifications had to be made to tool in order to allow measurements on a predefined database instance. The fact that YCSB is an open source initiative greatly helped in understanding how the tool works and allowed for necessary modifications. The final version of the YCSB tool used in this research is available online under <https://github.com/KamilKolonko/YCSB>. List of code changes introduced over the base code is listed below:

- Add measured database to the repository
- Configure workloads for Oracle and MongoDB
- Add benchmark workload methods

The detailed definition of YCSB benchmark is published in [22]. The aim in the development of the application was to create an open source, precise and extensible tool that would allow future application in diverse environments. These goals are satisfied in the architectural design of the client where separate modules are responsible for various functionalities. The following modules are defined in the client from the high-level perspective: workload executor, client threads, analytics module and database interface layer. Workload executor component is responsible for parsing the information from configuration files and triggering client threads that are performing the operations in the database. Multiple threads are started, and each thread executes a sequential series of operations. The communication between the application and database is managed by the DB interface layer and is supported by database-specific Java drivers. In parallel the information on operation execution times is being collected and reported in the analytics module.

Database performance measurement operations in YCSB client are executed based on defined workloads. Each workload is a set of configuration properties that define the

proportions of the operations that are being executed on a database. Six workloads are defined in the source code with five explanations presented in the client description [22]. The distribution of the operations were defined by the research authors based on the examinations of various data systems and their applications. The types and proportions of the operations were chosen in order to explore the differences in the design of various databases where read, update, insert and scan transactions are optimized with different tradeoffs. Example usage scenarios were identified for each of the defined workloads and are referenced in Table 11. Random execution of the operations was achieved by the implementation of several distribution techniques e.g. uniform, zipfian, latest or multinomial. Distribution algorithm was selected for each of the defined workloads and applied when execution the workload. The selection defined in the research, along with workload operation details was not modified in the below experiment.

Table 11 Example applications for YCSB workloads. From [22]

Workload	Operations	Record selection	Application example
A—Update heavy	Read: 50% Update: 50%	Zipfian	Session store recording recent actions in a user session
B—Read heavy	Read: 95% Update: 5%	Zipfian	Photo tagging; add a tag is an update, but most operations are to read tags
C—Read only	Read: 100%	Zipfian	User profile cache, where profiles are constructed elsewhere (e.g., Hadoop)
D—Read latest	Read: 95% Insert: 5%	Latest	User status updates; people want to read the latest statuses
E—Short ranges	Scan: 95% Insert: 5%	Zipfian/Uniform*	Threaded conversations, where each scan is for the posts in a given thread (assumed to be clustered by thread id)

Workloads defined above were designed in order to verify and measure performance differences between database architectural models. Example descriptions for several databases are referenced in [22] with a clear indication on engine optimization between read and write operations. Trade-offs and system optimization between various operation types is also necessary in OracleDB and MongoDB. Research [81] indicates good optimization of MongoDB engine for random and parallel access, i.e. queries to the data while parallel write operations exhibit poor performance due to the global write lock. OracleDB's performance trade-offs between read and write operations are hard to unambiguously describe since the multitude of the configuration options that allows vast efficiency tuning. The default values however, described in [82], indicate well designed write performance with frequent usage of buffer cache and asynchronous write IO mechanism which, based on the analysis in [22] of similar architecture, reveals write-optimization strategy of the engine. The workload definition is therefore expected to verify the performance of each database model and test the trade-offs made when designing each system's architecture. Specific example is with workload A that is designed to be update heavy where OracleDB should represent good performance and workload B that is read heavy where MongoDB is expected to present its architectural benefits.

Neither of the databases selected for the performance measurements in this study were additionally optimized. No configuration that could affect the measured performance was applied. Both databases were running in an out-of-the-box state. This fact might have a big effect on the performance, especially in the case of the relational databases since the addition of indexes might have a significant impact on the query performance. This is discussed in the Threats to Validity section.

7.3 Workload definition

YCSB Benchmark comes with 6 predefined workloads that specify the type and the number of operations that are executed on a database. Example operation types could be inserting, reading or updating entities stored in the database. Abovementioned

workflows are defined as stated below with the following definitions for each property: *operationcount* – number of executed operations, *readproportion* – proportion of read operations (e.g. 0.5 means 50% of total operations are database value reads), *updateproportion* – proportion of update operations, *scanproportion* – proportion of scan (full collection reads) operations, *insertproportion* – proportion of insert operations, *readmodifywriteproportion* – proportion of batches of read, update and insert operations.

Workload A:

- `operationcount=1000`
- `readproportion=0.5`
- `updateproportion=0.5`
- `scanproportion=0`
- `insertproportion=0`

Workload B:

- `operationcount=1000`
- `readproportion=0.95`
- `updateproportion=0.05`
- `scanproportion=0`
- `insertproportion=0`

Workload C:

- `operationcount=1000`
- `readproportion=1`
- `updateproportion=0`
- `scanproportion=0`
- `insertproportion=0`

Workload D:

- `operationcount=1000`
- `readproportion=0.95`
- `updateproportion=0`
- `scanproportion=0`
- `insertproportion=0.05`

Workload E:

- `operationcount=1000`
- `readproportion=0`
- `updateproportion=0`
- `scanproportion=0.95`
- `insertproportion=0.05`

Workload F:

- `operationcount=1000`
- `readproportion=0.5`
- `updateproportion=0`
- `scanproportion=0`
- `insertproportion=0`
- `readmodifywriteproportion=0.5`

All workloads executed in below research were configured to run with an asynchronous database drivers.

7.4 Hardware outline

All of the tests were performed on a machine with a following hardware specification:

- CPU: Intel Core i7-7500U @ 2.70GHz 2.90GHz
- RAM: 16GB
- GPU: Intel HD Graphics 620
- HDD: NVMe 512GB Toshiba THNSN5512GPU7

8 RESULTS

All of the workloads above were executed on both of the measured databases. Default values were not modified as it allows for an easy comparison between various studies on database performance that used YCSB as a benchmarking method and upheld the default workload distribution e.g. [35] or [34]. Results of executed measurements are presented below.

8.1 MongoDB

Workload A

- Total runtime: 979 ms
- Throughput: 1021.45 ops/sec

	Read	Update
Average latency [μ s]	599.85	1087.50
Min latency [μ s]	92.0	174.0
Max latency [μ s]	14663.0	131711.0
95 th percentile latency [μ s]	1125.0	1459.0
99 th percentile latency [μ s]	1431.0	1934.0

Workload B

- Total runtime: 1254 ms
- Throughput: 797.45 ops/sec

	Read	Update
Average latency [μ s]	1095.04	1632.70
Min latency [μ s]	73.0	485.0
Max latency [μ s]	121279.0	15223.0
95 th percentile latency [μ s]	2013.0	2863.0
99 th percentile latency [μ s]	2883.0	3451.0

Workload C

- Total runtime: 824 ms
- Throughput: 1213.59 ops/sec

	Read
Average latency [μ s]	687.11
Min latency [μ s]	78.0
Max latency [μ s]	118975.0
95 th percentile latency [μ s]	1117.0
99 th percentile latency [μ s]	1740.0

Workload D

- Total runtime: 620 ms
- Throughput: 1612.90 ops/sec

	Read	Insert
Average latency [μ s]	487.10	631.73
Min latency [μ s]	63.0	191.0
Max latency [μ s]	105471.0	12279.0

95 th percentile latency [μs]	1155.0	746.0
99 th percentile latency [μs]	1765.0	932.0

Workload E

- Total runtime: 1169 ms
- Throughput: 855.43 ops/sec

	Insert	Scan
Average latency [μs]	970.32	988.57
Min latency [μs]	413.0	198.0
Max latency [μs]	15743.0	152831.0
95 th percentile latency [μs]	1568.0	1854.0
99 th percentile latency [μs]	15743.0	2703.0

Workload F

- Total runtime: 839 ms
- Throughput: 1191.90 ops/sec

	Read	Read-Modify-Write	Update
Average latency [μs]	434.29	1073.04	445.24
Min latency [μs]	82.0	234.0	147.0
Max latency [μs]	155755.0	186239.0	14495.0
95 th percentile latency [μs]	537.0	1339.0	770.0
99 th percentile latency [μs]	1031.0	2175.0	1604.0

8.2 Oracle

Workload A

- Total runtime: 18369.0 ms
- Throughput: 54.44 ops/sec

	Read	Update
Average latency [μs]	24226.56	9724.71
Min latency [μs]	83.0	4244.0
Max latency [μs]	10870783	22159.0
95 th percentile latency [μs]	4131.0	12439.0
99 th percentile latency [μs]	4647.0	12943.0

Workload B

- Total runtime: 4721 ms
- Throughput: 211.82 ops/sec

	Read	Update
Average latency [μs]	2756.91	10926.16
Min latency [μs]	98.0	6132.0
Max latency [μs]	650239.0	13183.0
95 th percentile latency [μs]	4087.0	12879.0

99 th percentile latency [μs]	4683.0	13183.0
--	--------	---------

Workload C

- Total runtime: 4011.0 ms
- X`Throughput: 249.31 ops/sec

	Read
Average latency [μs]	2856.57
Min latency [μs]	114.0
Max latency [μs]	565759.0
95 th percentile latency [μs]	3661.0
99 th percentile latency [μs]	4111.0

Workload D

- Total runtime: 3745 ms
- Throughput: 267.02 ops/sec

	Read	Insert
Average latency [μs]	2135.34	8526.81
Min latency [μs]	64.0	4768.0
Max latency [μs]	1017343.0	13311.0
95 th percentile latency [μs]	3943.0	11479.0
99 th percentile latency [μs]	4251.0	13263.0

Workload E

- Total runtime: 4889 ms
- Throughput: 118.40 ops/sec

	Insert	Scan
Average latency [μs]	7604.23	3357.46
Min latency [μs]	4576.0	207.0
Max latency [μs]	11591.0	620543.0
95 th percentile latency [μs]	11135.0	4523.0
99 th percentile latency [μs]	11255.0	5443.0

Workload F

- Total runtime: 8446 ms
- Throughput: 118.40 ops/sec

	Read	Read-Modify-Write	Update
Average latency [μs]	2967.66	11010.91	8633.52
Min latency [μs]	83.0	4312.0	4112.0
Max latency [μs]	565247.0	24111.0	19663.0
95 th percentile latency [μs]	4155.0	14447.0	10479.0
99 th percentile latency [μs]	4591.0	15127.0	11271.0

9 ANALYSIS AND DISCUSSION

Database performance can be measured by a number of different factors. The method used in this research allowed us to collect multiple metrics on Oracle Database's and MongoDB's efficiency. Having gathered all the results from executed experiment it is necessary now to analyze and compare the values. Amongst the parameters measured in the test are: total runtime of the workload, throughput represented in number of operations per second and average, minimum, maximum, 95th and 99th percentile latencies. The direct comparisons between the performance scores of each database are presented below separately for each workload.

9.1 Workload A

The results measured for both of the databases are presented in Figure 2.

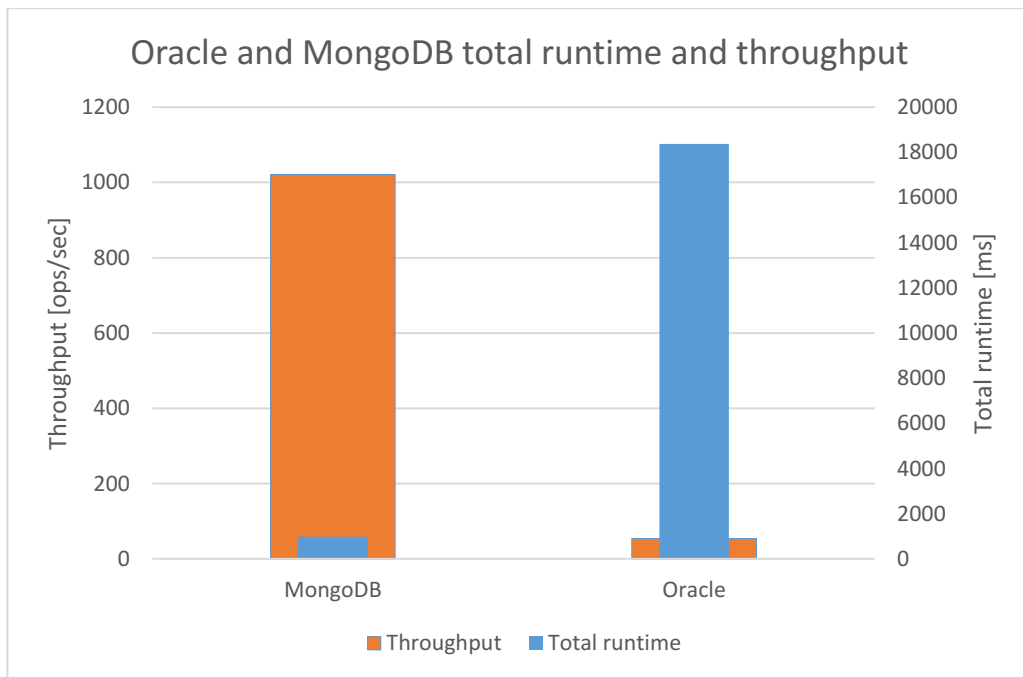


Figure 2 Workload A - Oracle and MongoDB total runtime and throughput

As it is visible in the figure above, running Workload A against both databases resulted in drastically different results. As it will turn out soon, this is the case in all of the measurements. This workload balances read and update operations in 50/50 ratio. MongoDB has a clear advantage here over Oracle, reaching 1021.45 operations per second which is 23 times more than the relational database.

Another characteristic that should be analyzed here is the operation latency. Average, minimum, maximum, 95th and 99th percentile values were calculated for this aspect have been investigated. Visual representations are presented in the figures below.

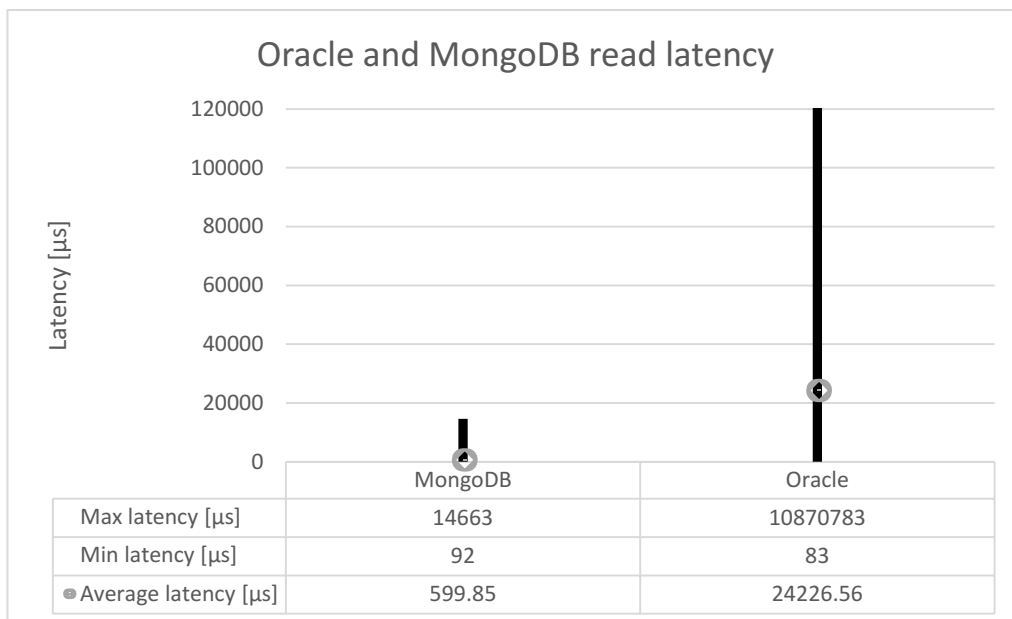


Figure 3 Workload A - Oracle and MongoDB read latency

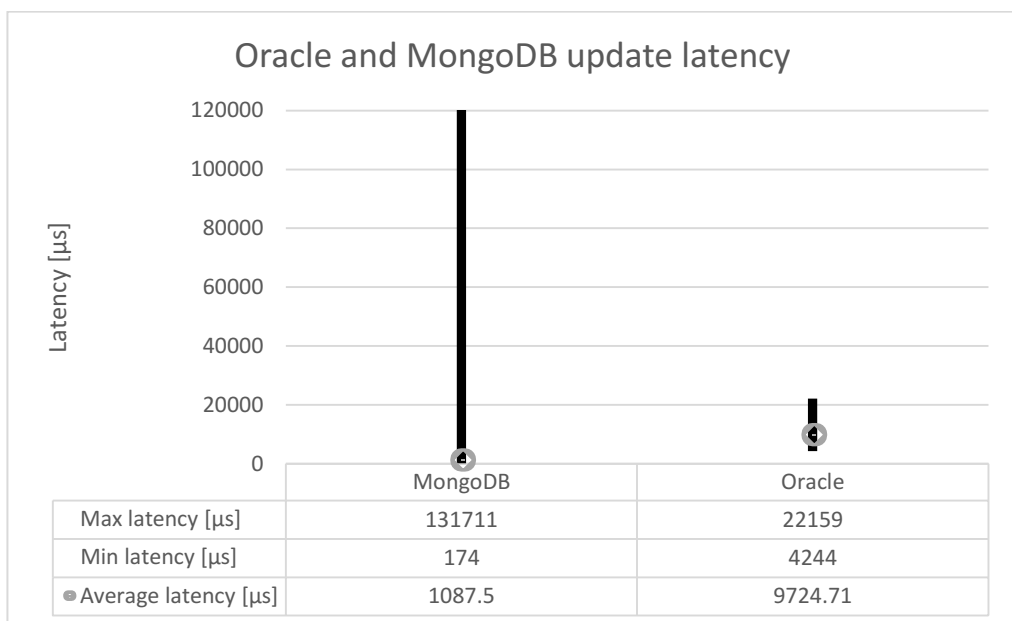


Figure 4 Workload A - Oracle and MongoDB update latency

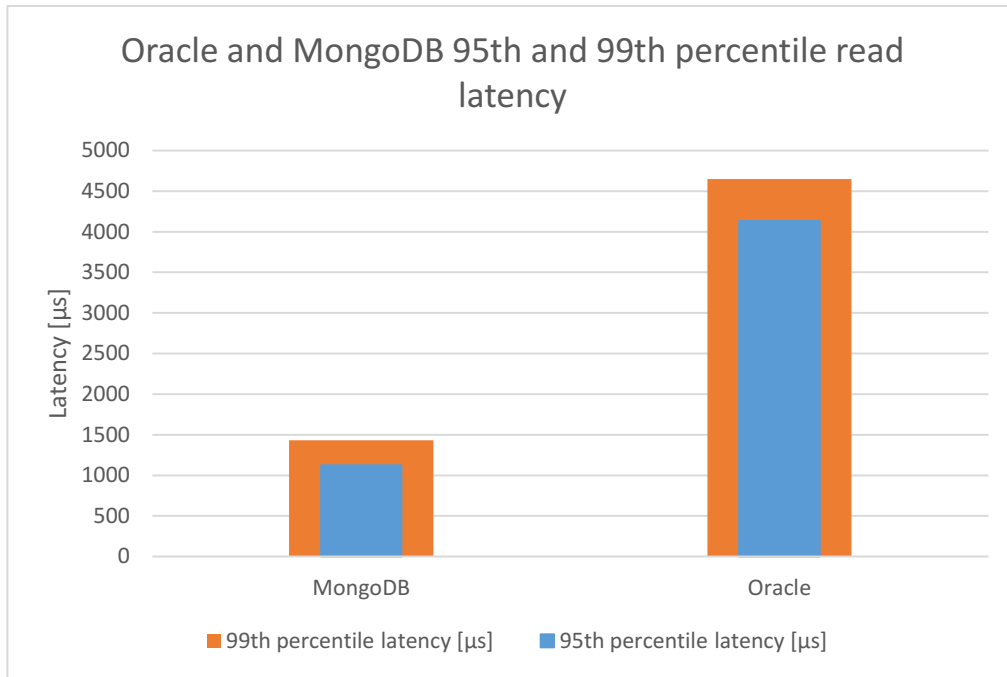


Figure 5 Workload A - Oracle and MongoDB 95th and 99th percentile read latency

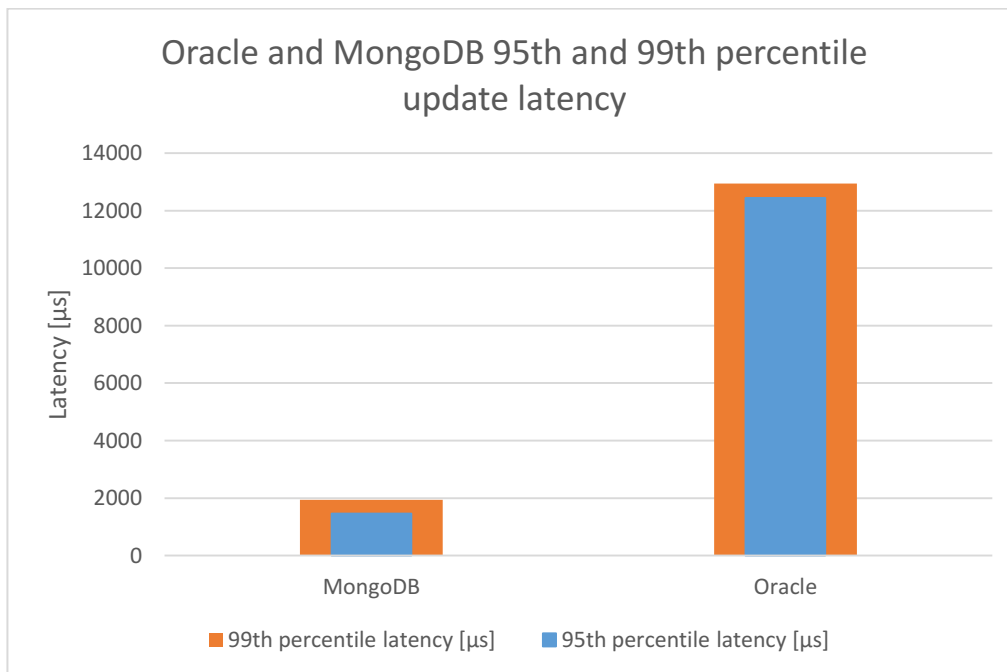


Figure 6 Workload A - Oracle and MongoDB 95th and 99th percentile update latency

The figures above again present an advantage for NoSQL representative over the traditional relational database. Figure 3 clearly defines the difference between the measured values for both databases. The average latency of OracleDB is higher than competitor's by over 2300μs. An interesting observation is the minimum value for that aspect as it is lower for Oracle than for MongoDB. That might be caused by a nearby location in memory of randomly chosen entities. Nevertheless, it is safe to assume that the advantage of Oracle over MongoDB is incidental here as the average value proves the dominance of the latter.

Another interesting situation is visible in Figure 4 where although the lead of MongoDB is visible again, the maximum latency of such is significantly higher than the competitor's. OracleDB represents here a much more stable results where a relatively small gap between minimum and maximum latencies is a proof of that. This observations is however put into consideration when reviewing results presented in Figure 5 and Figure 6. 95th and 99th percentile latencies for both read and update operations are lower for MongoDB indicating that the maximum latency analyzed in Figure 4 is not registered often.

The conclusion of the performance analysis of Oracle database and MongoDB in the workload A scenario clearly favors the latter as the more efficient. It is worth noting however that Oracle has its strong points as well and delivered more stable results when considering update operations.

9.2 Workload B

Workload B is a read heavy configuration with 95% of the tasks being reads and only 5% writes. This is a good representation of a system that is focused on responding with information that are already saved in the database rather than frequent writing and updating the data. Benchmarking delivered interesting results that, similarly to the workload A, outline the performance advantages of MongoDB. The visual representations of the comparison between the registered numbers are presented in Figure 7.

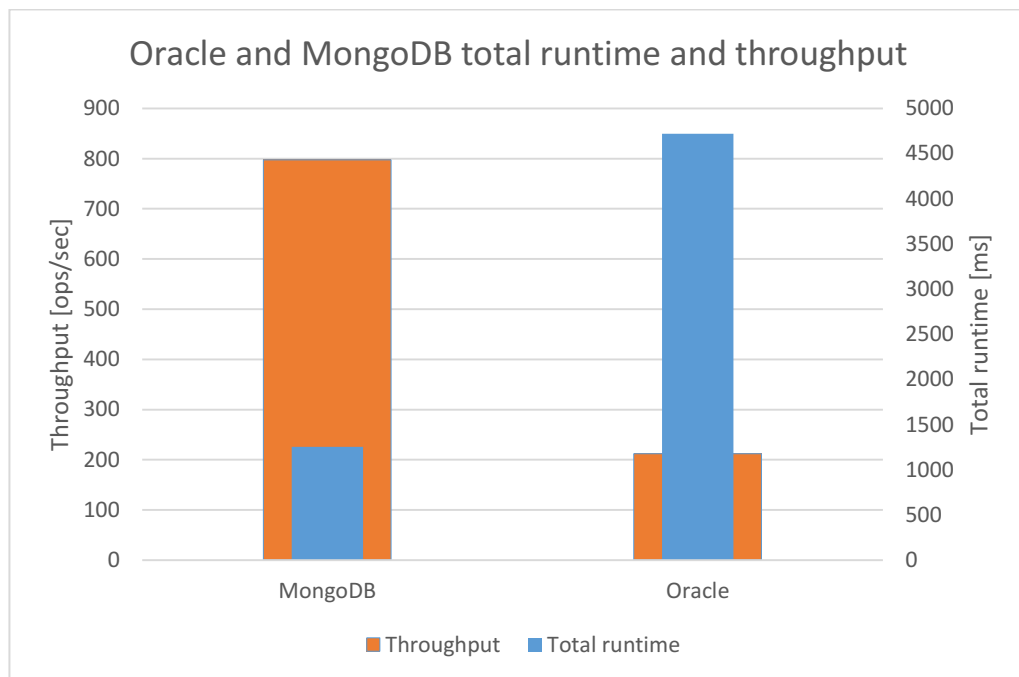


Figure 7 Workload B - Oracle and MongoDB total runtime and throughput

As easily observed on the picture above, the average throughput of MongoDB database is around 4 times higher than Oracles. 800 operations per second is however a decrease for Oracle when comparing the results with the previous workload while Oracle database performed better than in the former test.

Lower efficiency of Oracle is also visible when comparing latencies of the operations. The average value for the read operation is almost 3 times higher for the database compared to MongoDB. The situation is similar when investigating write

operations where the latency of the non-relational database is over 6 times lower than the competitor's. The comparison of such is visible in Figure 8 and Figure 9.

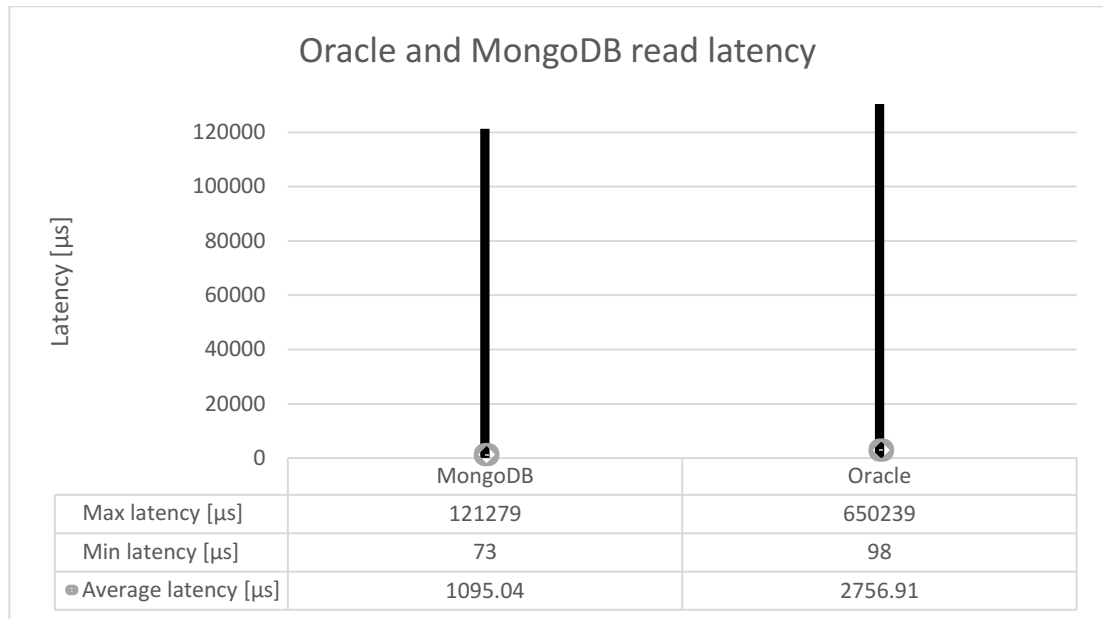


Figure 8 Workload B - Oracle and MongoDB read latency

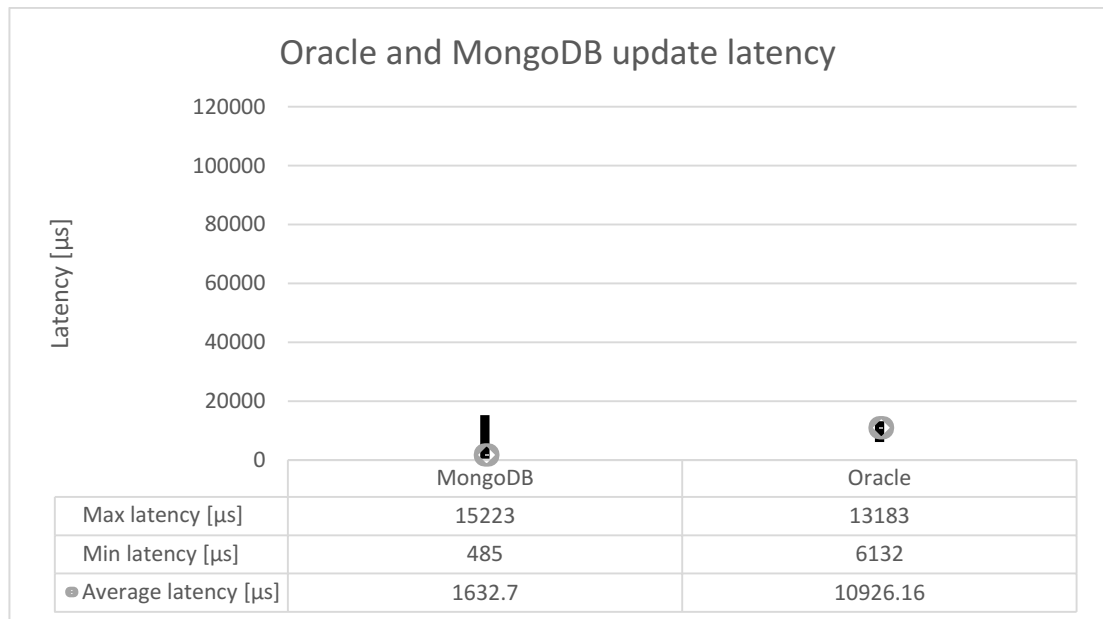


Figure 9 Workload B - Oracle and MongoDB update latency

It is necessary to mention that the vertical scale presented in Figure 8 was limited to the maximum value of 130,000µs to make the results more visible. The interesting point here, when analyzing latencies of both instances is the stability of update operation in Oracle database engine. Again, relational database presented much smaller distribution between the minimum and maximum values rather than the competitor. This however didn't improve the overall operation efficiency while the average latency value for that type of the operations was much lower for the non-relational database.

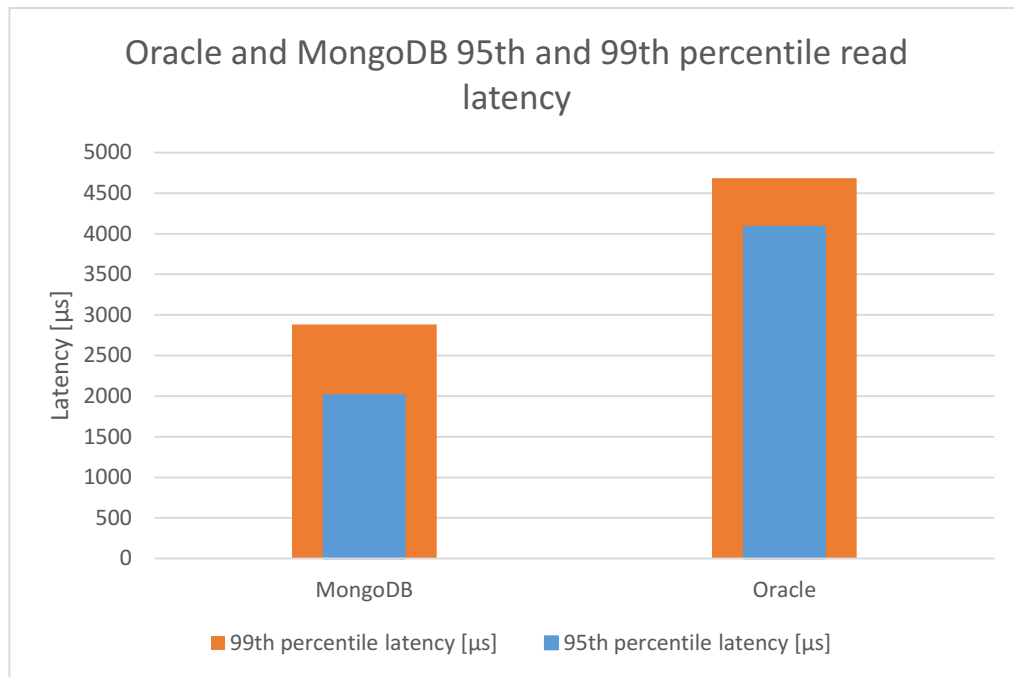


Figure 10 Workload B - Oracle and MongoDB 95th and 99th percentile read latency

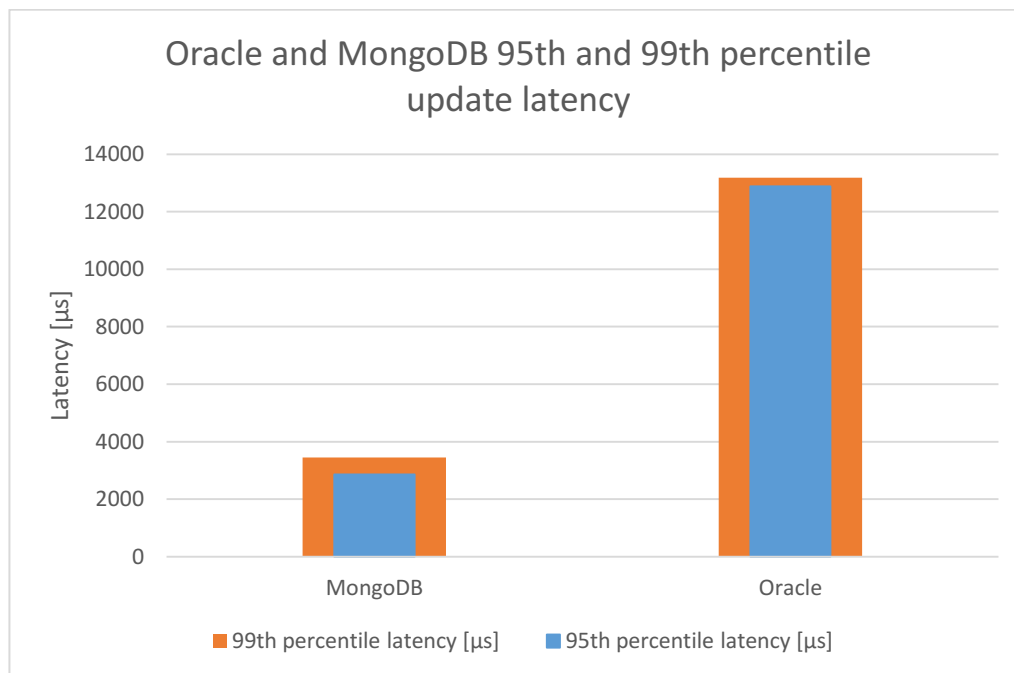


Figure 11 Workload B - Oracle and MongoDB 95th and 99th percentile update latency

An observation that becomes apparent when depicting the numbers for 95th and 99th percentile latencies calculated for Workload B is the relatively small gap between two analyzed databases in the read operation. It is also worth noting that the differences between 95th and 99th percentile values for each database alone, are higher for MongoDB. This leads to a conclusion that the number of extreme latency values for Oracle database was comparably lower than for MongoDB. The analysis of Workload C should clarify the influence of write operations on the database efficiency as it contains only read tasks.

9.3 Workload C

With an emphasis on read operations, workload C is a good reference model to the tasks analyzed in previous chapters. While mixing the types of the queries certainly affects the overall database performance, working on a set of single-typed transactions provides a better synthetic view into the comparison. Intuition suggests that the number of business use cases for a read only database is lower than for multi-operational ones. This however does not undermine the importance of the benchmark as we must remember that multipurpose databases can happen to operate with long sequences of read only operations frequently.

The results of the benchmarking experiment presenting the total time and throughput of each of the analyzed databases are presented in Figure 12.

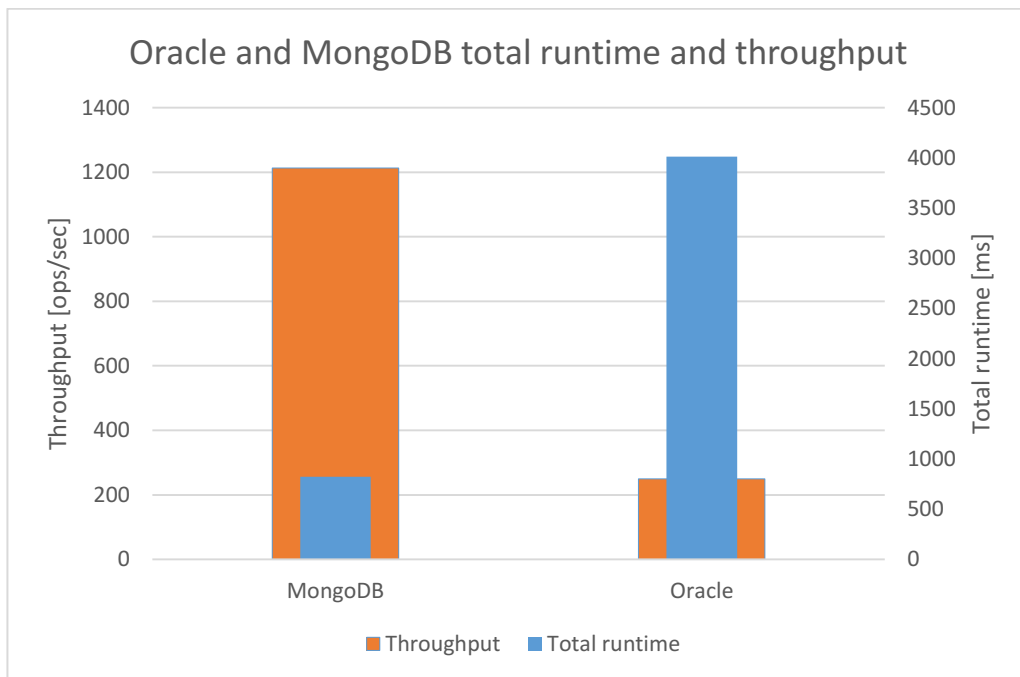


Figure 12 Workload C - Oracle and MongoDB total runtime and throughput

As it is presented above, the time scored by competitors are again favoring MongoDB. What is interesting however, is the proportion between total runtime and throughput of both databases. Taking into consideration that both databases executed the same number of operations, the difference between runtime to throughput ratio for both databases represents the scale of the additional operations that were performed during the experiment i.e. garbage collection and thread management. The results above indicate that this difference between measured objects is minimal therefore the observation, that both executions spent proportionate time on extra activities, can be made.

The values scored by both, relational and non-relational representative, in the read latency comparison, are not a surprise given the previously analyzed workload. The visual descriptions are presented in Figure 13 and Figure 14. MongoDB proves its dominance over relational competitor with an average operation latency more than 4 times lower than Oracle's. Please note that the scale representing the maximum latency in Figure 13 has been limited to 120000 μ s to improve readability of the diagram. Before switching to another workload analysis it is worth looking at the comparison of the 95th and 99th percentile latencies. It is visible again that the proportion between both values

for each database, visible as a gap between orange and blue bars in Figure 14 indicate that Oracle registered proportionally lower distribution between operation latencies. A conclusion that might be drawn from the fact is that the results registered for such favor Oracle in stability factor.

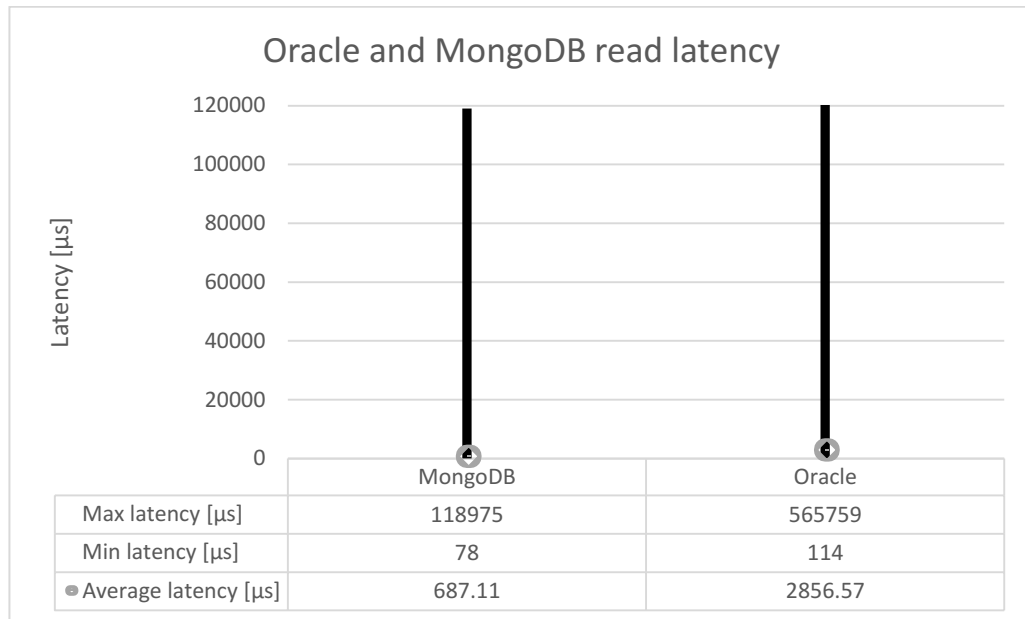


Figure 13 Workload C - Oracle and MongoDB read latency

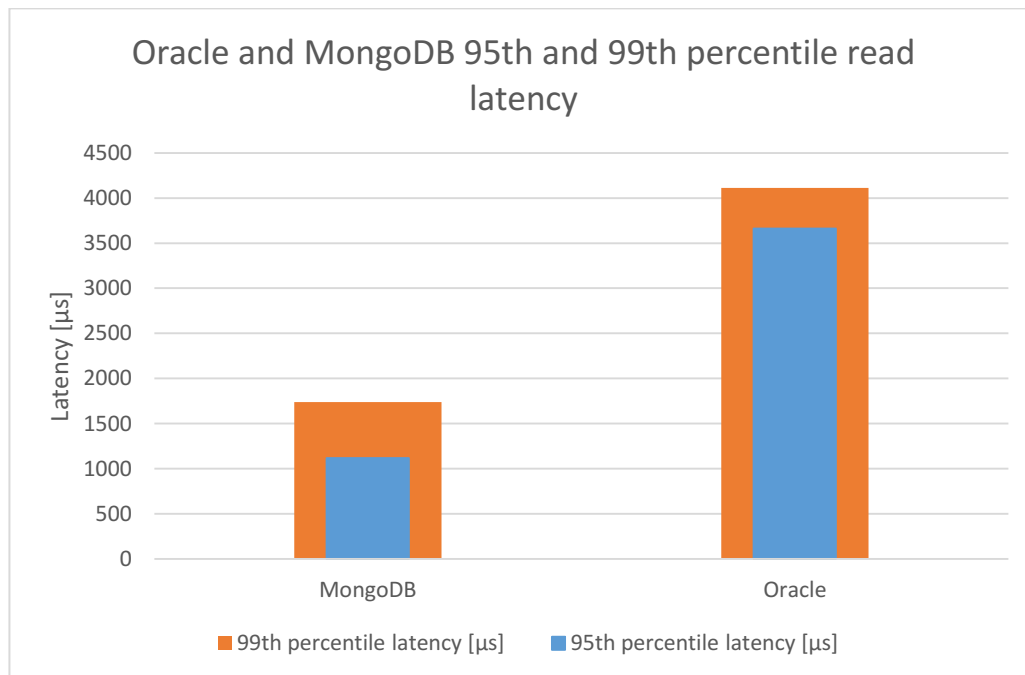


Figure 14 Workload C - Oracle and MongoDB 95th and 99th percentile read latency

9.4 Workload D

Having measured the performance in read and update heavy scenarios, workload D puts more emphasis onto insert operations. It is designed to balance reads and inserts in 95% to 5% ratio. The total runtime and throughput results for this experiment are presented in Figure 15.

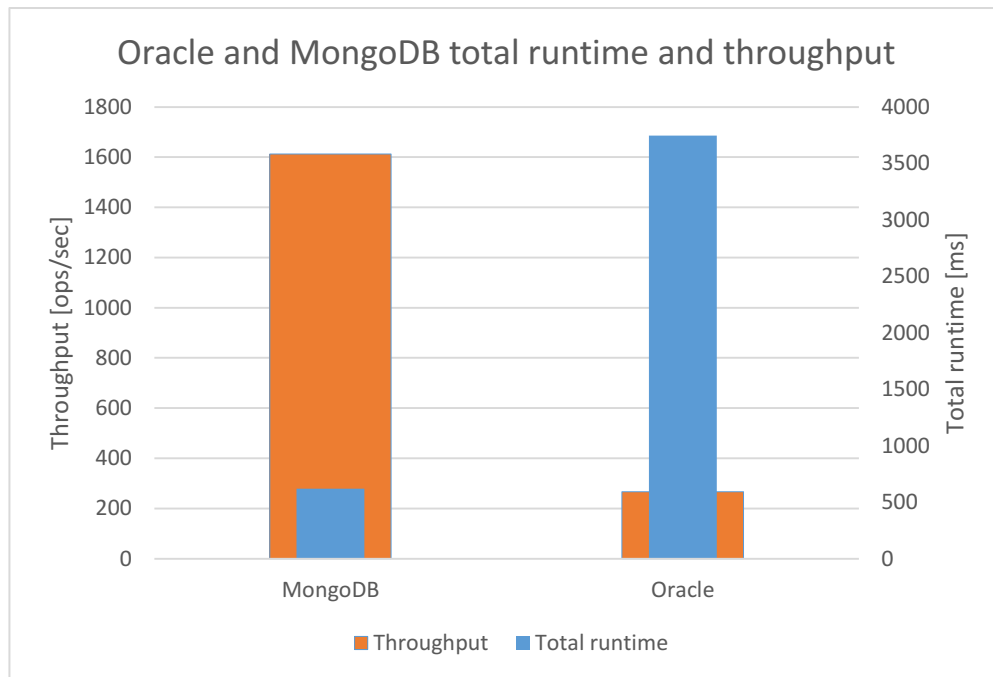


Figure 15 Workload D - Oracle and MongoDB total runtime and throughput

As it is visible above, the results are similar to the previous experiments. This is caused by a significant proportion of read operations over other types of queries for all precedent scenarios. MongoDB delivers a throughput result over 5 times higher than the competitor.

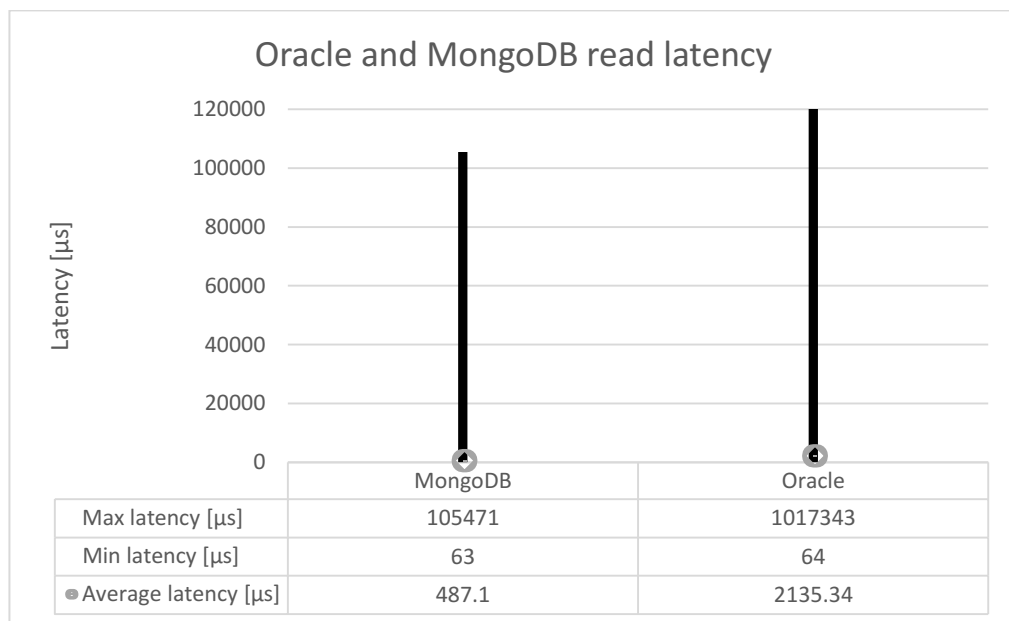


Figure 16 Workload D - Oracle and MongoDB read latency

More interesting results are observed in Figure 16 where the read latency for investigated databases is presented. Again, the vertical axis's scale was limited to offer better visibility. The eye-catching phenomena here is that Oracle database registers a minimum latency which is higher than competitor's by only 1μ s, the value negligible measurement error. This proves the relational database representative to be able to perform as fast as MongoDB under some circumstances.

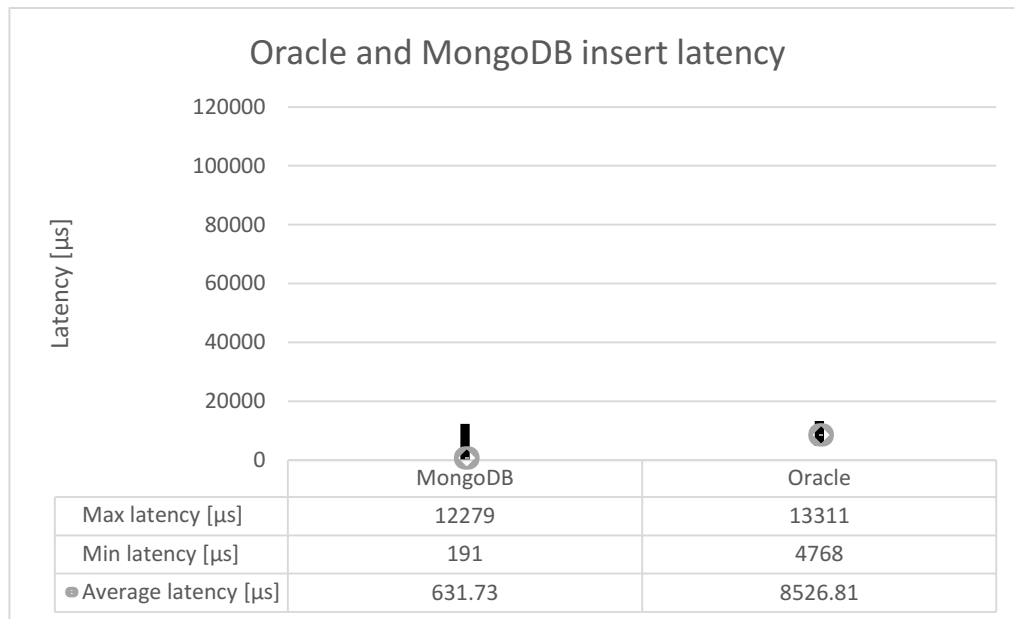


Figure 17 Workload D - Oracle and MongoDB insert latency

Figure 17 describes the insert latency of both databases registered under workload D. The first observation here, other than the dominance of MongoDB, is the narrow difference between the minimum and maximum latency for Oracle. The obtained results look stable and the amplitude is relatively small. Another valid point is the minimal inequality between MongoDB's and Oracle maximum latency. While being better performant, MongoDB has troubles with keeping the latencies balanced over time.

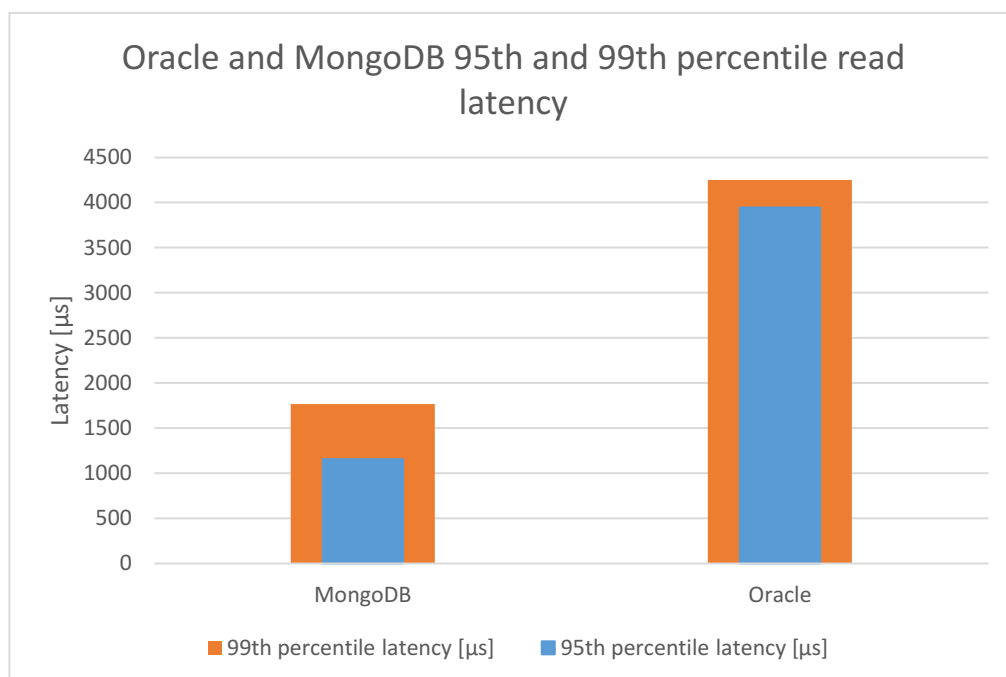


Figure 18 Workload D - Oracle and MongoDB 95th and 99th percentile read latency

The information presented in Figure 18 does not differ from the readings for same the characteristics in previous workload examples. Read latency again proves to be an advantage of MongoDB while the 95th to 99th percentile ratio is more favorable to Oracle database.

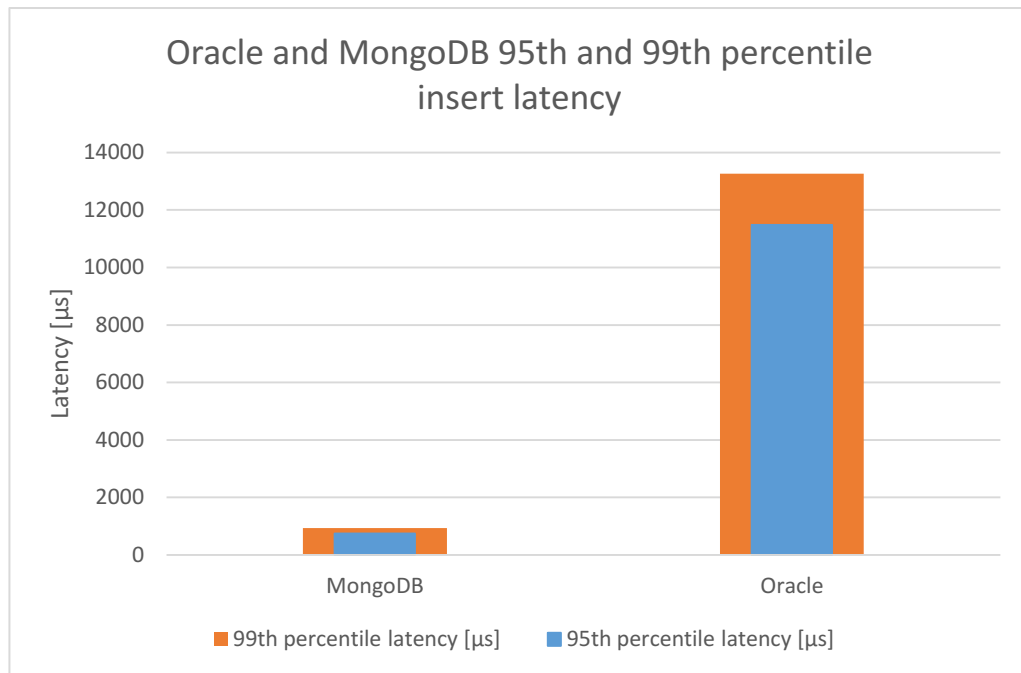


Figure 19 Workload D - Oracle and MongoDB 95th and 99th percentile insert latency

An interesting observation visible in Figure 19 is the unexpectedly low value for 99th percentile insert latency for MongoDB. The numbers for the maximum insert latency that we've previously observed and discussed near Figure 17 turn out to be single operations not visible in the top 1% of latency measures. Curious is also the minimal difference between MongoDB's 95th and 99th percentile latencies. The advantage of MongoDB here is contrary to what we've observed previously, when analyzing the read latency.

9.5 Workload E

The definition of workload E is significantly different than previous examples. The file outlines the proportion of operations to be 95% scans and 5% inserts. The scan operation as much as it is similar to a read, differs from one in that collection index is not used and the whole set of data is analyzed to get query result. Such execution schema obviously affects performance thus the results of the experiment should bring some new information to database comparison.

Figure 20 below presents total runtime for the workload execution and throughput of each database. The first observation that strikes mind is the correlation of throughput to runtime ration of both databases. The chart indicates that garbage collection and thread management was relatively more time expensive for MongoDB. This is the first situation where such an outcome is observed. However, the advantage of MongoDB over Oracle in the workload throughput gets even more visible with almost eight times more operations executed per second.

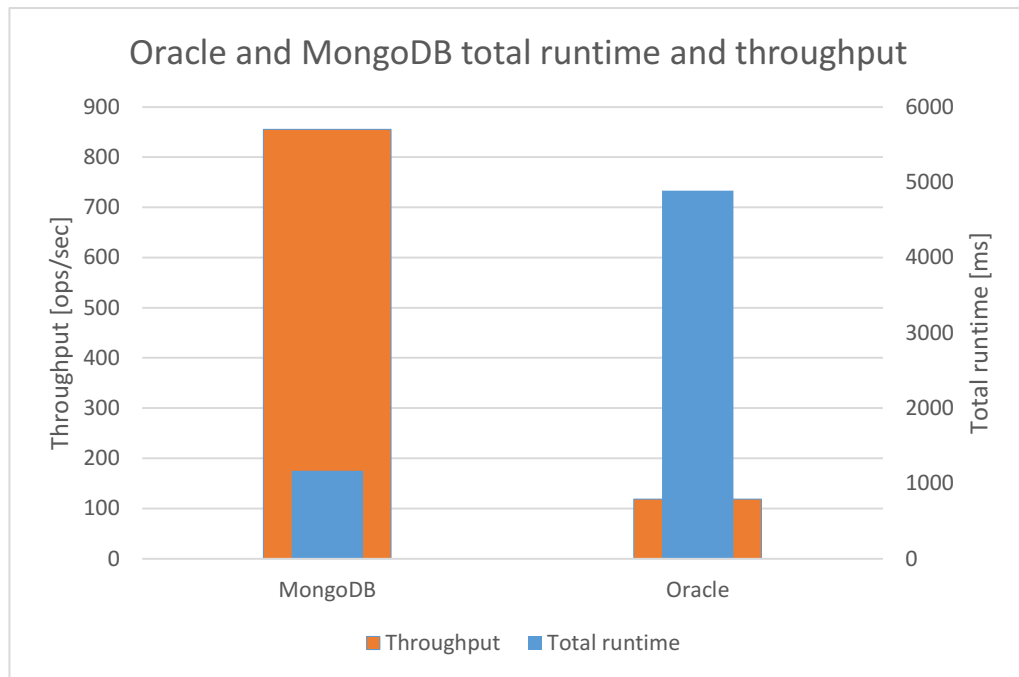


Figure 20 Workload E - Oracle and MongoDB total runtime and throughput

An interesting observation is visible in Figure 21 where the average scan latency is only ~3.4 times higher for Oracle than for MongoDB. It's also worth noting that MongoDB for the first time registered the maximum latency higher than 120000 μ s.

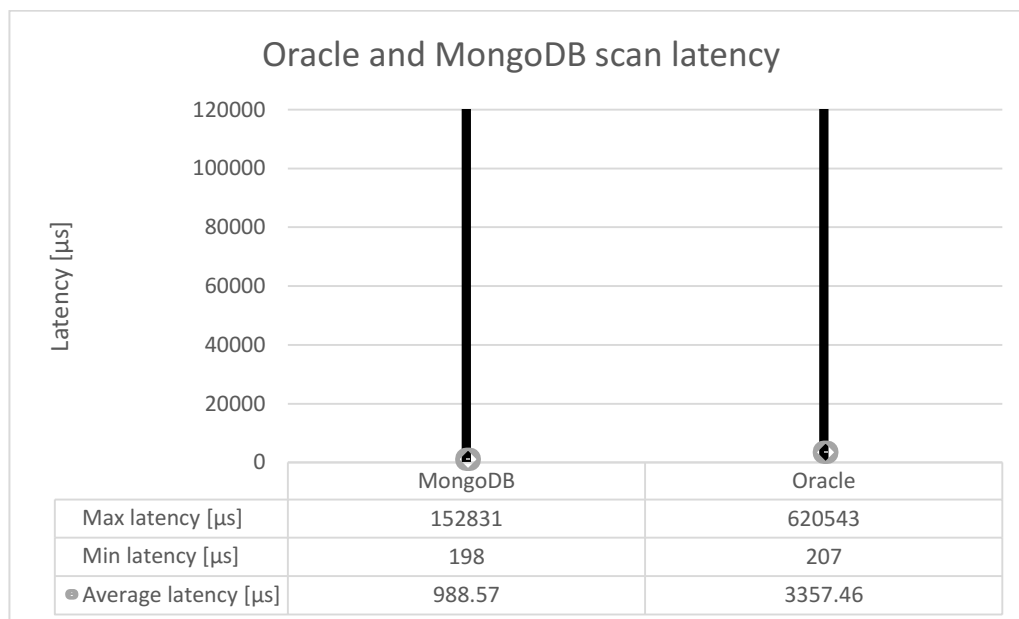


Figure 21 Workload E - Oracle and MongoDB scan latency

Figure 22 delivers more information on databases insert latency. The results here are very much comparable to the ones already analyzed with workload D. Oracle database proves its stability again with the low amplitude between minimum and maximum latency. The situation is contrary for the competitor. MongoDB scores a significantly higher maximum latency outcome however the minimum value for such is also notably lower than Oracle's. Overall, the average insert latency of MongoDB is 11 times lower than Oracles.

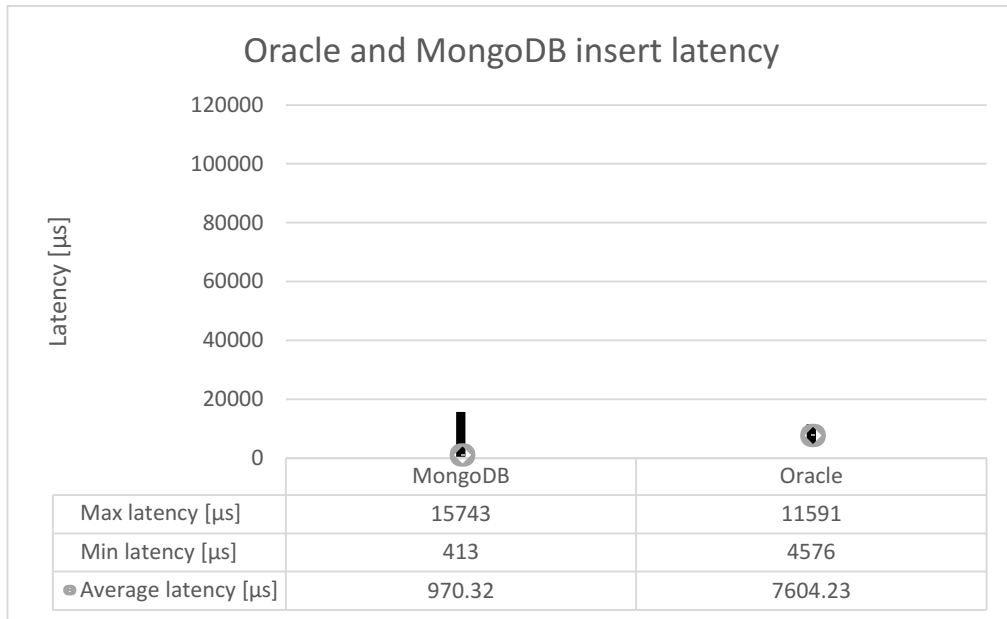


Figure 22 Workload E - Oracle and MongoDB insert latency

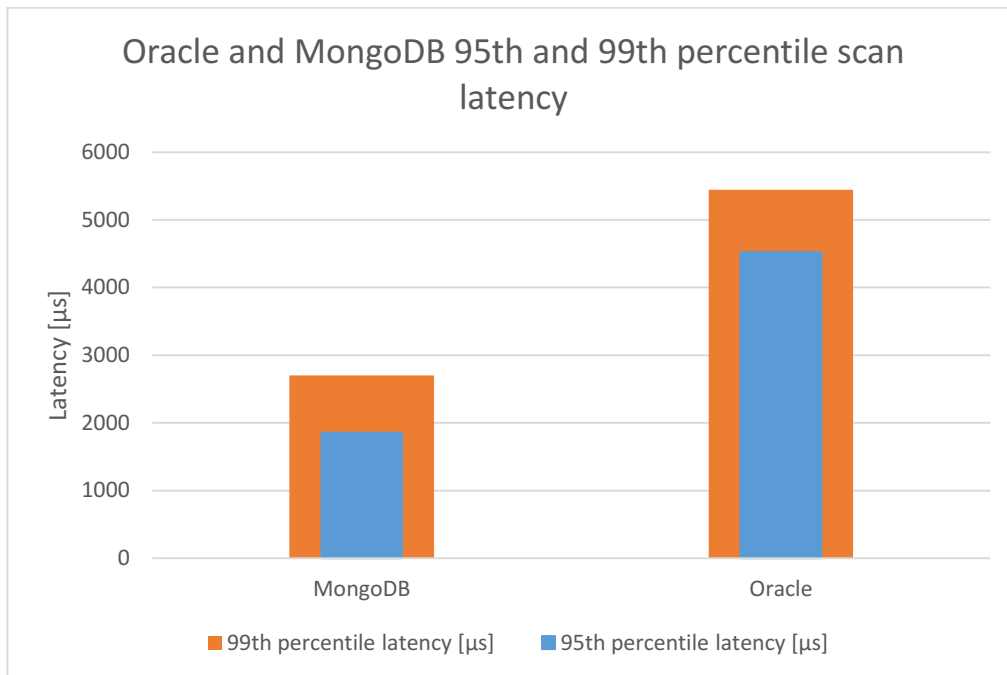


Figure 23 Workload E - Oracle and MongoDB 95th and 99th scan latency

As Figure 23 shows no particularly interesting observation with MongoDB having a clear advantage in scan operations over Oracle database, the information in Figure 24 introduces more new outcomes. As it is visible in the picture below, the values for MongoDB's 99th percentile insert latency significantly exceeds the one registered for Oracle. This means that the high latency values registered previously and depicted in Figure 22 are not incidental and account for at least 5% of total operations. Additional eye-catching phenomena is the nearly identical 95th and 99th percentile latencies registered for Oracle. This confirms relatively low distribution of the values presented in Figure 22.

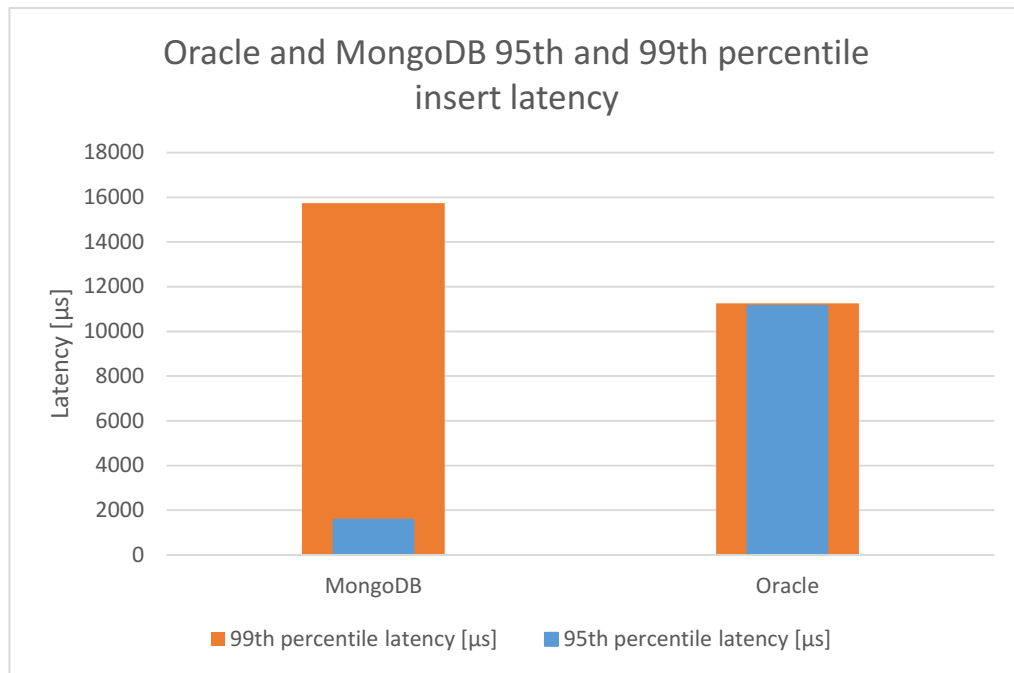


Figure 24 Workload E - Oracle and MongoDB 95th and 99th percentile insert latency

9.6 Workload F

Workflow F introduces a new type of the operations defined as read-modify-write. This is aimed to measure the performance of reading an entity in the database, modifying the data and then saving the result in an update transaction. Additionally, the definition of the workload incorporates a significant proportion of reads, totaling in 50%-50% ratio for both activity types.

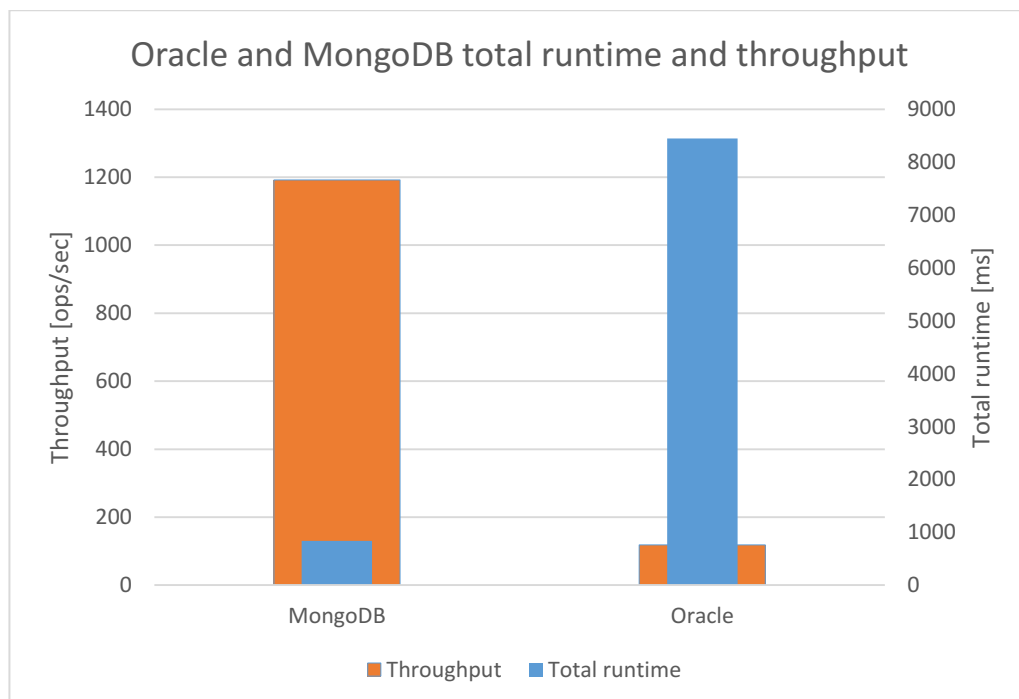


Figure 25 Workload F - Oracle and MongoDB total runtime and throughput

Analysis of the throughput values registered for both databases quickly indicates the advantage of MongoDB over Oracle's relational system. NoSQL database

representative resulted in over 10 times higher throughput than competitor. Such a number indicates an enormous advantage of the first system in the more complex workload definition however the exact numbers will be analyzed below in more detail.

When analyzing workload results, the operations will be divided into three categories. Since the read-modify-write operation consists of three smaller, divisible parts, the groups of data will be investigated in the following configurations:

- Read operations – total of 1000 operations summed as 500 reads in the workload definition and 500 reads from read-modify-writes
- Read-modify-write operations – total of 500 read-modify-writes
- Updates – total of 500 writes from read-modify-write

The results for each of the above categories are presented in Figure 26, Figure 27 and Figure 28.

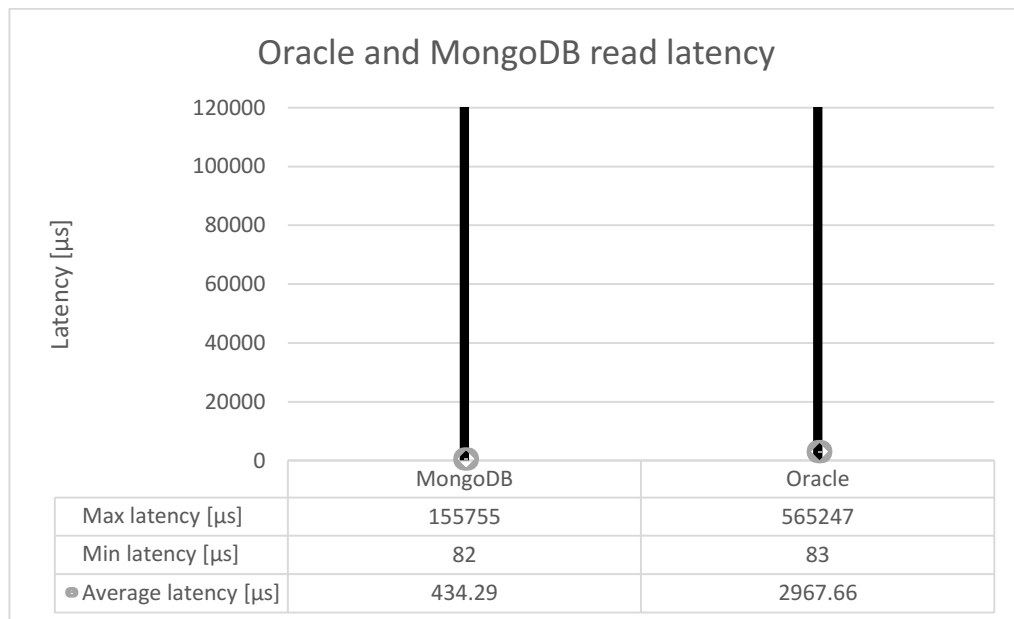


Figure 26 Workload F - Oracle and MongoDB read latency

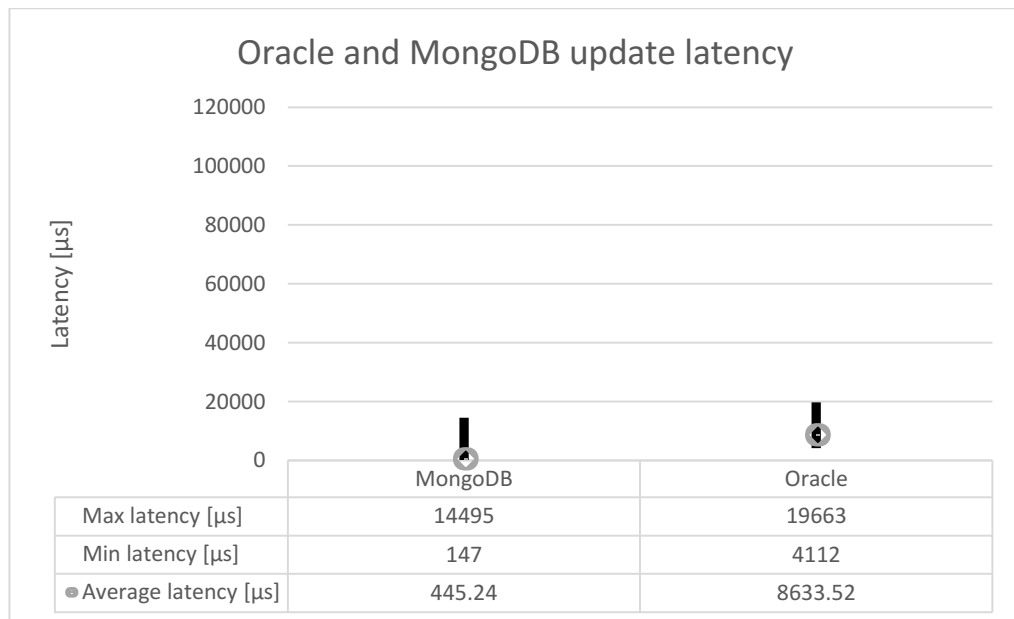


Figure 27 Workload F - Oracle and MongoDB update latency

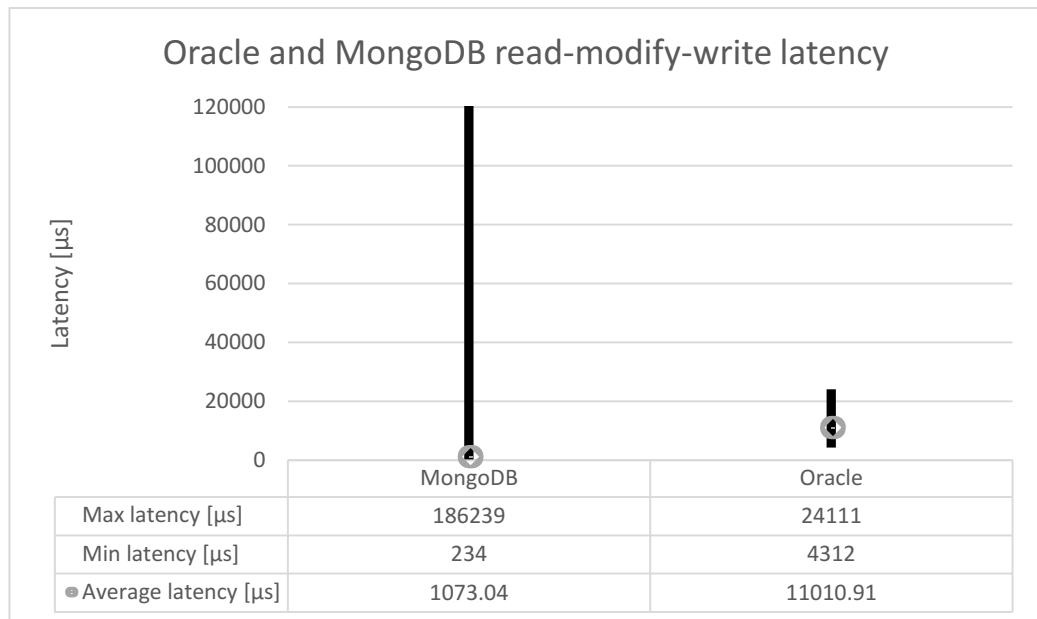


Figure 28 Workload F - Oracle and MongoDB read-modify-write latency

As it is visible in the pictures above MongoDB scores lower average latency values in each of the investigated categories. The results range from around 7 to almost 20 times lower latencies of non-relational database compared to traditional RDBMS. MongoDB scores an astonishing 445µs average latency for update operation while its competitor struggles to go below 8650µs latency per transaction. What is interesting are also the values for read-modify-write latency where MongoDB characterizes with relatively very high maximum latency. Read operation results also introduce a nuance where maximum latencies for both databases are higher than it was observed in the previous experiments.

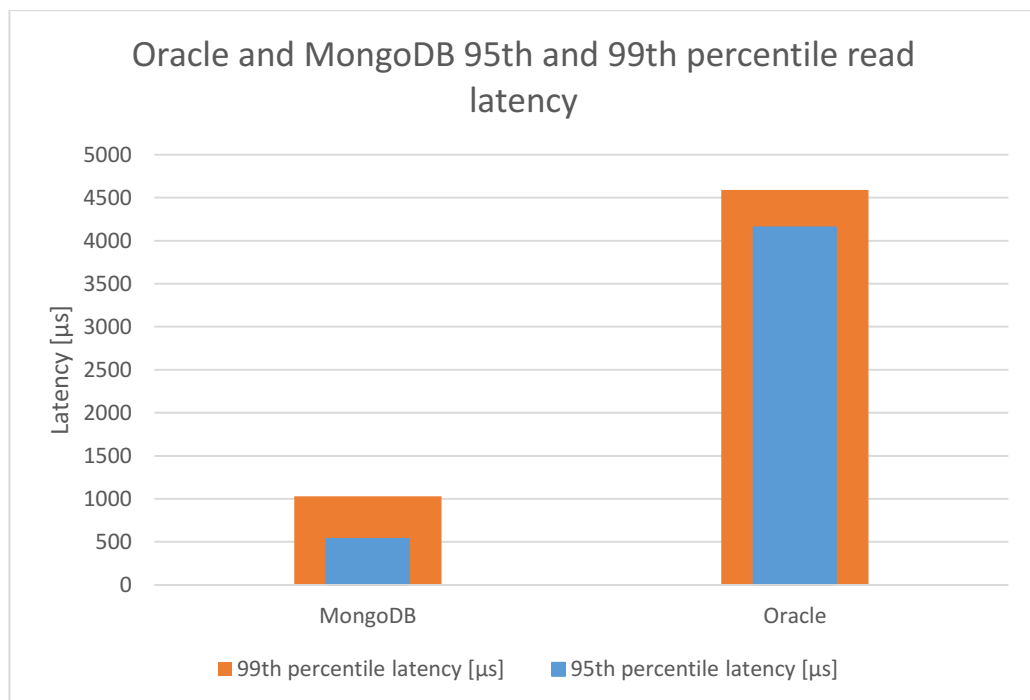


Figure 29 Workload F - Oracle and MongoDB 95th and 99th percentile read latency

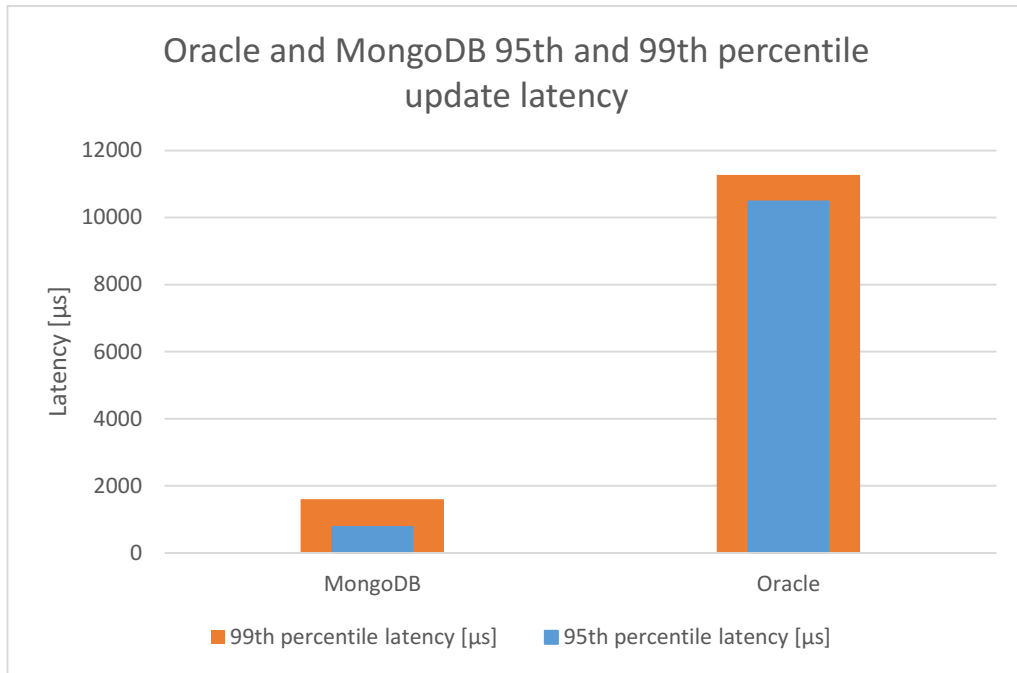


Figure 30 Workload F - Oracle and MongoDB 95th and 99th percentile update latency

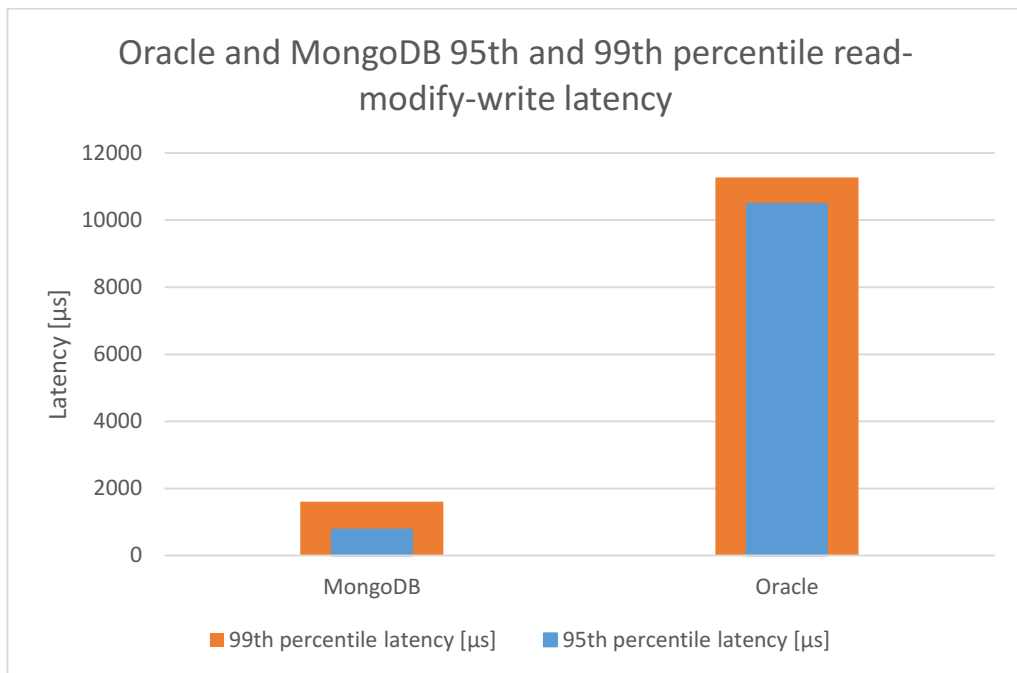


Figure 31 Workload F - Oracle and MongoDB 95th and 99th percentile read-modify-write latency

Visualizing the results for 95th and 99th percentile latencies for all operation types in workload F quickly gives us information on the stability of the registered latencies. It turns out that the values for MongoDB's maximum latency in Figure 28 were incidental and didn't affect the proportion between 95th and 99th percentile latencies visible in Figure 31. The ratio between 95th and 99th percentile latencies in all three above figures for each database does not incline for any observation that has not already been made when investigating previous workloads. Although, it is worth to mention that the extreme distribution between the average latency values visible in Figure 26, Figure 27 and Figure 28 is not visible with 95th and 99th percentiles.

9.7 Cross-workload analysis

Given that each of the workflows characterizes with different operation type distribution, multiple performance aspects of each database engine can be observed when executing them. It is easy to catch similarities between above-mentioned workloads not only in operation proportions but also transaction types. That proves the chance of receiving interesting results and justifies the sense of cross-workload analysis.

The first observation that comes to mind is the comparison between workloads B, C and D. The distribution of operation proportions allows for an analysis of read times and the influence of update and insert operations on such. Workload C will serve as an example as it contains only read operations. On average, MongoDB scored 687 μ s latency time per operation while for Oracle the same characteristic was evaluated as 2856 μ s. The values recorded for workloads B and D are presented in Table 12. What strikes the attention is the big difference between operations times for each workload registered for MongoDB. The average operation times are as much as 60% higher with workload B and 30% lower with workload D. The values were calculated in comparison with the base workload C. The observation for MongoDB allows OracleDB to present much higher stability, again proving that the engine, however slower than competitor, is better adjusted to switching between operation types as the registered values are more consistent. Another obvious observation here are the latencies for both databases and the fact that the numbers for workload B and D are with one exception lower than in the base workload. This might indicate some optimizations in both database engines. It is evident that both databases were not designed with single operation type in mind but rather optimized for interchangeable create, read, update, delete transactions. This is especially visible in workload D where both databases were, on average, performing reads faster than in read-only scenario. Workload B, however, indicated the impact of interchanging update and read operations in MongoDB. The operations were significantly slower than with reads alone.

Table 12 Average read latencies per workload per database

	MongoDB	OracleDB
Workload C - Reads	687.11 μ s	2856.57 μ s
Workload B - Reads	1095.04 μ s	2756.91 μ s
Workload D - Reads	487.1 μ s	2135.34 μ s

Table 13 Average operation latencies for workload D and E

	MongoDB	OracleDB
Workload D - Reads	487.1 μ s	2135.34 μ s
Workload E – Scans	988.57 μ s	3357.46 μ s

The data obtained in workloads D and E tempts for and analysis of these scenarios. Workload D was defined with a read-heavy operation proportion with inserts constituting of only 5% of total transaction count. The balance is similar to Workload E where the read operations were substituted with scans. The comparison of collected data could therefore indicate the differences in both operations' performance. Scans are by design similar to reads in how they fetch data from the collection. A fundamental difference is however the way specific entity is obtained. In order to improve the performance and lower operation latencies, reads take advantage of collection indices. These are not utilized in scans and therefore all data fetches require browsing through the whole collection before the searched value is found. Values for the average latencies registered for workloads D and E are presented in Table 13. The difference between latencies is visible and clear. The first observation is the lower performance of scan

operations. The interesting part is the proportion between read and scan latencies where MongoDB performs the latter around 100% slower than reads. OracleDB is delivering results only 57% slower than read operations taking a lead in the comparison when proportional results are validated. This could outline better efficiency optimizations of collection managing in investigated relational database.

Additional interesting observation is the dominant performance of MongoDB across all workloads despite the different database architectural design trade-offs of each verified database. As mentioned above in chapter 7.2, MongoDB and OracleDB were constructed with different qualities in mind where read efficiency was the predominant quality for MongoDB and write performance to be the driving factor for OracleDB model. These differences are however barely visible when comparing the measured results across defined workloads with MongoDB scoring lower latency scores in a write-heavy workload A where the preferable OracleDB's architecture should be observed. Nevertheless, as expected, workload B with heavy combination of read operations allows MongoDB to prove its performance achieved by the architectural design trade-offs. A conclusion that might be drawn from the results is that Oracle database requires performance configuration in order to represent the full potential of the included options and architectural designs. This is especially visible when verifying various IO models available in the system and described in [82]. Nevertheless, it is expected that the engine's default configuration should be already tuned to perform well in a general case scenario.

Initial review of the related research identified one particularly interesting study [29] that undertook a similar performance comparison between MongoDB and OracleDB. An interesting conclusion from the analysis of the paper is that MongoDB scored significantly lower operation latency times than its competitor. The results observed there in majority differed between investigated databases by a level of magnitude in favor of MongoDB. This confirms the results that were obtained in this study as similarly, MongoDB outperformed OracleDB with far lower latencies.

Cross-workload analyses that were performed above outline the design differences between verified databases. It is evident that verified relational and non-relational engines, besides having common characteristics, significantly differ in operation execution and optimization. OracleDB proves its value in the predictability of the latencies and stability of measured operation times. It is although worth mentioning that MongoDB consistently scored essentially lower latency times and higher throughput values.

10 THREATS TO VALIDITY

This section is to explain the problems that may threaten the validity of the study. These were divided into internal, external and construct validity types.

10.1 Threats to internal validity

In case of the below study, the biggest internal validity threat comes from the selection of the data to calculate average database characteristics. Of multiple open dataset sources available in the Internet, only one was used in this study. The consequence of such limitation was also that analyzed data was representative of a single business domain. The limited choice was caused by the impossibility of analyzing all collectable data in a time span predicted for a Master's Thesis project. Furthermore, an additional limitation for the source choice was the financial cost required to access the presented information. In order to minimize research bias, the study was designed to be easily extendable by incorporating additional data sources in the future. Moreover, a data source of significant size that represents data collected from geographically distributed locations was chosen. Such selection criteria is believed to partially mitigate the risk of choosing unrepresentative data.

Above-described selection of data source has additional consequence in lack of relations between datasets which posed a threat to study validity. The information gathered in Quandl databases was stored in a way that made it impossible to collect metadata on inter-dataset relations. This posed a potential risk of affecting database performance measurements. The issue was mitigated by replicating the database image obtained by measuring Quandl databases' characteristics in a non-relational model with the attention to lack of relationship. NoSQL database model was consistent with relational counterpart in having no connections between data collections. As much as additional information on dataset connections could be valuable to the research and improve study quality, it has been assured that compared database implementations are of identical characteristics. This threat could be addressed in future research.

Selection of the study that was used to determine the method to compare database performance is another example of potential threat to study validity. This applies in particular to the process of narrowing down the number of results of the search query to a quantity that was possible to analyze by the author. Out of the obtained and sorted list of results, twenty documents were selected for further analysis. There is a risk that results that were filtered out of the list contained information that could be relevant to this research. In order to minimize the possibility of excluding results that could affect study, the "sort by relevance" functionality of Summon search engine was used to order the query outcomes. The details of this algorithm are described above in the document. Possible extension of the database measurement study by analysis of additional available studies could further minimize this research validity threat.

An example of additional potential validity threat that could be observed in the research is the possibility to tune each of the verified database systems with additional configuration in order to increase its performance. Both database systems measured in the experiment were running with default preferences and were not optimized additionally. This fact might have a big effect on the performance, especially in the case of the relational databases where measures such as indexes are commonly used to improve engine's performance. Additional configuration tuning was however out of the scope of this research as it would result in significant extension of the research scope beyond the timeline predicted for a master's course. In order to minimize this threat, the

study was designed in order to allow an easy extension by measurements of additional databases, database configurations and, in particular, performance-tuned configurations.

An additional internal validity risk observed in this study was the choice of database characteristics that were used in order to obtain the average database image. Of multiple possibilities for attributes that describe each individual database only a limited number was chosen. Furthermore, there is a risk that the analysis of chosen characteristics is not properly suited for database performance comparison. The purpose for such a limited selection is similar to the above-described threat and can be described in study's time limitations. The risk was mitigated by selection of the database attributes' definitions that were already used in other database performance comparison research. An additional practice taken in order to minimize research bias was designing the study in order to allow easy extension by additional database characteristics.

10.2 Threats to external validity

Another aspect that affects the quality of the study is external validity. A criteria that is especially interesting when examining below study is the repetitiveness of the research in historical perspective. Continuous developments in database area and the constant growth of data generated over the world is a factor that directly affects the ability to define and calculate average database characteristics. This creates a threat to the validity of this research. The risk was mitigated by selecting the data from actively maintained and constantly enlarged data sources. It is believed that the increase in the data growth tendency is proportionally visible in the expansion of open datasets available in the Internet. Future extension of the study should therefore compensate the growth of data over the world by incorporating the analysis of larger data portion. Such activity is possible as the method for database performance comparison defined in this research was defined in a way to allow the extension by any size of collected data.

10.3 Threats to construct validity

One of the values that measures construct validity is the potential for invalidity of the measured scores. This metric is especially visible when analyzing the results of the experiment that were performed in this study. The most significant validity threat visible here is the choice of database performance measurement method and known method flaws. YCSB was chosen as the benchmarking method however it is designed and implemented in Java programming language and requires a middle-layer driver between the application and database. Communicating with the database through a Java driver may affect the measured results. This is considered as a systematic observational error. The thread of obtaining incorrect measurements was mitigated by taking multiple actions the first of which was researching literature for database performance measurement methods. Additionally, specific criteria were defined to discover the method that best fits the below research. Methods were compared by cost, measurement precision, source code availability, and offered metric diversity. The chosen tool was selected with all these attributes in mind. The second action taken in order to minimize research bias was the selection of newest available Java driver for each database. Drivers for OracleDB and MongoDB are both delivered by the owner companies of the products thus it is believed that they are properly optimized in terms of performance.

11 CONCLUSION AND FUTURE WORK

Performance comparison between software databases is an interesting topic beneficial for multiple parties. The activity of comparing databases of different families and data-model structures delivers additional practical value as the results visualize the technological advantages and disadvantages of each solution. The aim of this research was to measure and compare performance of two database management systems. Two representatives were chosen, each from different database family (relational and non-relational). In result of the review on most popular RDBMS and NoSQL database systems, MongoDB and Oracle Database were chosen as examples for each family.

The analysis of benchmarking results states a clear latency and throughput advantage of MongoDB over its competitor. This database system consequently scored lower latency times and higher throughput values in all workload executions. MongoDB turned out to outperform Oracle in read, scan, insert, update and read-modify-write operations which accounts for all tested transaction types. This however does not mean that Oracle does not present any advantages. The benefit of Oracle was better consistency for executed operations. The times for each measured action for that representative were proportionally less distributed and more consistent in 95th and 99th percentiles. It is necessary to emphasize that the conclusions of the above experiment apply only to the direct comparison of two selected databases and only under circumstances that were described in the research.

The byproduct outcome of this thesis is a development of the methodology to measure database performance in a representative scenario. This is believed to be of great benefit for researchers that study this topic. The technique that is worked out here can be applied to any database management system and leads to believe that the results will deliver values that are representative for a given use case. This certainly saves time of research and allows for a quick and meaningful database comparison. The method can be additionally beneficial when deliberating on database characteristics that can affect performance. This was taken into consideration here and was included in the study. As much work as this method required to be developed, it is imperfect and leaves many areas for improvement. The selection of the sources that delivered the data for a calculation of average database characteristic values in this research had to be narrowed down to one data hub. This is undoubtedly a risk that can be addressed by extending the list. Increasing the number of practical examples of database implementations would be of great value to study representativeness. Attention should be put to select a data source that stores information on relations between collections which would greatly increase study quality. Additionally the number of characteristics measured for each database could be extended as it was not possible in the scope of this thesis to address all values that could potentially affect databases' performance. This is especially complicated as some characteristics simply do not exist in various database systems and therefore cannot be measured there.

Researchers could also benefit from the review of database efficiency measurement methods that was performed in chapter 4. This list of tools and techniques can be used to decide on the best approach to verify performance of database engines in any future study. An example of the future work that could extend this thesis could be updating above-mentioned list with the cutting-edge techniques if any new are developed. Incorporation of sources other than scientific reports that define such methodologies could also increase the value of the research.

An interesting development necessary to complete this project is also the review of currently most popular database systems. This certainly delivers research value for

scientists that need to apply their work to databases selected based on popularity criteria. The results of the study allow for the comparison between relational and non-relational database engines. This part of the research could itself be the reason for an analysis of database technology adoption and market distribution between database vendors. Furthermore, the practical value of collecting the data on most popular database systems cannot be underestimated as it could often happen that employees in software and IT domain are interested in the most known and most widely-used database management engines. This part of the research could facilitate business decisions and deliver quantitative business value. The knowledge on databases' popularity could be used as a practical measure of community size and could imply the chance of reaching customer support in case of issues.

The overall results of this thesis, deliver the answer on comparison of two most popular databases from relational and non-relational family. This undoubtedly delivers value for practitioners in IT and especially in the software engineering domain. The outcomes of this study facilitate the decision on the best database choice in a project for architects, product owners, project owners and other software engineering specialists. The architectural design of the product and the discussion on the database engine choice in projects that heavily rely on performance can be a crucial activity for software engineers. The selection of the technologies, including the database management system, affects the entire lifecycle of the product and therefore is a challenging decision. The results presented in this thesis deliver data that could be helpful to take these decisions. Nevertheless, performance is only a single aspect that influences the database choice with purchase cost, maintenance cost, additional offerings, stability or access to source code being examples of other criteria that could impact the selection. Extension of the study with an analysis and performance comparison of different database engines could potentially increase the value of the research even further.

This thesis extends current knowledge in software engineering area by developing a methodology of measuring database performance and comparing two database engines in practice. The information presented here delivers value for researchers and practitioners as it is stated in the examples above. The whole research acts as an extension point with several possible paths for future work.

REFERENCES

- [1] K. L. Berg, T. Seymour and R. Goel, "History of databases," *International Journal of Management & Information Systems*, vol. 17, no. 1, pp. 29-35, 2013.
- [2] S. W. Dietrich, "An Animated Introduction to Relational Databases for Many Majors," *IEEE Transactions on Education*, vol. 58, no. 2, pp. 81-89, 2014.
- [3] C. J. Date, *An Introduction to Database Systems*, 6th ed., Z. Grzejszczak, Ed., Warszawa: Wydawnictwa Naukowo-Techniczne, 1995, pp. 24,31.
- [4] U. Bhat and S. Jadhav, "Moving Towards Non-Relational Databases," *International Journal of Computer Application*, vol. 1, no. 13, pp. 40-46, 2010.
- [5] B. Grad, "Relational Database Management Systems: The Business Explosion [Guest editor's introduction]," *IEEE Annals of the History of Computing*, vol. 35, no. 2, pp. 8-9, 2013.
- [6] J. Celko, *Joe Celko's Complete Guide to NoSQL : What Every SQL Professional Needs to Know about Non-Relational Databases*, 1st ed., Morgan Kaufmann, 2013.
- [7] R. P. Padhy, M. R. Patra and S. C. Satapathy, "RDBMS to NoSQL: Reviewing Some Next-Generation," *International Journal of Advanced Engineering Science and Technologies*, vol. 11, no. 1, pp. 15-30, 2011.
- [8] J. Han, H. E. G. Le and J. Du, "Survey on NoSQL database," in *Int. Conf. on Pervasive Computing and Applications*, Port Elizabeth, 2011.
- [9] N. T. Bhuvan and M. S. Elayidom, "A Technical Insight on the New Generation Databases: NoSQL," *International Journal of Computer Applications*, vol. 121, no. 7, pp. 24-26, 2015.
- [10] J. McKendrick, "Accelerating cloud-enabled enterprise: 2015 IOUG survey on database manageability," Unisphere Research, 2015.
- [11] Gartner, "The High Performance Data Warehouse Bigger or Better, Not Always Both," Gartner, 2010.
- [12] C. de Souza Baptista, C. E. Pires, D. F. Leite and M. G. de Oliveiraa, "NoSQL geographic databases: an overview," in *Geographical Information Systems: Trends and Technologies*, E. Pourabbas, Ed., London, CRC Press, 2014, p. 77.
- [13] S. Yen, "NoSQL is a horseless carriage," 1 October 2009. [Online]. Available: <https://www.yumpu.com/en/document/view/24663547/nosql-steve-yen>. [Accessed 28 February 2016].
- [14] S. Edlich, "List of NoSQL Databases: NoSQL Databases," [Online]. Available: <http://nosql-database.org/>. [Accessed 28 February 2016].
- [15] R. Cattell, "Scalable SQL and NoSQL Data Stores," December 2011. [Online]. Available: <http://www.cattell.net/datastores/Datastores.pdf>. [Accessed 28 February 2016].
- [16] T. Haerder and A. Reuter, "Principles of transaction-oriented database recovery," *ACM Computing Surveys (CSUR)*, vol. 15, no. 4, pp. 287-317, 1983.
- [17] D. Pritchett, "BASE: An Acid Alternative," *Queue - Object-Relational Mapping*, vol. 6, no. 3, pp. 48-55, 2008.
- [18] S. Gilbert and N. Lynch, "Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services," *ACM SIGACT News*, vol. 33, no. 2, pp. 51-59, 2002.
- [19] Blekinge Institute of Technology, "Summon@BTH," SerialsSolutions, [Online]. Available: <http://bth.summon.serialssolutions.com/>.
- [20] Google LLC, "Google Scholar," [Online]. Available: <https://scholar.google.com>.

- [21] V. Abramova, J. Bernardino and P. Furtado, "SQL or NoSQL? Performance and scalability," *International Journal of Business Process Integration and Management*, vol. 7, no. 4, pp. 314-321, 2015.
- [22] B. F. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan and R. Sears, "Benchmarking Cloud Serving Systems with YCSB," Santa Clara, 2010.
- [23] B. G. Tudorica, "A comparison between several NoSQL databases with comments and notes," in *Roedunet Int. Conf.*, Iasi, 2011.
- [24] A. Floratou, N. Teletia, D. J. DeWitt, J. M. Patel and D. Zhang, "Can the Elephants Handle the NoSQL Onslaught?," *Proceedings of the VLDB Endowment*, vol. 5, no. 12, pp. 1712-1723, 2012.
- [25] P. G. Prateek Nepaliya, "Performance Analysis of NoSQL Databases," *International Journal of Computer Applications*, vol. 127, no. 12, pp. 36-39, 2015.
- [26] R. Henricsson, "Document Oriented NoSQL Databases: A comparison of performance in," Blekinge Institute of Technology, Karlskrona, 2011.
- [27] C. Hadjigeorgiou, "RDBMS vs NoSQL: Performance and Scaling," 2013.
- [28] Y. Li and S. Manoharan, "A performance comparison of SQL and NoSQL databases," in *IEEE Pacific Rim Conf. on Communications, Computers and Signal Processing*, Victoria, 2013.
- [29] A. Boicea, "MongoDb vs Oracle - Database comparison," in *Int. Conf. on Emerging Intelligent Data and Web Technologies*, Bucharest, 2012.
- [30] A. Gandini, M. Gribaudo, W. J. Knottenbelt, R. Osman and P. Piazzolla, "Performance Evaluation of NoSQL Databases," in *European Workshop on Computer Performance Engineering*, Florence, 2014.
- [31] J. S. van der Veen and R. J. Meijer, "Sensor Data Storage performance: SQL or NoSQL, Physical or Virtual," in *Int. Conf. on Cloud Computing*, Honolulu, 2012.
- [32] S. Schmid, "WMS performance of selected SQL and NoSQL databases," in *Int. Conf. on Military Technologies*, Brno, 2015.
- [33] Z. Parker, S. Poe and S. V. Vrbsky, "Comparing NoSQL MongoDB to an SQL," in *ACM Southeast Conference*, Savannah, 2013.
- [34] R. Escriva, B. Wong and E. G. Sirer, "HyperDex: a distributed, searchable key-value store," *ACM SIGCOMM Computer Communication Review - Special october issue*, vol. 42, no. 4, pp. 25-36, October 2012.
- [35] J. Kuhlenkamp, M. Klems and O. Ross, "Benchmarking scalability and elasticity of distributed database systems," *Proceedings of the VLDB Endowment*, pp. 1219-1230, 2014.
- [36] Y. Abubakar, S. T. Adeyi and I. Gambo Auta, "Performance Evaluation of NoSQL Systems Using YCSB in a resource Austere Environment," *Applied Information Systems, International Journal of*, vol. 7, no. 8, pp. 23-27, 2014.
- [37] "Google," Google Inc., [Online]. Available: <https://www.google.com/>. [Accessed 8 January 2016].
- [38] "Desktop Search Engine Market Share: Market Share Statistics for Internet Technologies," NETMARKETSHARE, [Online]. Available: <https://www.netmarketshare.com/search-engine-market-share.aspx?qprid=4&qpcustomd=0>. [Accessed 8 January 2016].
- [39] "Search operators: Google Support," Google Inc., [Online]. Available: <https://support.google.com/websearch/answer/2466433?hl=en>. [Accessed 8 January 2016].
- [40] The Open Government Working Group, "Open Government Data," [Online]. Available: <http://opengovernmentdata.org/>. [Accessed 20 March 2016].

- [41] DataPortals.org, "DataPortals.org," [Online]. Available: <http://dataportals.org/>. [Accessed 20 March 2016].
- [42] B. Kitchenham, "Procedures for Performing Systematic Reviews, Joint Technical Report," Keele University, Keele, 2004.
- [43] ProQuest, "Relevance Ranking in the Summon Service," 15 August 2013. [Online]. Available: <http://media2.proquest.com/documents/Summon-RelevanceRanking-Datasheet.pdf>. [Accessed 10 March 2018].
- [44] C. Bornhoevd and P. Guerrero, "Systems and methods for repeatable database performance". United States of America Patent 7 403 954, 22 July 2008.
- [45] D. J. Shee, "Database performance monitoring method and tool". United States of America Patent 6 910 036, 21 June 2005.
- [46] T. Teorey, "Dependability and performance measures for the database practitioner," *IEEE Transactions on Knowledge and Data Engineering*, vol. 10, no. 3, pp. 499-503, 2002.
- [47] J. Gray, "A view of database system performance measures," in *ACM SIGMETRICS conference*, New York, 1987.
- [48] A. K. Dvivedi, "Performance Analysis of Column Oriented Database Vs Row Oriented," *Int. Journal of Computer Applications*, vol. 50, no. 14, pp. 31-34, 2012.
- [49] R. W. Olshavsky, A. B. Aylesworth and D. S. Kempf, "The price-choice relationship: A contingent processing approach," *Journal of Business Research*, vol. 33, no. 3, pp. 207-218, 1995.
- [50] DB-Engines, "DB-Engines Ranking," February 2017. [Online]. Available: <http://db-engines.com/en/ranking>. [Accessed 23 February 2017].
- [51] EverSQL Team, "EverSQL, Online SQL Query Optimization, Easily, Automatically," [Online]. Available: <https://www.eversql.com/most-popular-databases-in-2017-according-to-stackoverflow-survey>. [Accessed 15 August 2017].
- [52] F. Stroud, "ServerWatch," [Online]. Available: <https://www.serverwatch.com/server-trends/slideshows/top-10-enterprise-database-systems-to-consider-2015.html>. [Accessed 15 August 2017].
- [53] DB-Engines, "Meghod of calculating the scores of the DB-Engines ranking," [Online]. Available: http://db-engines.com/en/ranking_definition. [Accessed 23 February 2017].
- [54] A. van den Bosch, T. Bogers and M. de Kunder, "Estimating search engine index size variability: a 9-year longitudinal study," *Scientometrics*, vol. 107, no. 2, pp. 839-856, 2015.
- [55] Alexa Internet Inc., "Alexa Internet - About us," [Online]. Available: <http://www.alexa.com/about>. [Accessed 23 February 2017].
- [56] Alexa Internet Inc., "Alexa Top Sites by category: Computers/Internet/Searching/Search Engines," [Online]. Available: http://www.alexa.com/topsites/category/Computers/Internet/Searching/Search_Engines. [Accessed 23 February 2017].
- [57] NetMarketShare - Market Share Statistics for Internet Technologies, "Search engine market share," [Online]. Available: <https://www.netmarketshare.com/search-engine-market-share.aspx?qprid=4&qpcustomd=0>. [Accessed 23 February 2017].
- [58] P. Sarathi, "A deep dive into NoSQL: A complete list of NoSQL databases," Big Data Made Simple, 21 July 2014. [Online]. Available: <http://bigdata-madesimple.com/a-deep-dive-into-nosql-a-complete-list-of-nosql-databases/>. [Accessed 21 June 2016].

- [59] Wikipedia Org., "List of relational database systems," [Online]. Available: https://en.wikipedia.org/wiki/List_of_relational_database_management_systems. [Accessed 23 February 2017].
- [60] C. Bryant, "A Guide to Open Source Cloud Computing Software," tom's IT PRO, 12 June 2014. [Online]. Available: <http://www.tomsitpro.com/articles/open-source-cloud-computing-software,2-754-8.html>. [Accessed 21 June 2016].
- [61] M. Vilas, "Using Google Search from your Python code," 29 February 2016. [Online]. Available: <https://breakingcode.wordpress.com/2010/06/29/google-search-python/>. [Accessed 22 June 2016].
- [62] K. Kolonko, "GitHub - google-fetcher repository," [Online]. Available: <https://github.com/KamilKolonko/google-fetcher>.
- [63] L. D. Shapiro and G. Graefe, "Data Compression and Database Performance," in *Symposium on Applied Computing*, Kansas City, 1991.
- [64] C. S. Mullins, "Database Performance Design," in *Database Administration: The Complete Guide to DBA Practices and Procedures*, 2nd ed., Addison-Wesley Professional, 2012.
- [65] Z. A. Al-Khanjari and S. Al-Kindy, "Metadata Extraction in Database Testing," *Information Management and Business Review*, vol. V, no. 3, pp. 108-112, 2013.
- [66] Oracle, "Describing Database Metadata," Oracle, [Online]. Available: https://docs.oracle.com/cd/B10501_01/appdev.920/a96583/cci06met.htm#1000847. [Accessed 1 March 2017].
- [67] MySQL, MySQL Reference Manual : INFORMATION_SCHEMA Tables, [Online]. Available: <https://dev.mysql.com/doc/refman/5.7/en/information-schema.html>. [Accessed 1 March 2017].
- [68] Microsoft Technet, "Querying the SQL Server System Catalog," [Online]. Available: [https://technet.microsoft.com/en-us/library/ms189082\(v=sql.105\).aspx](https://technet.microsoft.com/en-us/library/ms189082(v=sql.105).aspx). [Accessed 1 March 2017].
- [69] J. Gantz and D. Reinsel, "The Digital Universe in 2020," International Data Corporation, 2013.
- [70] Datahub.io, "Datahub," Datahub.io, [Online]. Available: <https://datahub.io/>. [Accessed 20 August 2016].
- [71] Quandl, "Quandl," Quandl, [Online]. Available: <https://www.quandl.com/>. [Accessed 20 August 2016].
- [72] Open Data Network, "Open Data Network," [Online]. Available: <https://www.opendatanetwork.com/>. [Accessed 12 March 2017].
- [73] IOGDS: International Open Government Dataset Search, "IOGDS: International Open Government Dataset Search," IOGDS: International Open Government Dataset Search, [Online]. Available: <https://logd.tw.rpi.edu/>. [Accessed 20 August 2016].
- [74] Quandl, "Free data on Quandl," [Online]. Available: <https://blog.quandl.com/free-data-on-quandl>. [Accessed 12 March 2017].
- [75] Quandl, "Change to Quandl API limits," 22 November 2016. [Online]. Available: <https://blog.quandl.com/change-quandl-api-limits>. [Accessed 18 March 2017].
- [76] M. Hamburg and P. Young, "Statistical Investigations and Suvey Sampling," in *Statistical Analysis for Decision Making*, 6th ed., 1994, pp. 183-212.
- [77] Changing Minds, "Choosing a sampling method," Changing Minds, [Online]. Available: http://changingminds.org/explanations/research/sampling/choosing_sampling.htm. [Accessed 12 March 2017].
- [78] K. Kolonko, "List of investigated databases," [Online]. Available: <https://1drv.ms/b/s!As3U9B9XHy-j-Wxq6Ud5Ld4-4b3p>.

- [79] K. Kolonko, “Raw Quandl Results,” [Online]. Available: https://1drv.ms/x/s!As3U9B9XHy-j-W0DLpYpyGZ4X2_m.
- [80] I. MongoDB, “RDBMS to MongoDB Migration Guide - Considerations and Best Practices,” MongoDB, Inc., 2016.
- [81] E. Dede, M. Govindaraju, D. Gunter, R. Shane Canon and L. Ramakrishnan, “Performance Evaluation of a MongoDB and Hadoop Platform for Scientific Data Analysis,” in *4th ACM workshop on Scientific cloud computing, Proceedings of the*, New York, 2013.
- [82] G. Harrison, Oracle Performance Survival Guide: A Systematic Approach to Database Optimization. Part VI. IO Tuning and Clustering, 1st ed., Prentice Hall, 2009.
- [83] R. Mallet, J. Hagen-Zenker, R. Slater and M. Duvandack, “The benefits and challenges of using systematic reviews in international development research,” *Journal of Development Effectiveness*, vol. 11, no. 9, pp. 445-455, 2012.
- [86] DataPortals.org, “DataPortals,” DataPortals.org, [Online]. Available: <http://dataportals.org/>. [Accessed 20 August 2016].
- [87] A. Fox, S. D. Gribble, Y. Chawathe, E. A. Brewer and P. Gauthier, “Cluster-based scalable network services,” in *ACM symposium on Operating systems principles, Proceedings of the sixteenth*, Saint Malo, 1997.

ATTACHMENT 1. LIST OF DATABASE SEARCH QUERIES AND RESULTS

Query	Number of results	Query	Number of results
Oracle database	9430000	ArangoDB database	1050
MySQL database	643000	Starcounter database	1050
SQL Server database	535000	BaseX database	1010
Microsoft Access database	480000	Recutils database	952
SQLite database	459000	Adabas D database	947
PostgreSQL database	452000	Openbase database	942
MongoDB database	167000	VaultDB database	939
IBM DB2 database	156000	AllegroGraph database	889
Teradata database	140000	RDM Server database	888
H2 database	127000	LSM database	887
SAP HANA database	115000	Trafodion database	881
Greenplum database	110000	SmallSQL database	875
Informix database	97800	eXtremeDB Financial Edition database	866
Firebird database	94500	solidDB database	859
SQL Azure database	74800	Sedna database	845
Oracle NOSQL Database	67500	ObjectDB database	827
FileMaker Pro database	60700	CodernityDB database	825
MariaDB database	57700	R%3ABase database	824
Cassandra database	54700	Druid database	820
mSQL database	48200	IBM DB2 Express-C database	800
SQL Anywhere database	48200	Model 204 Database	796
dBase database	44800	Cloud Datastore database	781
Visual FoxPro database	40300	RocksDB database	751
Redis database	38500	CSQL database	718
InterBase database	36700	Clustrix database	701
Hadoop database	34100	Lintier database	696
UniVerse database	30500	DBreeze database	682
Database Management Library	28600	Cloudera database	674
Apache Derby database	28100	Scylla database	640
IBM Informix database	27600	ClickHouse database	621
Tarantool database / Box database	26360	MapR database	619
Neo4J database	26300	Onyx database	594
BM Lotus database / Domino database	26200	NDatabase database	588
Ingres database	25600	eXtremeDB database	568
Netezza database	20600	Voldemort database	561
HSQldb database	20500	FileDB database	521
Vertica databas	20300	EyeDB database	515

CouchDB database	20200	NeDB database	499
MaxDB database	17900	Splice Machine database	499
eXist database	12700	KirbyBase database	478
Berkeley DB database	11500	SequoiaDB database	472
Serenity database	9880	siaqodb database	453
HBase database	9100	HPCC database	433
RDM Embedded database	8830	SciDB database	432
SAP Sybase database	8150	OpenQM database	387
Mnesia database / ErlangDB database	7640	PicoLisp database	356
TimesTen database	7580	EXASolution database	355
db4o database	7280	KAI database	337
Btrieve database	7240	SAND CDBMS database	334
Reality database	6300	BoltDB database	327
Couchbase Server database	6160	EMC Documentum xDB database	299
CA IDMS database	6090	MemcacheDB database	289
Panorama database	6040	TiDB database	264
Sybase Advantage Database Server	5860	Tokutek database	259
Sophia database	5540	Oracle Coherence database	241
Eventsourcing for Java database	5230	Hortonworks database	238
Amazon Aurora database	5190	Dataphor database	229
OpenLink Virtuoso database	5170	Apache Flink database / Stratosphere database	227
kdbB database	4940	illuminate databas	226
Watcom SQL database	4710	STSdb database	220
Sterling database	4580	Hazelcast database	207
DynamoDB database	4550	Symas LMDB database	192
CUBRID database	4520	VelocityDB database	163
Clarion database	4500	Scalaris database	139
Elasticsearch database	4460	TigerLogic PICK database	138
4th Dimension database	4390	Infinispan database	137
HSS database	4270	Derby aka Java DB database	136
OpenLDAP database	4270	ToroDB database	129
RethinkDB database	4220	acid-state database	122
TITAN database	4130	KUDU database	104
Intersystems Cache database	4100	Qizx database	102
FoundationDB database	4000	GroveSite database	77
Bigdata database	3990	quasardb database	77
Globals database	3880	Prevayler database	75
UniData database	3700	GenieDB database	74
U2 database	3680	InfoGrid database	72
GemStone database	3660	KitaroDB database	67
Helix database	3650	allegro-C database	66
Alpha Five database	3610	HyperDex database	66

Progress Software database	3440	pipelinedb database	57
Riak database	3440	NCache database	46
Aerospike database	3390	FramerD database	43
Berkeley DB XML database	3390	GraphBase database	41
Amazon SimpleDB database	3320	GigaSpaces database	38
Datomic database	3290	BangDB database	34
LevelDB database	3150	FlockDB database	34
RavenDB database	3090	Meronymy database	27
OrientDB database	3010	WhiteDB database	23
OrientDB database	2990	BrightstarDB database	10
OrientDB database	2990	BrightstarDB database	10
RDM Embedded database	2890	The SAS system database	10
NonStop SQL database	2880	BergDB database	9
ScimoreDB database	2860	Dynomite database	9
IBM Lotus Approach database	2800	Execom IOG database	9
SQLBase database	2720	Oracle Rdb for OpenVMS database	9
OpenLink Virtuoso database	2710	WonderDB database	9
Sybase Adaptive Server Anywhere database	2680	Clusterpoint Server database	8
Hypertable database	2650	RaptorDB database	8
SDB database	2650	GPUdb database	7
Google Fusion Tables database	2640	Infinite Graph database	7
EJDB database	2630	JasDB database	7
Tokyo Cabinet database	2580	Unisys RDMS database	7
MarkLogic Server database	2550	upscaledb database	7
InterSystems Cache9 database	2540	RaptorDB database	6
Trinity database	2460	Vyhodb database	6
Pervasive PSQL database	2380	djondb database	5
Mimer SQL database	2290	iBoxDB database	5
eXtremeDB database	2220	Moonshadow database	5
GT.M database	2210	Yserial database	5
LibreOffice Base database	2180	Axibase database	4
MonetDB database	2120	CortexDB database	4
NexusDB database	2090	Crate Data database	4
Postgres Plus Advanced Server database	2020	Pincaster database	4
Tibero database	2010	Scalien database	4
MonetDB database	1960	ThruDB database	4
Magma database	1930	BayesDB database	3
OpenOfficeorg Base database	1930	CoreObject database	3
JADE database	1770	NoSQL embedded db database	3
NEO database	1740	SisoDB database	3
Accumulo database	1720	Genomu database	2
MemSQL database	1720	LightCloud database	2
weaver database	1710	PickleDB database	2

Versant database	1700	Queplich database	2
Empress Embedded database	1660	AlchemyDB database	1
Event Store database	1660	AmisaDB database	1
Azure Table Storage database	1650	Chordless database	1
OpenInsight database	1610	Cloudata database	1
Aster Data database	1580	densodb database	1
Faircom C-Tree database	1580	influxdata database	1
NuoDB database	1580	JSON ODM database	1
Infobright database	1570	Maxtable database	1
HyperGraphDB database	1550	nessDB database	1
Hibari database	1540	Ninja Database Pro database	1
ZODB database	1490	SharedHashFile database	1
Perst database	1460	TIBCO Active Spaces database	1
Objectivity database	1410	Tieto TRIP database	1
EnterpriseDB database	1400	BinaryRage database	0
GemFire database	1380	Chronicle Map database	0
rasdaman database	1340	ConcourseDB database	0
jBASE database	1320	DaggerDB database	0
ESENT database	1290	Elliptics database	0
IBM Cloudant database	1290	es4j database	0
Vectorwise database	1240	Fallen 8 database	0
DataEase database	1190	gunDB database	0
CA Datacom database	1140	gunDB database	0
Sparksee database	1130	gunDB database	0
Tyrant database	1110	MarcelloDB database	0
FrontBase database	1100	MiniM DB database	0
Polyhedra database	1100	Morantex database	0
Altibase database	1080	RockallDB database	0
Starcounter database	1060	SQream DB database	0
Tarantool database	1060	TazyGrid database	0
txtSQL database	1060	Terrastore database	0
ArangoDB database	1050	TreodeDB database	0
ArangoDB database	1050	VertexDB database	0