Performance Evaluation of Query Response Time in The Document Stored NoSQL Database

Rohmat Gunawan
Department of Informatics
Siliwangi University
Tasikmalaya, Indonesia
rohmatgunawan@unsil.ac.id

Alam Rahmatulloh
Department of Informatics
Siliwangi University
Tasikmalaya, Indonesia
alam@unsil.ac.id

Irfan Darmawan
Department of Information System
Telkom University
Bandung, Indonesia
irfandarmawan@telkomuniversity.ac.id

Abstract*—NoSQL Database is one solution that can be used to store large amounts of data and have an excellent performance during the reading and writing process. Analysis of data with large volumes increasingly feels the benefits. Database exploration is one of the activities carried out in data analysis. Information on estimated response time queries can help in planning database exploration. This study aims to evaluate the response time of each query for the process of creating, read, update, delete (CRUD) on a document stored, NoSQL Database. MongoDB, ArangoDB, and CouchDB were selected for use in experiments. Experiments are carried out by processing queries for repeated create, read, update, and delete commands with different quantities. The experimental results in the study revealed that MongoDB had the lowest average query response time for the read process (0.017 seconds), update (25,358 seconds) and delete (0.055 seconds) compared to ArangoDB and CouchDB. However, for the creation process, the ArangoDB response time is the smallest, which is 28,493 seconds compared to MongoDB and CouchDB.

Keywords—Query, Response Time, Document Stored

I. INTRODUCTION

The rapid development of the internet and cloud computing has encouraged the availability of databases to be able to effectively store and process extensive data and demand high performance when reading and writing. NoSQL Database is one solution that can be used to store large amounts of data [1], [2], [3] and fast during the reading and writing process [4]. NoSQL databases can work faster than relational databases [5].

Analysis of data with large volumes increasingly feels the benefits. Query exploration is one of the activities carried out in the data analysis process [6]. Query optimization for creating and read operations and lower functionality support for update and delete queries is a single NoSQL database feature [7].

NoSQL Databases are designed for scalability and are more efficient when certain data models are used. A document stored is one type of NoSQL database that is currently available [8], [9], [10]. Create, Read, Update and Delete (CRUD) is a common operation performed on NoSQL Databases [11], [12].

Measurement of NoSQL performance is one of the exciting topics to be studied in research. Several studies related to speed measurement, performance comparisons have been carried out in previous studies, including measuring the performance of asynchronous NoSQL database replication [7], comparison of NoSQL performance in thread usage [13], MongoDB and Cassandra performance measurements using Yahoo Cloud Service Benchmark (YCSB) [11].

The purpose of this study is to conduct performance measurements on a document stored, NoSQL Database CRUD processes. Data on NoSQL document stored is stored in the form of JavaScript Object Notation (JSON) or eXtensible Markup Language (XML) [14]. MongoDB, ArangoDB, and CouchDB were selected for use in experiments. The experiment is done by processing several CRUD queries on each selected NoSQL database. Each trial result is recorded and presented in graphical form. At the final stage, conclusions were drawn from all the experiments that had been carried out.

II. RELATED WORK

Research conducted by [7], trying to measure the synchronous replication performance of three document stored database based databases: MongoDB, CouchDB, CouchBase. The execution time of a CRUD operation is measured, and then averages are calculated. The results of his research show that CouchDB has a perfect overall performance time for insert, update, and delete operations. As for the reading process, MongoDB is the fastest.

Research conducted by [11], try comparing the performance of the NoSQL database: MongoDB and Cassandra. Performance measurement is done by using the Yahoo Cloud Service Benchmark (YCSB). The results showed that Cassandra's execution time for calculating various information operations was better than that of MongoDB. For overall runtime and throughput, Cassandra's performance is better than MongoDB. During the data update process, Cassandra is 75% faster than MongoDB. Whereas in the data reading process, Cassandra is 70% faster.

In research [13], performance comparisons from three NoSQL databases (Couchbase, MongoDB, RethinkDB). The experiments were carried out in two different scenarios: single thread and multiple threads. The experimental results in the study revealed that Couchbase performed better in most operations, except for taking lots

of documents and entering documents with many threads, MongoDB got a better score.

In the study [15], trying to measure the response times on the query Create Read Delete Update (CRUD) using MongoDB. The experimental results in the study show that the response time for querying read data on MongoDB is the fastest compared to the response time query for the create, update, and delete processes.

Measuring the response time of each query in the NoSQL CRUD process Database document stored based is the main focus of this study. MongoDB, ArangoDB, and CouchDB were selected for use in experiments. NoSQL Database is installed on a PC Server that is connected using wireless media with NoSQL Client. Experiments are carried out by processing queries for repeated create, read, update, and delete commands with different quantities. Each trial result is recorded in a table and presented in graphical form. At the final stage, conclusions are drawn from all the experiments that have been carried out.

III. EXPERIMENTAL SETUP

There are four stages carried out in this study, namely: system analysis, identification of hardware & software requirements, design, implementation, and measurement.

A. System Analysis

At this stage, an analysis of the general scope of the system will be carried out. The measurement of query response time involves software, hardware, and data used when executing queries in a NoSQL database.

B. Identification of Hardware and Software Needs

At this stage, identification of hardware and software is needed in the process of measuring response time queries. The hardware specifications used are shown in Table 1.

TABLE 1. HARDWARE SPECIFICATIONS USED

No	Item	Description
1	Client	- Intel Processor Core i3-2328 M dual-core @ 2.20 GHz (4 CPU) - 4 GB Memory - Intel HD Graphics 3000
2	Server	- Intel Processor Core i5-4460T @ 1.90 GHz (4 CPU) - 4 GB Memory - Graphics AMD R7 A360 GPU
3	Access Point	TP-Link TL-WR841N

Apart from hardware, some software is also needed in the experiments conducted in this study. The software specifications needed in the experiment are shown in Table 2.

TABLE 2. SOFTWARE SPECIFICATIONS USED

No	Software	Version
1	Microsoft Windows (installed in Server)	10
2	Microsoft Windows (installed in Client)	8.1
3	MongoDB	3.4
4	Robo3T (MongoDB GUI Client)	3.0
5	ArangoDB	3.2.3
6	CouchDB	2.3.0

C. Designing

At this stage, technical planning is carried out at the experimental stage. Some things are prepared before the technical measurement process is carried out:

1. Data Preparation
   The data used as experiments in this study are of type string. The data entity entered is the databook with attributes: title, author, paperback, publisher, language.
2. Experimental Design
   The experiment was carried out by repeating multiple queries: 1,000, 2,000, 3,000, 4,000, 5,000. The query is done for the create, read, update, and delete (CRUD) commands.
3. Measurement
   Every query that is carried out is calculated the response time generated. The results of the experimental data are then inputted into a table and presented in graphical form.

D. Implementation and Measurement

At this stage, the NoSQL Database Server is installed on the prepared hardware. Access points are configured in the Dynamic Host Configuration Protocol (DHCP) mode so that the Server and Client get the IP Address in DHCP mode. After the Server and Client are connected, then configure the database connection on the client to connect with the NoSQL Database Server, and the trial process can be done.

IV. EXPERIMENTAL RESULT

A. Query Preparation

The initial stage starts with making a query for each CRUD process. This stage is done to ensure the query syntax is suitable for each NoSQL database that will be used. The query syntax for the creating process for MongoDB and ArangoDB is shown in Figure 1.

```
// Query for Create Data with MongoDB
for (var i = 1; i <= 1000; i++){
  db.book.insert({
  Title : "NoSQL Distilled: A Brief Guide to the
          Emerging World of Polyglot Persistence",
  Author: "Pramod J. Sadalage; Martin Fowloer",
  Paperback : "192 pages",
  Publisher : "Addison-Wesley Professional;
              1 edition (August 18, 2012)",
  Language  : "English"
 })
}
// Query for Create Data with ArangoDB
FOR i IN 1..1000 INSERT {
  Title : "NoSQL Distilled: A Brief Guide
             to the Emerging World of
             Polyglot Persistence",
  Author: " Pramod J. Sadalage; Martin
            Fowloer",
  Paperback : "192 pages",
  Publisher : "Addison-Wesley Professional;
              1 edition (August 18, 2012)",
  Language  : "English"
} IN book
```

Figure 1. The query syntax for creating data on MongoDB and ArangoDB

```
// Query for Create Data with CouchDB
for /L %%i IN (1,1,1000) do
 ( curl -X PUT
   "http://192.168.0.100:5984/book/%%i" -d
   "{ Title : "NoSQL Distilled: A Brief Guide
  to the Emerging World of Polyglot
  Persistence", Author: " Pramod J. Sadalage
  ; Martin  Fowloer", Paperback : "192
pages",  Publisher : "Addison-Wesley
Professional;  1 edition (August 18, 2012)",
  Language  : "English" )
```

Figure 2. The query syntax for creating data CouchDB

In Figure 2, the query syntax of the create or insert process is displayed to be applied to CouchDB. The first experiment is repeated 1000 times using the for-do syntax. The next query is for the data reading process, as shown in Figure 3.

```
// Query for Read Data with MongoDB
db.buku.find({})

// Query for reading Data with ArangoDB
FOR i IN book RETURN i

// Query for reading Data with CouchDB
curl -X GET
http://192.168.0.100:5984/book/_all_docs
```

Figure 3. The query syntax for read data

In Figure 3 the query syntax is displayed for the read process or displaying data. The command to display data without parameters as shown in Figure 3 will display all collections contained in the database.

```
// Query for Update Data with MongoDB
for (var i = 1; i <= 1000; i++)
db.book.update({ Language  : "English" },
{ Title     : "NoSQL Distilled: A Brief
Guide to the Emerging World of Polyglot
Persistence",  Author    : "Pramod J.
Sadalage ; Martin Fowloer", Paperback : "192
pages",  Publisher : "Addison-Wesley
Professional; 1 edition (August 18, 2012)",
Language  : "English", Price     : "32.90" }

// Query for Update Data with ArangoDB
FOR doc IN book UPDATE doc WITH {Price :
"32.90" } IN book

// Query for Update Data with CouchDB
for /L %%i IN (1,1,1000) do (curl -X PUT
"http://192.168.0.100:5984/book/%%i" -d
"{""id"":%%i,
""_rev"":""1e52d722ce163c2fe367ba8738966bb26
"", ""Title""     : ""NoSQL Distilled: A
Brief Guide to the Emerging World of
Polyglot Persistence"", ""Author"": ""Pramod
J. Sadalage ; Martin Fowloer"",""Paperback""
: ""192 pages"",""Publisher"" : ""Addison-
Wesley Professional; 1 edition (August 18,
2012)"",""Language""  :
""English"",""Price"":""32.90""}")
```

Figure 4. The query syntax for Update data

In Figure 4, the query syntax for the data update process is displayed. The data update is done by adding the attribute "Price" with the value "32.90". The last query is prepared for the data delete process. All data or collections in the database will be deleted. In general, the query syntax for the data delete process is shown in Figure 5.

```
// Query for Delete Data with MongoDB
db.book.remove({})

// Query for Delete Data with ArangoDB
FOR i IN book
  REMOVE i IN book
  LET removed = OLD
  RETURN removed._key

// Query for Delete Data with CouchDB
for /L %%i IN (1,1,5000) do (curl -X DELETE
"http://192.168.0.100:5984/book/%%i?rev=1-
e52d722ce163c2fe367ba8738966bb26")
```

Figure 5. The query syntax for delete data

In Figure 5, the query syntax for the data delete process is displayed. The command to delete data without parameters, as shown in Figure 6, will cause the entire collection to be deleted in the database to be deleted after the query has been executed.

B. Query Execution

The first experiment is done using MongoDB. Query execution is done using Robo3T, as shown in Figure 6.

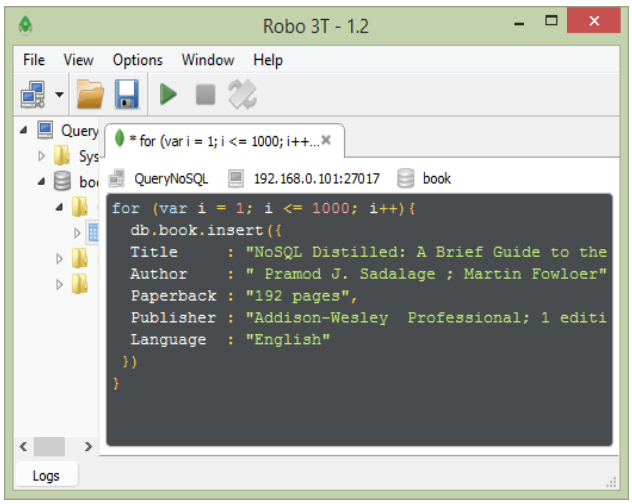

Figure 6. Insert query on Robo3T

Figure 6 displays the insert data query (collection) on the first experiment using Robo3T. The database with book names is used to hold all collections in this experiment. The next query creates data query is performed on ArangoDB. Query execution is performed on theArangoDB Web Interface, as shown in Figure 7.

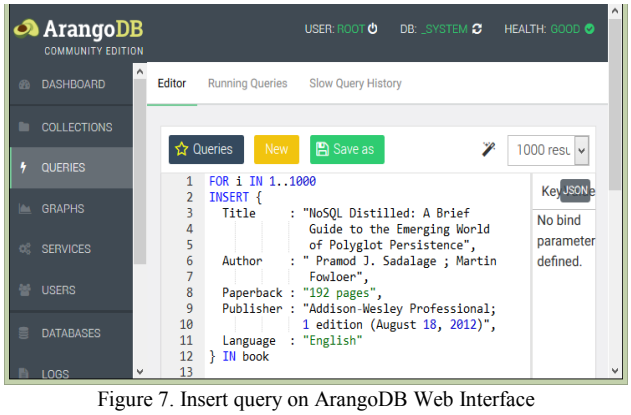

Figure 7. Insert query on ArangoDB Web Interface

The next query creates data is performed on CouchDB. Query execution on CouchDB is done via the command line. Curl is used in query execution stored in * .bat files, as shown in Figure 8.

```
::=====1 Create.bat=====
for /L %%i IN (1,1,1000) do
 (curl -X PUT "http://192.168.0.100:5984/book/%%i" -d
 "{""Title""     : ""NoSQL Distilled: A Brief Guide to
          the Emerging World of Polyglot Persistence"",
   ""Author""    : ""Pramod J. Sadalage ; Martin Fowloer"",
   ""Paperback"" : ""192 pages"",
   ""Publisher"" : ""Addison-Wesley Professional;
                  1 edition (August 18, 2012)"",
   ""Language""  : ""English""
  }"
  )
```

Figure 8. CouchDB insert query stored in a batch file

The Measure-Command command that is in Windows PowerShell is selected to be used to display the response time after the execution is done. The use of the Measure-Command command in this experiment is shown in figure 9.

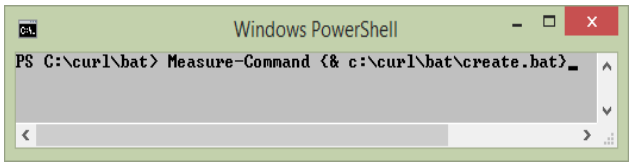

Figure 9. Measure-Command for executing a batch file

C. *Measurement Results Query Response Times*

After each query is executed, the response time generated is recorded in the table. The experimental results of each query create execution are shown in Table 3.

TABLE 3. MEASUREMENT RESULTS OF QUERY RESPONSE TIMES FOR CREATE COMMANDS

No	Qty	MongoDB (second)	Arango DB (second)	CouchDB (second)
1	1.000	24,100	15,571	107,428
2	2.000	47,200	15,998	216,994
3	3.000	70,000	27,980	340,162
4	4.000	106,000	39,979	444,580
5	5.000	137,000	42,935	593,960
Average		**76,860**	**28,493**	**340,625**

Table 3 displays the measurement results of the response time query for the create data (collection) command on the MongoDB, ArangoDB, CouchDB databases. After five attempts, the average ArangoDB response time is the smallest, which is 28,498 seconds compared to MongoDB and CouchDB.

TABLE 4. MEASUREMENT RESULTS OF QUERY RESPONSE TIMES FOR READ COMMANDS

No	Qty	MongoDB (second)	Arango DB (second)	CouchDB (second)
1	1.000	0,011	1,005	0,116
2	2.000	0,015	1,006	0,129
3	3.000	0,018	2,016	0,171
4	4.000	0,020	2,017	0,239
5	5.000	0,023	2,979	0,276
Average		**0,017**	**1,805**	**0,186**

Tabel 4 menampilkan data hasil pengukuran query response time untuk perintah read data (collection) pada database MongoDB, ArangoDB, CouchDB. Setelah dilakukan 5 kali percobaan rata-rata response time MongoDB paling kecil yaitu 0,017 second dibanding dengan ArangoDB dan CouchDB.

TABLE 5. HASIL PENGUKURAN QUERY RESPONSE TIMES UNTUK PERINTAH UPDATE

No	Qty	Mongo DB (second)	Arango DB (second)	CouchDB (second)
1	1.000	7,290	33,062	112,174
2	2.000	16,700	39,227	241,076
3	3.000	23,900	56,280	370,104
4	4.000	31,600	72,149	492,207
5	5.000	47,300	87,139	539,846
Average		**25,358**	**57,571**	**351,081**

Table 5 displays the data measured by the response time query for data update commands (collection) on the MongoDB, ArangoDB, CouchDB databases. After five attempts, the average MongoDB response time is the smallest, which is 25,358 seconds compared to ArangoDB and CouchDB.

TABLE 6. MEASUREMENT RESULTS OF QUERY RESPONSE TIMES FOR CREATE COMMANDS

No	Qty	Mongo DB (second)	Arango DB (second)	CouchDB (second)
1	1.000	0,030	8,939	106,757
2	2.000	0,037	16,997	214,273
3	3.000	0,049	24,984	340,379
4	4.000	0,071	33,003	462,097
5	5.000	0,088	42,989	617,816
Average		**0,055**	**25,382**	**348,264**

Table 6 displays the measurement results of query response time for data update commands (collection) on the MongoDB, ArangoDB, CouchDB databases. After five trials, the average MongoDB response time was the lowest, which was 0.055 second compared to ArangoDB and CouchDB.

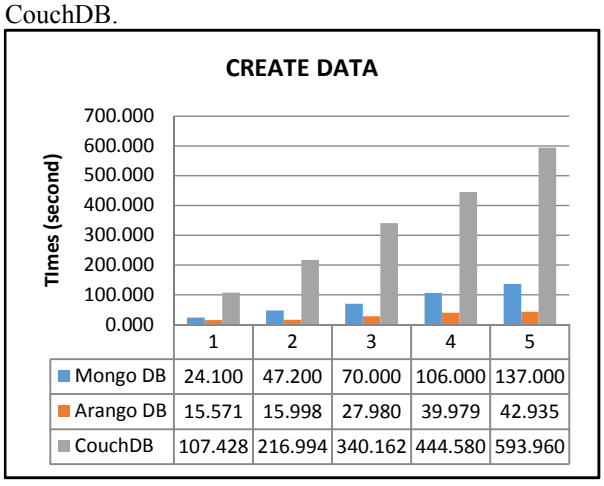

Figure 10. Query Response Time for Create Data

In Figure 10, the time increase of the query response times is displayed for the create data process that is carried out from 5 attempts. The response times for the create data process increase according to the amount of data entered. In the process of creating data, the Response Time query in

ArangoDB is the smallest compared to MongoDB and CouchDB.

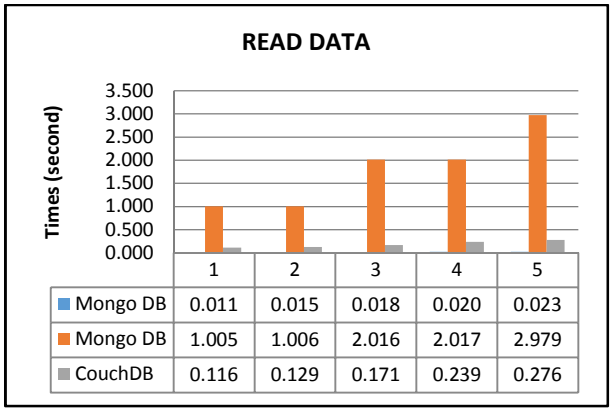

Figure 11. Query Response Time for reading Data

In Figure 11, the time increase of the query response times is displayed for the create data process that is carried out from 5 attempts. In the process of reading data, the Response Time query in MongoDB is the smallest compared to ArangoDB and CouchDB.

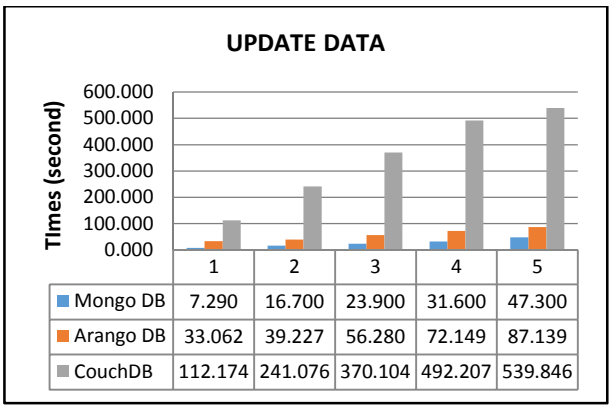

Figure 12. Query Response Time for Update Data

In Figure 12, the time increase of the query response times is displayed for the create data process that is carried out from 5 attempts. In the process of update data, the Response Time query in MongoDB is the smallest compared to ArangoDB and CouchDB.

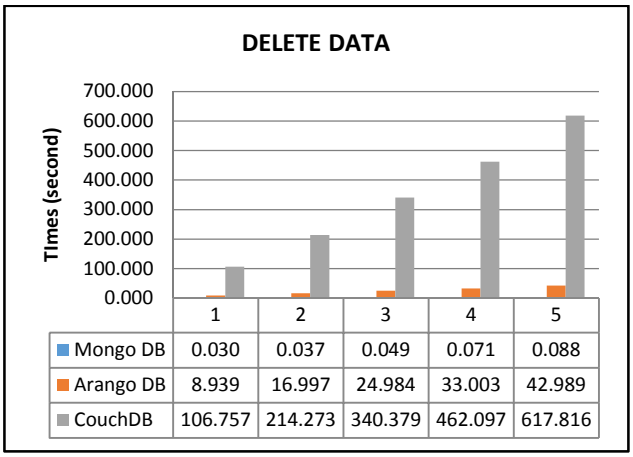

Figure 13. Query Response Time for Delete Data

In Figure 12, the time increase of the query response times is displayed for the create data process that is carried out from 5 attempts. In the process of update data, the Response Time query in MongoDB is the smallest compared to ArangoDB and CouchDB.

D. Comparison of Query Response Times

After all, experiments have been carried out, in this section compared to the average demand, the response of each request creates, reads, deletes, and updates each NoSQL database. The values obtained after calculating the average time are calculated in Table 7.

TABLE 7. COMPARISON OF QUERY RESPONSE TIMES

Query	MongoDB (second)	ArangoDB (second)	CouchDB (second)
Create (Avg)	76,860	**28,493**	340,625
Read (Avg)	**0,017**	1,805	0,186
Update (Avg)	**25,358**	57,571	351,081
Delete (Avg)	**0,055**	25,382	348,264

Table 7 shows the average query response times resulting from the entire experiment. MongoDB has the lowest average query response time for the read process (0.017 seconds), update (25.358 seconds), and delete (0.055 seconds) compared to ArangoDB and CouchDB. However, for the creation process, the ArangoDB response time is the smallest, which is 28,493 seconds compared to MongoDB and CouchDB.

V. CONCLUSION

Based on the results of experiments on the study, it is known that: MongoDB has the lowest average query response time for read, update, and delete processes compared to ArangoDB and CouchDB. The response time of ArangoDB for the creating process is the smallest compared to MongoDB and CouchDB. Apart from that, MongoDB and ArangoDB also provide a Client GUI that provides query syntax support that is very helpful to developers. Whereas query syntax exploration on CouchDB can be done with the help of curl, which can be accessed via the command line.

Comparison of the response time query performance on other NoSQL databases such as key-value, graph, a full column, and exploration queries other than the CRUD command is the arrival that can be done in the next study.

REFERENCES

[1] V. Abramova, "NOSQL databases," *Adv. Inf. Knowl. Process.*, pp. 77–84, 2019.

[2] M. A B M, "NoSQL Database: New Era of Databases for Big data Analytics," *New Sci.*, vol. 6, no. 4, pp. 1–14, 2012.

[3] T. Li, Y. Liu, Y. Tian, S. Shen, and W. Mao, "A storage solution for massive IoT data based on NoSQL," *Proc. - 2012 IEEE Int. Conf. Green Comput. Commun. GreenCom 2012, Conf. Internet Things, iThings 2012 Conf. Cyber, Phys. Soc. Comput. CPSCom 2012*, pp. 50–57, 2012.

[4] E. Mitreva and K. Kaloyanova, "NoSQL Solutions to Handle Big Data," *Proc. Dr. Conf. MIE*, no. September 2013, pp. 77–86, 2013.

[5] Y. Li and S. Manoharan, "A performance comparison of SQL and

NoSQL databases," *IEEE Pacific RIM Conf. Commun. Comput. Signal Process. - Proc.*, no. November, pp. 15–19, 2013.

[6] S. Agarwal, A. Panda, B. Mozafari, S. Madden, and I. Stoica, "BlinkDB: Queries with Bounded Errors and Bounded Response Times on Very Large Data," in *International Conference 10th Edition: Networking in Education and Research*, 2012.

[7] C. Truica, "Performance evaluation for CRUD operations in asynchronously replicated document oriented database," in *2015 20th International Conference on Control Systems and Computer Science*, 2015.

[8] C. J. M. Tauro, Aravindh S., and A. B. Shreeharsha, "Comparative Study of the New Generation, Agile, Scalable, High Performance NOSQL Databases," *Int. J. Comput. Appl.*, vol. 48, no. 20, pp. 1–4, 2012.

[9] M. Chevalier, "How Can We Implement a Multidimensional Data Warehouse Using NoSQL," *ICEIS*, vol. 241, pp. 108–130, 2015.

[10] V. N. Gudivada, D. Rao, and V. V. Raghavan, "NoSQL Systems for Big Data Management," pp. 190–197, 2014.

[11] M. Wadhwa and A. Kaur, "PERFORMANCE EVALUATION ON CRUD OPERATIONS IN BIG DATABASES" no. 10, pp. 63–68, 2017.

[12] R. Zafar, E. Yafi, M. F. Zuhairi, and H. Dao, "Big Data: The NoSQL and RDBMS review," *ICICTM 2016 - Proc. 1st Int. Conf. Inf. Commun. Technol.*, no. May, pp. 120–126, 2017.

[13] D. A. Pereira, W. O. De Morais, and E. P. De, "NoSQL real-time database performance comparison," *Int. J. Parallel, Emergent Distrib. Syst.*, vol. 5760, no. April, pp. 1–13, 2017.

[14] M. V, "C Omparative S Tudy of V Arious Voip," *Int. J. Database Manag. Syst. (IJDMS)*, vol. 3, no. 5, pp. 29–39, 2013.

[15] R. Gunawan, "PENGUKURAN QUERY RESPON TIME PADA NOSQL DATABASE BERBASIS DOCUMENT STORED," vol. 4, no. 2, pp. 100–103, 2018.