J. Int. Intl. Sys. Soft, Vol. 1, No. 2, 2022
ejournal.utdi.ac.id/index.php/JISS/index – eISSN : 2962-7702

JISS Journal of Intelligent Software Systems

Journal of Intelligent Software System – JISS

Query Execution Performance Analysis of Column-Oriented Database in Dashboard

Original Article doi: 10.26798/jiss.v1i2.768

sumbit: *2022-12-13*, **accepted:** *2022-12-22*, **Publish:** *2022-12-29*

Bagas Triaji[1]*, Widyastuti Andriyani[2]†, Totok Suprawoto[3]‡, M. Agung Nugroho[4]§, Rikie Kartadie[5]¶

1 Software Engineering, Universitas Teknologi Digital Indonesia
2 Master in Information Technology Universitas Teknologi Digital Indonesia
3 Information Systems, Universitas Teknologi Digital Indonesia
4 Informatics, Universitas Teknologi Digital Indonesia
5 Computer Engineering, Technology Universitas Teknologi Digital Indonesia

Abstract: In making reports or dashboards from operational data, problems often occur in the query process with low speed in responding to an output, causing the server to experience overload. This condition often occurs in companies or higher education organizations in managing academic data. This condition can be improved by optimizing the database server by integrating relational databases with column-oriented databases to speed up query responses and save development costs. Based on the experiments that had been carried out, column-oriented has succeeded in optimizing with a significant difference in query execution time and the server does not crash.

Keywords: column-oriented database • relational database • column store • MySQL • clickhouse • bigdata

* *E-mail: bagastriaji@utdi.ac.id*
† *E-mail: widya@utdi.ac.id*
‡ *E-mail: totok@utdi.ac.id*
§ *E-mail: m.agung.n@utdi.ac.id*
¶ *E-mail: rikie@utdi.ac.id*

1. Introduction

In a modern organization, data are a very important asset. Data are used by top management to support logical decision-making, which decisions can be accepted by many parties by prioritizing considerations based on data and conditions in the field. Every organization has an information system that produces a structured relational database management system (RDBMS) such as MySQL [1] or MariaDB [2].

Unfortunately, as the number of data increases, when creating analytics or dashboards, the queries used to retrieve data from the database, often run very slowly. The process of data aggregation on the amount of data that can reach millions causes RDBMS such as MySQL to be less optimal [3].

In big data, data volume is a challenge for small-medium enterprise (SME) companies because they do not have the funds to upgrade the latest version of the server to improve database performance on their commodity servers [4].

Furthermore, changing the architecture as a whole and changing the source code side of the application might also take a lot of time and money. Thus, it is necessary to optimize the existing resources, in this case on the database server-side.

This study is a case study on a consulting firm and university software service provider who is currently conducting a study on the optimization of report generation. The researchers proposed a solution to the optimization problem by experimenting with the use of column-oriented databases by choosing one of the Column-oriented database brands, namely ClickHouse, which results are expected to be a solution for companies or industries.

The contributions made by this study include as follows; concurrent use of column-oriented relational databases, the use of commodity servers with the aim that the results of the study can also be utilized by SMEs, the use of Speedup metrics and independent t-test statistical tests to determine the significance of resource usage.

Section 2 discusses the background of column-oriented capabilities and discusses several previous related studies. Section 3 presents the method and experimental preparations such as hardware, software, and queries used. Section 4 presents the data in the form of graphs and tables of the results of the experimental analysis, including things that the researcher discusses based on the results of the analysis. Section 5 draws conclusions.

2. Literature Review

2.1. Column-oriented Database

Column-oriented or commonly called Columnar database is a form of sorting data in a relational database which is generally in the form of rows but the form of columns because columns are the smallest and lowest instance of data [5]. Column-oriented objectives are to speed up reading and sorting on storage because it minimizes random access, reduces disk Input-Output (I/O), and processes queries using vectorized execution [6, 7]. Vectorized execution query is an execution model that can be represented as small single-dimensional arrays (vectors), easily accessible for CPUs, and can also speed up heavy query processing in big data frameworks such as Online Analytical Processing (OLAP) which has a lot of aggregate commands on big data [8–10] The role of parallel processing in OLAP is a good solution to improve performance [11]. These conditions suggest that the column-oriented databases currently also play a role in the development of Big Data [12].

2.2. ClickHouse

Currently, there are many column-oriented DBMS products, one of which is currently popular is ClickHouse. This open-source product is made by Yandex Russia. ClickHouse has a familiar SQL dialect feature and can integrate with MySQL, making it easier for developers to structure queries. In terms of performance, several studies have stated that ClickHouse is able to process quickly on ATLAS [13] and modern analytical databases [14].

A previous study conducted by Felipe de Moura et al. [15] tried to shorten query execution time on financial report generation using the Hadoop ecosystem with Hbase as column-oriented DBMS. The data in the relational DBMS were transferred to Hbase and then the query execution was carried out. The result was that the query execution time for making reports is shorter. Other studies on column-oriented databases were also conducted by Alex Skidanov et al., Matei-Eugen Vasile et al., Tianqi Zheng et al. [7, 13, 14] using memSQL and ClickHouse for real-time data analytics and concluded that column-oriented databases are reliable because real-time analytics requires query execution speed. Meanwhile, another study conducted by Czerepicki [16] to shorten the time of data analysis of electric vehicles in calculating power consumption and the chosen route, which will affect energy savings in the electric vehicle.

3. Methods

Speeding up the response time of the relational database by combining other database models was conducted by Vyawahare et al. [17], which is to do a hybrid database approach

between relational databases integrated with graph databases to improve the performance of analytical queries. In this experiment, the researchers did not change the database that had been used and did not add hardware but ran a relational DBMS (MySQL) and column-oriented database (ClickHouse) on the same server. MySQL is a data source because it stores operational data and is integrated with ClickHouse, when it will process report generation, the query will be run on ClickHouse. Further detail on the experiment is visualized in Figure 1.

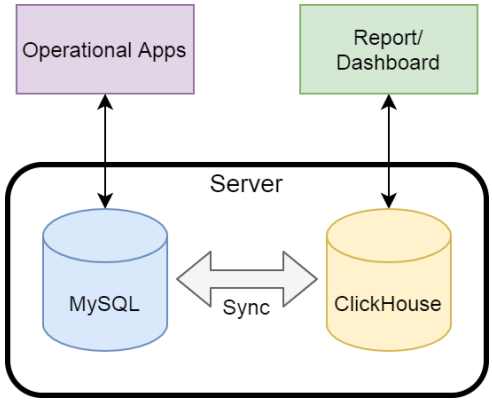


Figure 1: MySQL and ClickHouse in one server

Testing using hardware specifications was not quite high, since it adjusted the condition of the server owned by the client or customer. In this study, the researchers used 4 GB RAM, 50 GB storage disk, and Ubuntu OS 20.04.

3.1. MySQL and ClickHouse

MySQL versions used version 8.0 with the following tunning configuration:

```
sort_buffer_size=1M,read_rnd_buffer_size=1M,
innodb_buffer_pool_size=2G, innodb_log_file_size=128M,
innodb_buffer_pool_instances=3, join_buffer_size=512K
```

ClickHouse version used 20.9 stable. Without tuning and the type of database engine used was mysql. This type of database engine can remote database from Clickhouse to MySQL in real-time to synchronize the data.

3.2. Data and Query

The data used were the academic information system data of one of the customer's colleges. However, in this experiment, only 4 tables were used and the maximum number of data was

7 million records. This table was the core table used to create reports. Meanwhile, the query consisted of 4 types the query in Figure 2, is the highest query and takes a long time to execute.

Query 1:

```
SELECT DISTINCT
    kp1.field1,
    ...
    kp1.fieldn
FROM
  table_a AS kp1
INNER JOIN
    (SELECT field1, MAX(field2)
FROM table_a
WHERE condition1, condition2)
    GROUP BY field3) AS kp2 ON kp2.field1= kp1.field1 AND kp1. field2 = kp2.
field2
ORDER BY kp1.field1 DESC ;
```

Query 2:

```
SELECT
  field1,
  COUNT(field2) / 2,
  Field3
FROM
  table_a
GROUP BY field1, field3
ORDER BY field1 ;
```

Query 3:

```
SELECT
  table_b.field1,
  COUNT(table_a.field3) / 2,
    ...
  table_a.fieldn
FROM
  table_a
LEFT JOIN table_b ON table_b.matching_field = table_a. matching_field
LEFT JOIN table_c ON table_c.matching_field = table_b. matching_field
WHERE condition1
GROUP BY km.field3, mpt.field1, mhs.field2,  km.field4
ORDER BY km.field3 ASC;
```

Query 4:

```
SELECT
  table_d.field1,
  AVG(table_e.field2),
  ...
  fieldn
FROM
  table_a
LEFT JOIN table_b
ON table_b.matching_field = table_a.matching_field
LEFT JOIN table_c
ON table_c.matching_field = table_a.matching_field
LEFT JOIN table_d
ON table_d.matching_field = table_a.matching_field
LEFT JOIN table_e
ON table_e.matching_field = table_a.matching_field
GROUP BY field1,field2
ORDER BY field1 ;
```

Figure 2: Four types of queries used for testing

3.3. Query execution and logging

Query execution was divided into 2 types, the first was the first run where a query was executed the first time or had never been executed before, to avoid the use of cache. The second was the next run, a query executed after the first run had been executed or a query that had been executed before. During the execution, data such as execution time, CPU usage, and RAM usage were recorded using system activity report (sar) and pidstat tools. Each query was executed five times and then the average was taken. The average calculation used the arithmetic mean with the following notation:

$$A = \frac{1}{n}\sum_{i=1}^{n} a_1 \tag{1}$$

Where A is the arithmetic mean, the a_1 is the i-th data value and n is the number of data.

4. Result and Analysis

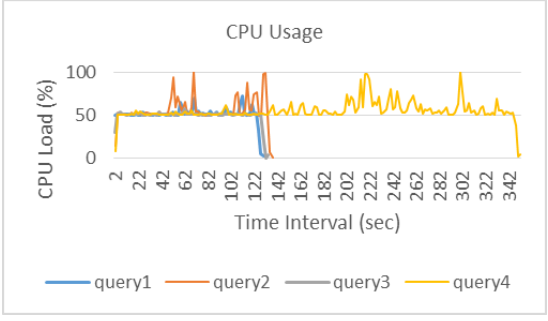
MySQL Query Execution

In order to measure the optimization level, it is necessary to execute the query using MySQL for comparison before and after optimization. The results of the query execution are presented in Table 1.

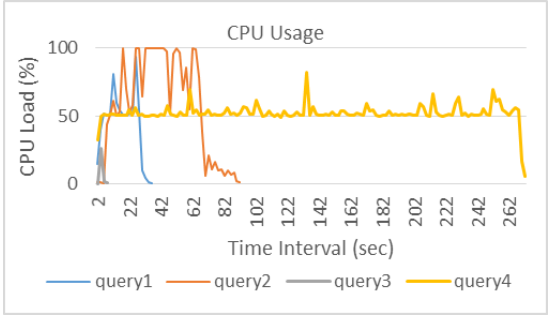
Table 1: MySQL query execution time

Query	First run (sec)	Next run (sec)
query 1	125.08	25.64
query 2	131.86	60.57
query 3	126.91	0.84
query 4	351.06	267.72

During the execution of query 3, there was a failure in the execution process. MySQL failed to complete the query and displayed an error message.



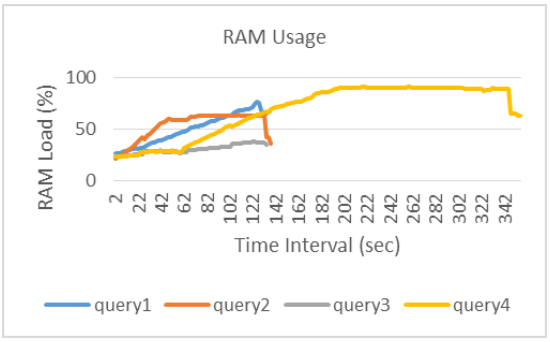
(a) First run query execution CPU Usage



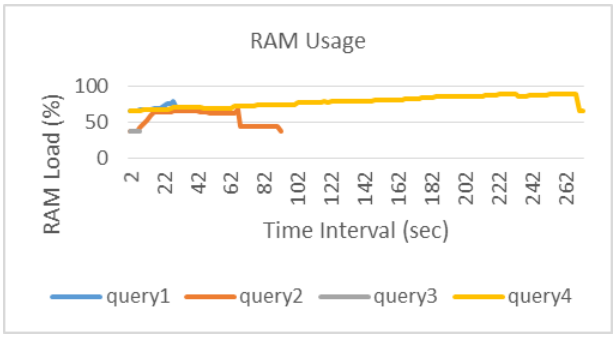
(b) Next run query execution CPU Usage

Figure 3: query execution CPU Usage

Execution of Query First Run in MySQL did not take too long to reach peak CPU usage as visualized in Figure 3a. In the Next Run execution, peak CPU usage was more frequent in query 2, other queries averaged 50% as in Figure 3b.



(a) First run query execution RAM Usage



(b) Next run query execution RAM Usage

Figure 4: Query execution RAM Usage

RAM usage in First Run execution tends to be higher than the next run execution, while query4 has similarities in RAM usage and query execution time as in Figure 4a and Figure 4b.

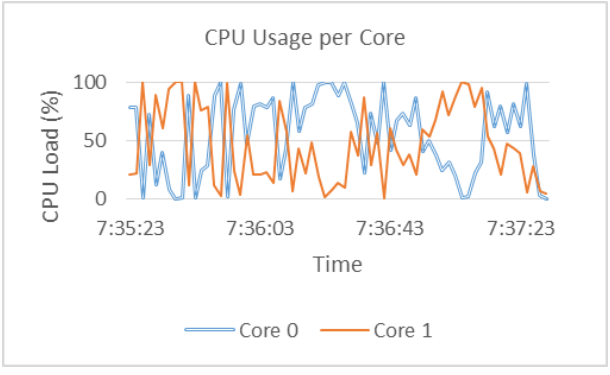


Figure 5: CPU usage per core

The CPU Core as shown in Figure 5, visualizes the alternate between core1 and core2 when executing the query.

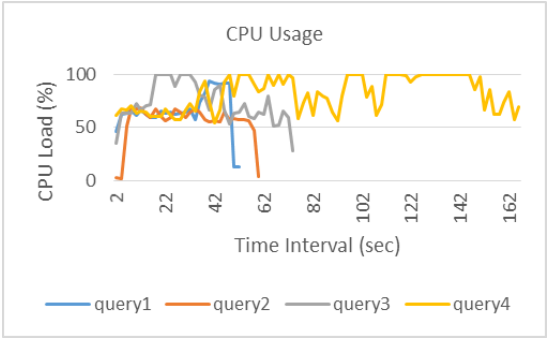
ClickHouse Query Execution

When executing the query used to generate reports on ClickHouse, it must first be ensured that ClickHouse is connected to MySQL. The results of the execution are presented in Table 2.

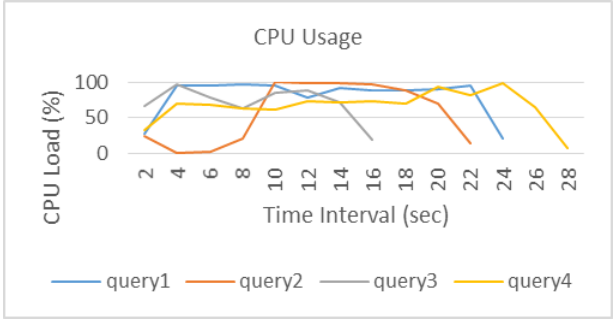
Table 2: ClickHouse query execution time

Query	First run (sec)	Next run (sec)
query 1	50.150	22.74
query 2	54.346	14.138
query 3	87.274	15.835
query 4	115.561	13.773

The first query shows the First run at a speed of 50.150 with the Next run at a speed of 22.74. Meanwhile, the second, third, and fourth queries experienced an increase in the speed of the First run (Query 2, 3, and 4) and a decrease in the Next run (Query 3 and 4)



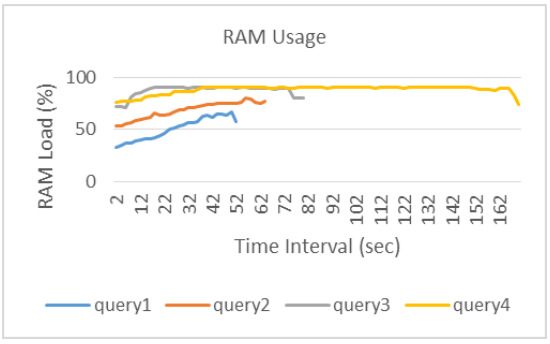
(a) First run query execution CPU Usage with ClickHouse



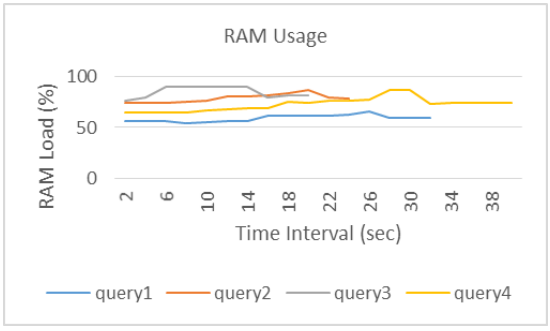
(b) Next run query execution CPU Usage with ClickHouse

Figure 6: Query execution CPU Usage with ClickHouse

CPU usage on First Run ClickHouse queries tends to be high, especially on heavy queries and requires a long execution time as shown in Figure 6a. In the execution of the Next Run Query, almost all queries use maximum CPU resources as shown in Figure 6b.



(a) First run query execution RAM Usage with ClickHouse



(b) Next run query execution RAM Usage with ClickHouse

Figure 7: Query execution RAM Usage with ClickHouse

RAM usage in the First Run execution seems to increase while the Next Run execution tends to ramp up and does not reach the peak resource usage as shown in Figure 7a and Figure 7b.

ClickHouse execution always consumes all the cores on the CPU at the same time, as in Figure 8.

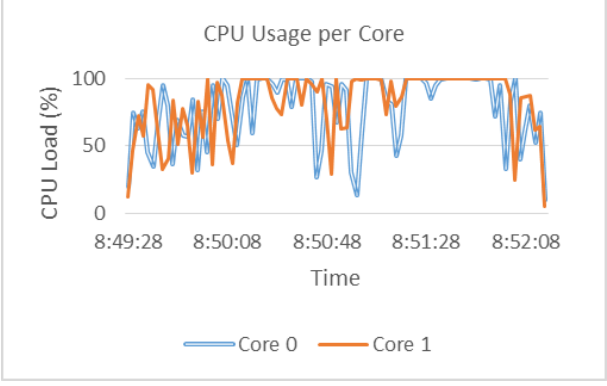
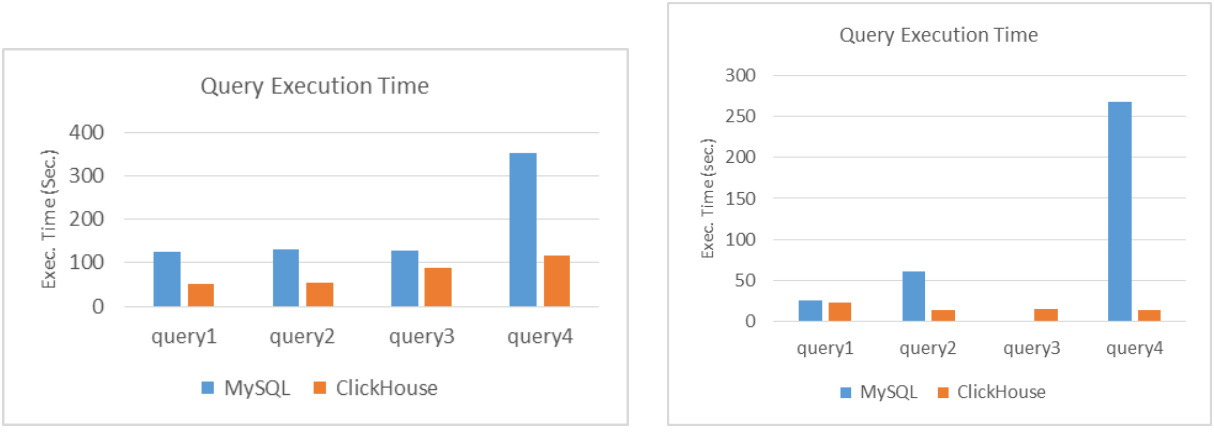


Figure 8: CPU usage per core with ClickHouse

The difference in query execution time between MySQL and ClickHouse on the first execution appears to have a difference as shown in Figure 9a. MySQL takes longer to execute the query. Meanwhile, in the execution of the next run, the difference is quite striking as in Figure 9b.



(a) First run query execution time

(b) Next run query execution time

Figure 9: Query execution time

ClickHouse is much faster than MySQL except for query3, ClickHouse is slightly slower than MySQL. Tests were carried out to find out the significant average difference in resource usage between when only using MySQL compared when it was optimized with ClickHouse. Both query execution first run and next run used an independent t-test with mathematical notation as follows:

$$t = \frac{X1 - X2}{\sqrt[n]{\frac{(n1-1)s_{1^2}}{n1+n2-2}\left(\frac{1}{n1} + \frac{1}{n2}\right)}} \tag{2}$$

Where Xi= average score/value of group i, = is the number of group i, S_{1^2}= variance score of group i. The significance level (α) = 0.05 was used. The results are shown in Table 3, indicating that the value of sig (2-tailed) $< \alpha$ means that the use of RAM, CPU and RAM+CPU between MySQL alone compared to after using ClickHouse has a significant difference. As seen in Table 3, almost every query execution experienced a significant reduction in time, even up to 94.86%, but did not experience a reduction in query 3 because, without ClickHouse, MySQL was faster. Furthermore, the calculation of the increase in speed (speedup) in parallel) computing compared to serial computing because using ClickHouse, queries are executed in parallel.

Table 3: Execution time comparisons

Query Exec.	Query	MySQL (sec)	ClickHouse (sec)	Time reduction (%)	Speedup
First run	query 1	125.08	50.15	59.91%	2.49
	query 2	131.86	54.346	58.79%	2.43
	query 3	126.91	87.274	31.23%	1.45
	query 4	351.06	115.561	67.08%	3.04
Next run	query 1	25.64	22.74	11.31%	1.13
	query 2	60.57	14.138	76.66%	4.28
	query 3	0.84	15.835	-	0.05
	query 4	267.72	13.773	94.86%	19.44

Speedup calculation is formulated as follows:

$$S_p = \frac{T_1}{T_p} \tag{3}$$

Where S_p= Speedup, T_1 = serial computation execution time, T_p = parallel computation execution time using as many P (processor). Therefore, the Speedup column can be seen how many times faster parallel execution $T_p$is compared to serial T_1.

From the experimental results, it is known that the use of MySQL in report generation takes a long time as described in Table 1. Only query 3 can be executed quickly. Compared to ClickHouse, it can complete in a faster time, with the average time as in Table 2. Meanwhile, the comparison can be seen in Figure 9a and Figure 9b. The results are striking in query 4. The time reduction is more than 253 seconds or 9 times faster as described in Table 3. These results are almost the same as other studies which reach 10 times faster in processing data, to produce efficient outputs such as what has been done by Djamaluddin et al [18].

Furthermore, in terms of server resource usage, CPU, and RAM, when using only MySQL, CPU usage is not different between the first run and next run as shown in Figure 3a and Figure 3b. In the execution of the first run, the CPU is not used optimally. Meanwhile, with ClickHouse, the CPU can reach 100% usage for a while and is accompanied by a shorter execution time to make the way it works more optimal. Besides, ClickHouse can execute queries in parallel on each CPU core, as shown in Fig. 12. The graph between Core 0 and Core 1 almost always coincides when it goes up and down. The execution time is up to 19 times faster when processed in parallel. This condition is in contrast to MySQL which tends to execute queries serially as shown in Figure 5. The graphics between Core 0 and Core 1 are opposite each other, indicating that execution will utilize one core first, while the other cores are idle. Parallel processing on multi-core CPUs becomes a necessity because the current era CPU architecture allows it and can minimize execution time [19]. The way it works is not optimal if the data being processed takes a long time because the query runs serially on the CPU.

The results of the t-test indicate that there was a significant difference between before and after using ClickHouse. The system succeeded in maximizing existing resources to speed up execution time. Thus, the query execution process became more optimal. RAM usage of both MySQL and ClickHouse is not much different. ClickHouse has a tendency to use higher RAM as shown in Figure 4a, Figure 4b, Figure 7a and Figure 7b. It is suspected that minor page faults (data reading from memory) [20] are more dominant than major page faults, indicating that it minimizes data reading from disk, but from memory pages.

During the query execution process, ClickHouse did not fail to complete the process. However, seen from the graph, the CPU and RAM usage which is always high might delay other processes because all resources are being used by ClickHouse. SQL dialect which is not much different between MySQL and ClickHouse can be taken into consideration, indicating that it does not take much time to learn a new query language. Since not all column-oriented databases use SQL, thus, SQL Layers such as Apache Phoenix [21] must be added to profit for the company.

5. Conclusions

A column-oriented database such as ClickHouse can be used to optimize server capabilities in generating reports or dashboards. Column-oriented databases do not replace but complement existing relational databases. However, this is still an early stage of the experiment and still needs to be developed further. In the next experiment, it is possible to use more complex queries, larger data, by tuning the configuration, using another type of ClickHouse table engine that has better performance, and

References

[1] W. Khan, W. Ahmad, B. Luo, and E. Ahmed, "SQL Database with physical database tuning technique and NoSQL graph database comparisons," in 2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC), 2019, no. Itnec, pp. 110–116, doi: 10.1109/ITNEC.2019.8729264.

[2] Hendra and W. Andriyani, "Studi Komparasi Menyimpan Dan Menampilkan Data Histori Antara Database Terstruktur Mariadb Dan Database Tidak Terstruktur Influxdb," J. Teknol. Technoscientia, vol. 12, no. 2, pp. 168–174, 2020.

[3] W. Puangsaijai and S. Puntheeranurak, "A comparative study of relational database and key-value database for big data applications," in 2017 International Electrical Engineering Congress (iEECON), 2017, no. March, pp. 1–4, doi: 10.1109/IEECON.2017.8075813

[4] A. Alharthi, V. Krotov, and M. Bowman, "Addressing barriers to big data," Bus. Horiz., vol. 60, no. 3, pp. 285–292, May 2017, doi: 10.1016/j.bushor.2017.01.002.

[5] N. Saeed, "Big Data with Column Oriented NOSQL Database to Overcome the Drawbacks of Relational Databases," no. February, 2020.

[6] Steve Ataky Tsham Mpinda, Patrick Andjasubu Bungama, and Luis Gustavo Maschietto, "From Relational Database to Column-Oriented NoSQL Database: Migration Process," Int. J. Eng. Res., vol. V4, no. 05, May 2015, doi: 10.17577/IJERTV4IS050021.

[7] A. Skidanov, A. J. Papito, and A. Prout, "A column store engine for real-time streaming analytics," in 2016 IEEE 32nd International Conference on Data Engineering (ICDE), 2016, pp. 1287–1297, doi: 10.1109/ICDE.2016.7498332.

[8] J. Sompolski, M. Zukowski, and P. Boncz, "Vectorization vs. compilation in query execution," in Proceedings of the Seventh International Workshop on Data Management on New Hardware - DaMoN '11, 2011, no. DaMoN, pp. 33–40, doi: 10.1145/1995441.1995446.

[9] S. Meraji, J. Keenleyside, S. Kamath, and B. Blainey, "Towards a Combined Grouping and Aggregation Algorithm for Fast Query Processing in Columnar Databases with GPUs," in 2015 IEEE International Parallel and Distributed Processing Symposium Workshop, 2015, pp. 594–603, doi: 10.1109/IPDPSW.2015.21.

[10] H. Lang, T. Mühlbauer, F. Funke, P. A. Boncz, T. Neumann, and A. Kemper, "Data Blocks," in Proceedings of the 2016 International Conference on Management of Data - SIGMOD '16, 2016, vol. 26-June-20, no. June, pp. 311–326, doi: 10.1145/2882903.2882925.

[11] R. S. Kalan and M. O. Unalir, "Leveraging big data technology for small and medium-sized enterprises (SMEs)," in 2016 6th International Conference on Computer and Knowledge Engineering (ICCKE), 2016, no. Iccke, pp. 1–6, doi: 10.1109/ICCKE.2016.7802106.

[12] S. Kalid, A. Syed, A. Mohammad, and M. N. Halgamuge, "Big-data NoSQL databases: A comparison and analysis of 'Big-Table', 'DynamoDB', and 'Cassandra,'" in 2017 IEEE 2nd International Conference on Big Data Analysis (ICBDA)(, 2017, pp. 89–93, doi: 10.1109/ICBDA.2017.8078782.

[13] M.-E. Vasile, G. Avolio, and I. Soloviev, "Evaluating InfluxDB and ClickHouse database technologies for improvements of the ATLAS operational monitoring data archiving," J. Phys. Conf. Ser., vol. 1525, no. 1, p. 012027, Apr. 2020, doi: 10.1088/1742-6596/1525/1/012027.

[14] T. Zheng, Z. Zhang, and X. Cheng, "SAHA: A String Adaptive Hash Table for Analytical Databases," Appl. Sci., vol. 10, no. 6, p. 1915, Mar. 2020, doi: 10.3390/app10061915.

[15] F. de Moura Rezende dos and M. Holanda, "Performance Analysis of Financial Institution Operations in a NoSQL Columnar Database," in 2020 15th Iberian Conference on Information Systems and Technologies (CISTI), 2020, vol. 2020-June, no. June, pp. 1–6, doi: 10.23919/CISTI49556.2020.9140981.

[16] A. Czerepicki, "Study on effectiveness of using column-oriented databases in the processing of measurement characteristics of an electric vehicle," Arch. Transp., vol. 51, no. 3, pp. 77–84, Sep. 2019, doi: 10.5604/01.3001.0013.6164.

[17] H. R. Vyawahare, P. P. Karde, and V. M. Thakare, "Hybrid Database Model For Efficient Performance," Procedia Comput. Sci., vol. 152, no. September, pp. 172–178, 2019, doi: 10.1016/j.procs.2019.05.040.

[18] B. Djamaluddin, P. Prabhakar, B. James, A. Muzakir, and H. AlMayad, "Real-Time Drilling Operation Activity Analysis Data Modelling with Multidimensional Approach and Column-Oriented Storage," in Day 3 Wed, March 20, 2019, 2019, vol. 2019-March, no. 1, pp. 1–13, doi: 10.2118/194701-MS.

[19] S. Deepak, S. U. Kumar, M. Durgesh, and P. B. K., "Query Processing and Optimization of Parallel Database System in Multi Processor Environments," in 2012 Sixth Asia Modelling Symposium, 2012, pp. 191–194, doi: 10.1109/AMS.2012.49.

[20] R. Titos-Gil, R. Fernández-Pascual, A. Ros, and M. E. Acacio, "PfTouch: Concurrent page-fault handling for Intel restricted transactional memory," J. Parallel Distrib. Comput., vol. 145, pp. 111–123, Nov. 2020, doi: 10.1016/j.jpdc.2020.06.009.

[21] H.-J. Kim, E.-J. Ko, Y.-H. Jeon, and K.-H. Lee, "Migration from RDBMS to Column-Oriented NoSQL: Lessons Learned and Open Problems," in Lecture Notes in Electrical Engineering, vol. 461, 2018, pp. 25–33.