Study of Search Algorithm Optimization from Multi-Version Data Warehouse using NoSQL Non-relational Database

L R Maghfiroh[1], **R A Yusuf**[1]

[1]Department of Statistical Computing, Politeknik Statistika STIS

*Corresponding author's e-mail: lutfirm@stis.ac.id

Abstract. Statistics Indonesia, which produces large-scale data, requires effective and optimal storage. Research related to Multi-Version Data Warehouse (MVDW), which utilizes document-based NoSQL itself, has attempted to be developed for the sake of BPS data storage and proposed an algorithm to store and search data. This paper is made to examine algorithm optimization methods to reduce the time used in the process of storing and searching data when needed. The algorithm proposed in this paper focuses on the data storage process by suggesting a storage model that generalizes the coding of variables in the data warehouse used so that later data searches can be carried out more easily and optimally. Other optimization methods are also carried out by applying query optimization methods to support and improve the optimization of the proposed algorithm. The results of the two optimization methods carried out can be said to be successful because the time used in the data search process by utilizing the algorithm after the application of the optimization method has been reduced when compared to the data search process using algorithms that have been developed by previous research.

1. Introduction

Statistics Indonesia (BPS) conducts surveys and censuses regularly intending to produce data and information that will later be published, both in the form of raw data and printed books. Before the data and information are processed, the data results and metadata from each survey and census from each period are stored in a data warehouse (DW). DW is a subject-oriented data repository that is integrated, non-volatile, and can support the collection of time-variant data that is used to make data storage more structured [1].

Eric Brewer [2] developed a theory which states that indicators of distributed data storage so that it can be said to be structured are by paying attention to consistency, data availability (availability), and tolerance for network partitions (partition tolerance). This is then better known as the CAP theorem, which describes the priority that can be prioritized from a data storage system. In the next few years, Brewer conducted a further study of this theorem, which later became a reference for the creation of a NoSQL (Not Only SQL) non-relational database system for data storage [3].

NoSQL itself has been used in various data storage implementations in recent years. This is because, in its use of large-scale data, the implementation of the NoSQL nonrelational database system is considered to have more advantages such as flexibility and ability to store and retrieve large volumes of data compared to relational databases [4].

According to an interview with one of the developers of Portal Indonesia Data Hub (INDAH) [19], the use of non-relational databases at BPS is the use of Hadoop Distributed File Service (HDFS) [5] as

a data lake, while the use of DW in BPS data storage still uses Microsoft SQL Server [6] and IBM DB2 with BLU Acceleration [7] which is a relational database. Over time, the use of relational databases in activities being carried out by BPS to build the One Data Indonesia Portal will face challenges related to handling the ever-increasing volume of data. This is in line with the nature of relational databases which have better performance when handling small and limited data [8].

As an effort to mitigate the challenges above, the use of a NoSQL database as an alternative to BPS data storage has begun to be developed. Apart from its flexibility and data management capabilities, this development is also carried out because data storage using a NoSQL database itself is not based on the form of the storage scheme, but is categorized based on the data structure [9]. Therefore, the NoSQL database that will be highlighted in this paper is a non-relational database with a document-based data structure, considering that the data source that is more often generated from BPS surveys and censuses is in the form of JSON or XML [10].

With the background of this research, Maghfiroh and Baskara [11] have studied the use of Multi-Version Data Warehouse (MVDW) uses document-based NoSQL as an alternative in storing data from BPS survey results. The MVDW developed in this study uses the MongoDB application, which is considered to have advantages in reading, write, and delete operations when compared to other document-based non-relational databases. [12]

This study was then continued with the development of the MVDW model and a more sophisticated algorithm by Maghfiroh and Santoso [13] to be more dynamic, to provide convenience in overcoming challenges in the data storage and retrieval process when needed in the analysis process. However, considering that the time required to run the query is still slightly higher, researchers in [13] themselves still suggest designing a simpler algorithm so that it can run queries faster.

Previous research on optimization of non-relational NoSQL databases has been minimal compared to other forms of databases, and has mostly focused on workload sharing or focused on database performance such as the implementation of server replication by Tinetti et al. [14] and Gu et al. [15]. On the other hand, there are still few who target database efficiency from the system aspect, one of which is the implementation of query optimization by Mahajan et al. [16].

Reference [16] itself mentions several methods that can be used to get a more optimal query, such as indexation, sorting, covering, aggregation, and sharding. The query optimization methods carried out in [16] also get quite large results, even hundreds of times more effective and faster than before the optimization.

Based on the brief description of the previous research above, this research will try to examine and apply optimization methods to the search process by dividing the research focus into two, namely in terms of the data collection algorithm and in terms of optimization of the query that will be used in the data collection process. With these proposed methods, we hope this research will get a faster data collection (store and search) process.

2. Methods

The initial stage of this research is to review the previous research that has been done to identify and review the problems that have not been resolved. The next stage is a literature study to find the previous reference and analyze the model that has been applied before, as well as defining the purpose of the research. After the literature study is carried out, the core of the research is carried out by conducting an experimental design, which consists of several stages, starting from empirical observation, theory building, empirical experimentation, and testing of the theory that has been built, which will be carried out circularly and repeatedly if needed [17]. The results obtained from this experimental design are then used as a reference in proposing a solution, which in this case is in the form of proposing a more optimal algorithm accompanied by other supporting optimization methods to achieve the goal.

Based on the initial study that the authors have done, supported by research results [11], [13] storing data from the BPS survey results has its challenges considering that the structure of the data obtained from each survey will change according to the needs of the survey objectives. This causes the

data storage in question will have different storage requirements, causing unavoidable structural differences in each survey.

One part of the data structure of the survey results that will experience differences from time to time is the coding of variables from each of these data. Changes in the coding of the variables themselves can be caused by various things such as additions, omissions, or specialization and generalization of variables in the previous survey, causing variables with certain coding to be shifted to another coding. For example, questions regarding the age of respondents in the 2015 SAKERNAS are stored in the B4_K5 variable, while in 2018 they are stored in the B4_K8 variable due to new questions in that year.

As the background of this research, the data search algorithm itself has been previously developed by [11] which can be seen in **Figure 1**. This algorithm will then be referred to as the basic search algorithm. In this algorithm, the data search process can be grouped into two major stages, namely the variable preparation stage (a) and the data search stage (b). At the variable preparation stage, a variable matching process is carried out that will be used in the input query with a list of coding columns in the variable metadata of each version considering the coding differences that can occur in each version. The list of columns obtained is then stored for later use in reordering the query so that it can be applied to all appropriate versions.

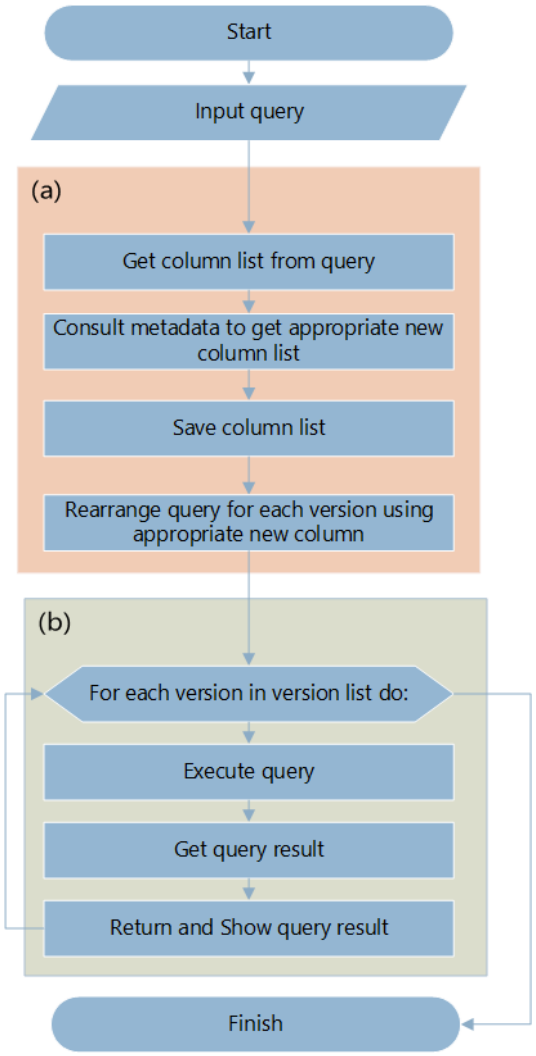

Figure 1. Basic algorithm based on [11]

Research on [11,13] using a basic query, without optimization. So in this research, we use query optimization methods, those are an indexed query and covered query [20].

2.1. Indexed Query

An index is a special data structure that is used to store a small part of data collection so that a system can be easily navigated for the purpose of retrieval of data. Data indexation in MongoDB itself uses a B-Tree structure, which allows indexed data retrieval queries (indexed queries) to scan only the index that has been created without having to review the entire document, thereby easing the burden on the database and the applications that access it.

2.2. Covered Query

Query Covering is one of the optimization methods in MongoDB that utilizes the index from the database that has been provided. A query can be said to be covered by an index if: all fields used in the query are part of an index; all projected fields are part of the same index; and none of the fields in the query will be null. In processing, the covered query no longer needs to check any original documents in the database because the required fields are already stored in the existing index.

Experiments conducted in this research are the data collection algorithm modification, to be proposed algorithm, and also implement query optimization methods. We will compare these proposed methods to the previous algorithm on query running time and the number of queries executed to evaluate them. This research was conducted using the 2015 and 2018 National Labor Force Survey (SAKERNAS) data with selected areas of Lampung, DKI Jakarta, DI Yogyakarta, Bali, and South Sulawesi provinces. This research flow can be seen in **Figure 2**.

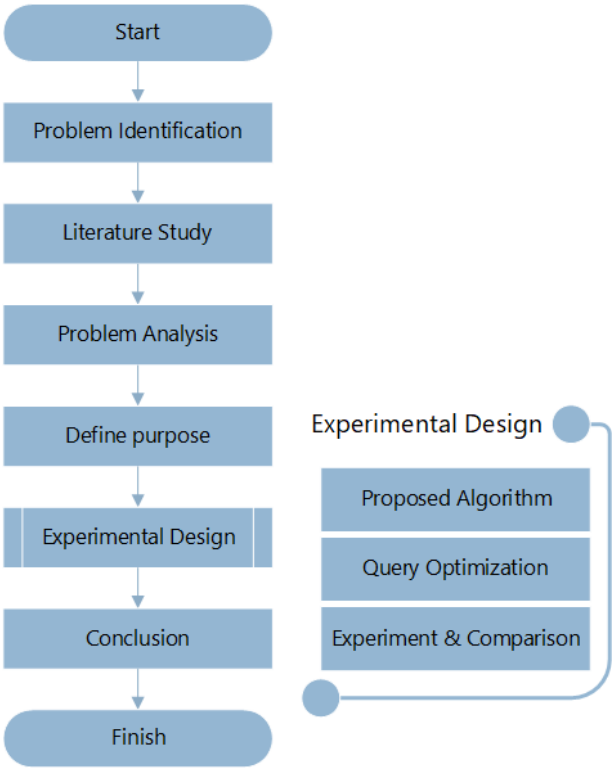

Figure 2. Research Flow

3. Result and Discussion

R. Munir [18] emphasized that a good algorithm is an efficient algorithm, or in other words, minimizes the need for time and space. He adds that the time and space requirements of an algorithm depend on the size of the input, which is typically the amount of data processed. If it is linked to the algorithm in **Figure 3**, the input size in the form of steps performed at the variable (a) preparation

stage itself will increase, considering that there are queries that need to be rearranged for each version with the corresponding new column.

From the description above, the author considers that the algorithm can be made more effective if the variable preparation stage (a) can be omitted so that there are fewer steps that must be done, increasing the effectiveness of the algorithm. In its implementation, omitting this step will cause the query that was inputted at the beginning can be used for every query of the required version, and later the process of repeating the query compilation can be avoided.

As for this research, the authors use it as a means to propose a method to eliminate the variable preparation stage (a) in the algorithm that has been developed in [11]. However, the method proposed by the author places more emphasis on changing the structure of the survey data storage itself. The BPS survey data storage algorithm currently used to store data is still simple. First, the raw data from the BPS survey will be collected in a data lake, which in this case utilizes HDFS. This data lake is used by BPS as a temporary data storage place, considering that the stored data is still in the form of raw data, and cannot be directly used for data analysis needs. Therefore, the data will be explored and transformed first. After passing through the transformation and exploration stages, the data is stored in a data warehouse. In this storage, the data stored is structured, although in this case, the storage structure used is still in a relational form, considering that BPS still uses Microsoft SQL Server and DB2 BLU as a data warehouse for data storage. The algorithm, which will later be referred to as the basic storage algorithm, can be visualized in **Figure 3**.

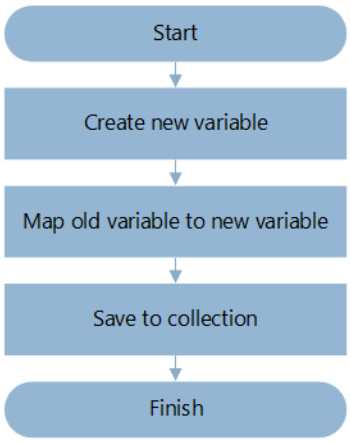

Figure 3. Basic algorithm to store data based on [19]

The changes to the storage structure proposed by the author are carried out before the data is stored in their respective data warehouses and applied to all versions of the existing survey results storage structure. Changes in the structure are carried out by classifying each variable containing the answers to the same questions in each version of the data into variables with the same coding for each data collection. Variables with the same coding will then be referred to as classification variables. For example, a question regarding the respondent's age would be coded as the AGE variable in each version. Because this classification process is carried out before the data is stored, this process only needs to be done once on all survey variables that contain answers to questions that have been asked. If in the future there are new variables containing answers to questions that have not been asked in the previous survey version, then the classification variable with the same name can be added directly to the data with a structure containing only these variables, without changing the data structure of the survey version previously.

The application of variable coding classification to this storage structure will later affect the data search process, where the variables that will be used in the inputted query will be directly used by all existing versions of the data. The data storage algorithm after the implementation of the variable classification process can be seen in **Figure 4** and produces a snowballing effect on the data search algorithm which is shorter than the basic algorithm in **Figure 5**.

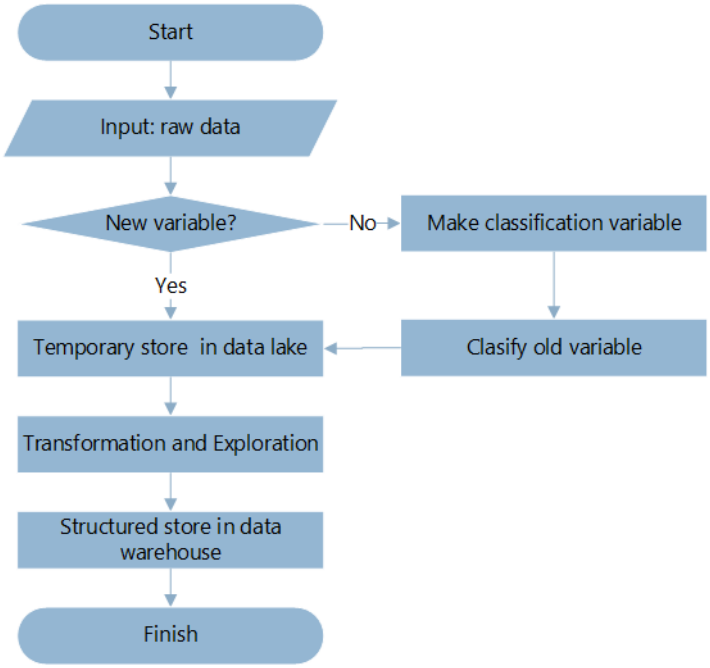

Figure 4. Proposed algorithm to store data.

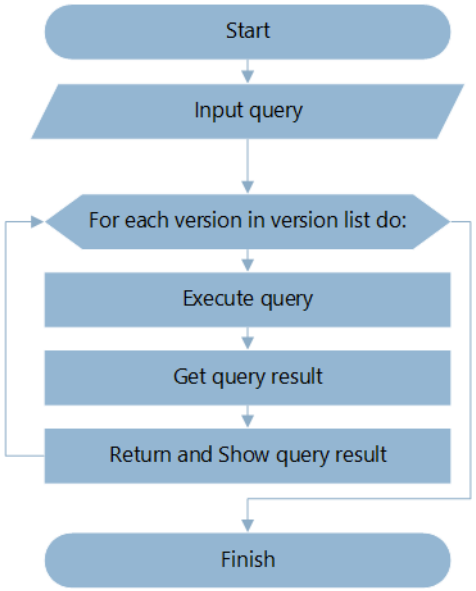

Figure 5. Proposed algorithm to search data.

In this case, the author tries to create a database in the form of a metadata collection that contains the mapping of each variable with the same data content from different years into a single document containing the name of the mapping variable, the description of the mapping variable, and its mapping to the actual data document based on each existing version. The collection containing the metadata will later be used as a bridge from the collection containing the actual data and information related to changes in the coding of variables in the data collection of each survey version. The metadata collection in question can be seen as an example of a snippet in **Figure 6** and a collection containing actual data can be seen as an example of a snippet in **Figure 7**.

```
[
    {"var_name":"UMUR",
        "var_ket":"UMUR RESPONDEN",
        "tahun":["2015","2018"],
        "data":[
            "2015":["B4_K5"],
            "2018":["B4_K8"]
            ]},
    {"var_name":"HUB_KRT",
        "var_ket":"HUBUNGAN RESPONDEN DG KRT",
        "tahun":["2015","2018"],
        "data":[
            "2015":["B4_K3"],
            "2018":["B4_K3"]
            ]},
    {"var_name":"JENIS_KELAMIN",
        "var_ket":"JENIS KELAMIN RESPONDEN",
        "tahun":["2015","2018"],
        "data":[
            "2015":["B4_K6"],
            "2018":["B4_K4"]
            ]},
    ...
```


Figure 6. Metadata snippet used in the basic algorithm simulation.


```
[
    {
        "URUTAN": 208482,
        "FINAL_WEIG": 39,
        "KODE_PROV": 31,
        "NAMA_PROV": "DKI JAKARTA",
        "KODE_KAB": 1,
        "NAMA_KAB": "KEPULAUAN SERIBU",
        "KLASIFIKAS": 1,
        "B2_R1": 7,
        "B2_R2": 7,
        "B4_K1": 7,
        "B4_K3": 7,
        "B4_K6": 2,
        "B4_K8": 78,
        "B4_K9": 3,
        "B4_K10": 4,
        ...
        },
    ...
```


Figure 7. Actual data snippet used in the simulation of the basic algorithm.

Furthermore, to compare the basic algorithm and the proposed algorithm, the authors conducted a data filtering trial using the two algorithms. The filtering test is carried out by repeating 1,000 times, and then comparing the average and standard deviation of the time used, starting from the query input process containing the filters that will be used until the data is displayed.

Following the initial goal which intends to compare the two algorithms, which in this case are the basic algorithm and the proposed algorithm, the author uses two kinds of queries, the first is a query with column coding that is still not the same for each version and the second is a query with column coding which has been equalized. In this case, the two queries are queried on different databases due to different coding structures but have the same data. As the context for the explanation of the query that will be used, the database used by the author is named SAK2018 for 2018 Sakernas data with initial coding, SAK2015 for 2015 Sakernas data with initial encoding, and SAK2018_v2 and SAK2015_v2, respectively, for 2018 and 2015 Sakernas data, respectively. generalized coding.

In this case, a simple query is used to display the number of rows returned from the search results without displaying each row. The author will compare it with its SQL form to be able to easily describe what kind of query filtering is used. The following is the form of a query on a database with coding that has not been generalized if it is queried using SQL:

- SELECT COUNT(*) FROM SAK2018 WHERE (B4_K8>20 AND B4_K6=1 AND B5_R1A>7)
- SELECT COUNT(*) FROM SAK2015 WHERE (B4_K5>20 AND B4_K4=1 AND B5_R1A>7)

The two queries above, if translated into everyday language, are to filter data from the survey results database with the criteria of being more than 20 years old, male, and having at least a high school education. The three criteria are equally applied to two different databases, namely SAK2018 and SAK2015. Although the criteria are the same, the use of column names from the two databases is different. This aligns with coding that is still not generalized and requires the author to rematch the desired criteria with the list of columns that exist in each version. It can be said that the use of this query is a worthy example for simulating existing algorithms.

The following second form of query is executed on a database whose coding has been generalized, namely:

- SELECT COUNT(*) FROM SAK2018_v2 WHERE (AGE>20 AND GENDER=1 AND EDUCATION_END>7)
- SELECT COUNT(*) FROM SAK2015_v2 WHERE (AGE>20 AND GENDER=1 AND EDUCATION_END_END>7)

The two queries above have the same output function as the previous query. The difference only lies in the column names that are already evenly distributed in both databases that store the data to be retrieved. This causes the author does not to need to perform repeated matching for each database, and it can be said that it is feasible to simulate the proposed algorithm.

The graph in **Figure 8** shows that the time used to display the results by the proposed algorithm has decreased compared to the basic algorithm. The average time used to run the basic algorithm is 816.43 milliseconds, while the time used to run the proposed algorithm is an average of 543.99 milliseconds.

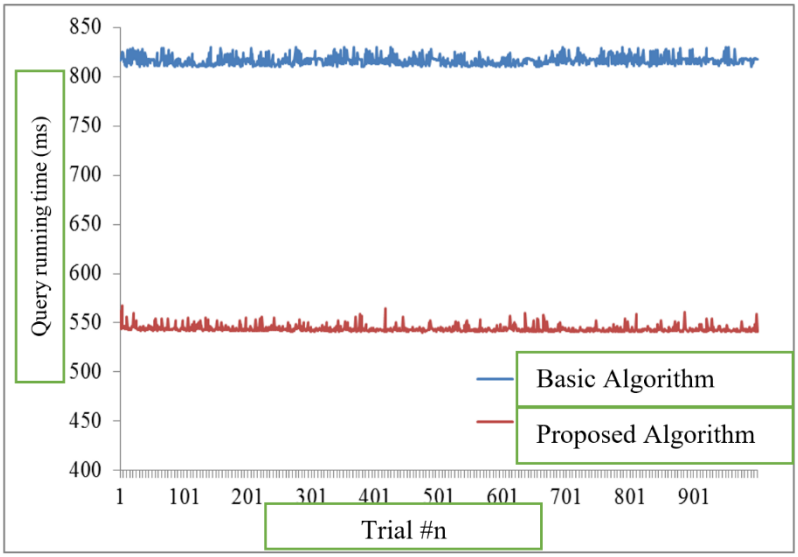

Figure 8. Query Running Time Comparison based on search algorithm.

The process of searching and filtering data stored in a database can be further optimized by the query optimization method. As has been applied in [16], several query optimization methods can be applied in the data search process using MongoDB. Therefore, in the next step, the author tries to apply optimization steps to the search query used. The optimization method that has been applied by the author in this study is the method of query indexing and query covering.

The queries used in searching the data, either before or after the application of the query indexing and query covering methods are the same. The use of the index in a search query is used by defining the index to be used first, which in this study can be done by utilizing the native features of MongoDB itself.

Like the definition of an index in general, this process is carried out before the query is executed. MongoDB itself has the cursor.explain and db.collection.explain() functions that can help define indexes. Both functions help by providing the best index plan for the query to be executed. For query covering itself, it is an advanced method of indexing, where all the keys used in the query must have their respective indexes. Query covering itself is intended so that the MongoDB application does not need to access the original data at all, and only accesses the index that has been defined.

To measure the optimization of the query used, the author compares the time and number of queries/second in the iteration process of data collection which is carried out 1000 times, then the average and standard deviation of the time used are taken. Regarding this, the result of each type of query is the printing of each row of search and filtering results. To describe the query used, here's what it looks like when translated in SQL:

- SELECT (KAB_NAME, KAB_KODE, B4_K3, B4_K8, B5_R1A) FROM SAK2018 WHERE (B4_K8>20, B4_K6=1, B5_R1A>7)

It can be seen that the above query has differences from the queries in the previous experiment. This is because the desired output is all rows, although not all columns are printed to reduce the time and energy used in this experiment. The results of the comparison of the time used can be seen in **Figure 9.**

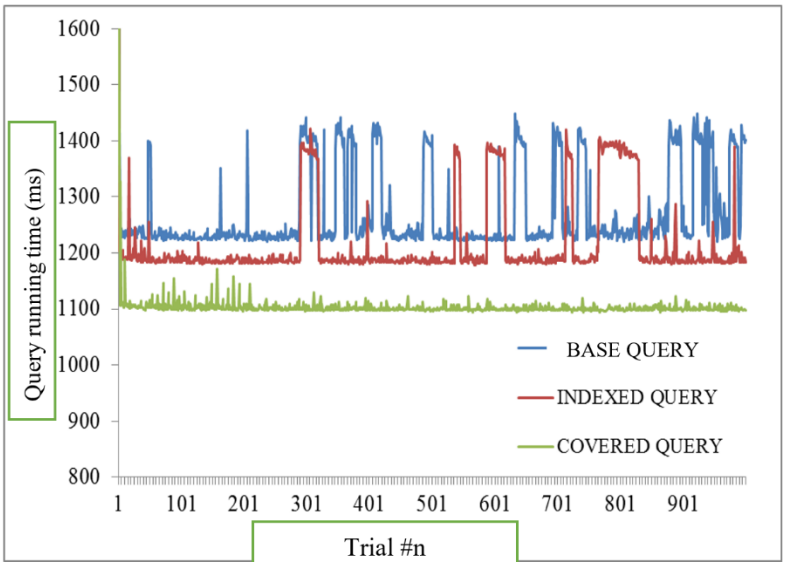

Figure 9. Comparison of query running time based on query type.

In **Figure 9**, it can be seen that each method applied tends to reduce the time used to print query results. The basic query used at the beginning has an average printing time of 1,271.24 milliseconds. In queries, the application of the query indexing method has an average printing time of 1,217.82 milliseconds, while the application of the query covering method on the basic query has an average printing time of 1,103.25 milliseconds. So far, the query covering method can be said to be more optimal than other types of queries.

Figure 9 also shows that the printing time at the beginning and end of the iteration tends to be higher, this happens because the power used on the MongoDB server-side has not been optimally deployed, and is in the process of deploying the server memory to perform query processing more per second. It can be seen from **Figure 10** which shows the number of queries performed per second by each query, that the application of the query covering method tends to process more queries than the other two. This also causes the printing of the results of these queries to tend to be faster than others.

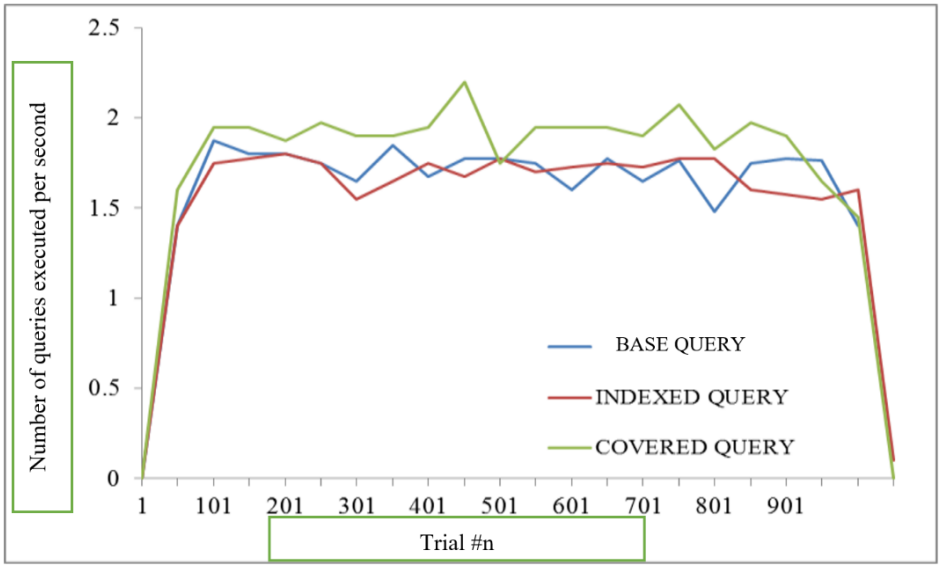

Figure 10. Comparison number of queries executed per second based on query type.

After knowing that the query covering method tends to support the optimization of the data search algorithm, the author performs a comparison again using a query to display the number of rows from the search results, and then juxtaposed with the results of a simple query using the proposed algorithm. The results of the comparison of the second stage can be seen in **Figure 11**, while the comparison of the mean and standard deviation of the time used by the three experiments to obtain the same results can be seen in **Table 1.**

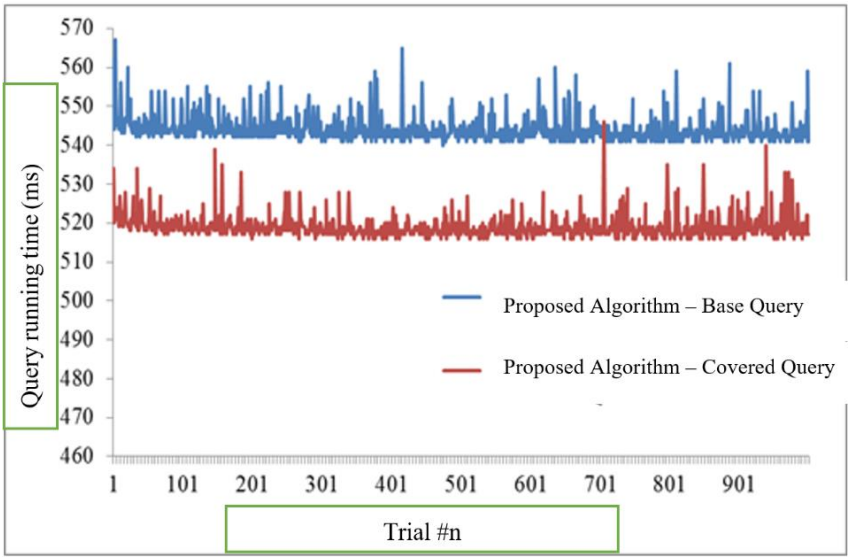

Figure 11. Comparison of query running time on the proposed algorithm based on query type.

Table 1. Comparison of average and standard deviation query time.

Query Type	Average	Deviation Standard
Base Query on Basic Algorithm	816.43	4.71
Base Query on Proposed Algorithm	543.99	3.31
Covered Query on Proposed Algorithm	518.91	2.96

Referring to **Table 1**, it can be seen that the application of the proposed algorithm reduces the average time used to obtain the same result by 34 percent, or in other words, causes an acceleration in the average time to reach 50 percent. The standard deviation of the time used also decreased by 1.75 or about 60 percent. Seeing the reduction of the two indicators, namely the average and standard deviation of the time used, it can be said that the research related to the application of algorithms and optimization can be said to be successful.

4. Conclusions and Suggestions

Some conclusions that can be obtained in this study are as follows:

- The data search algorithm that has been developed in previous studies can still be optimized further in terms of the time required to run search queries.
- The optimization of the algorithms carried out in this study focuses on the submission of new storage algorithms and optimization in terms of query usage.
- The use of data search time using the algorithm proposed by the author in this study is less than the algorithm that has been developed by previous studies, where there is an average acceleration of 50 percent.
- The query covering method applied as an advanced optimization process can be said to be effective in supporting the data search algorithm optimization process.
- In general, research related to the optimization of this algorithm can be said to be successful because the time required, both in the process of storing and retrieving data, has been reduced.

The suggestions from writing that can be done in the future to develop the results of this research are as follows:

- Designing an effective variable classification algorithm and can be used as a reference in the data storage process.
- Measure the time used in the variable classification process which is carried out in the explicit data storage process.
- Conduct further research on the storage of structural differences due to specialization or generalization of variables.
- Implement the use of non-relational databases as an alternative for storing data from BPS survey results in the future.

References

[1] W H Inmon 2002 Building the Data Warehouse,3rd Edition (3rd ed) (John Wiley & Sons, Inc, USA)

[2] E A Brewer, Towards robust distributed systems, *in Proceedings of the Nineteenth Annual ACM Symposium on Principles of Distributed Computing*, 2000, p 7, DOI: 101145/343477343502

[3] E A Brewer, CAP twelve years later: How the rules have changed, *Computer (Long Beach Calif)*, vol 45, no 2, pp 23–29, 2012, DOI: 101109/mc201237

[4] M O Fitri, Trend Penggunaan NoSQL Untuk Basis Data Non-Relasional, *J Teknosains*, vol 7 Nomor 1, pp 120–127, 2013

[5] "HDFS Architecture" https://hadoopapacheorg/docs/r121/hdfs_designhtml

[6] Microsoft, "Microsoft SQL Server," [Online] Available:https://wwwmicrosoftcom/en-us/sql-server

[7] IBM, "IBM Db2," [Online] Available: wwwibmcom/analytics/db2

[8] C Gyorödi, R Gyorödi, and R Sotoc, A Comparative Study of Relational and Non-Relational Database Models in a Web-Based Application, *Int J Adv Comput Sci Appl*, vol 6, no 11, 2015, DOI: 1014569/ijacsa2015061111

[9] C De Lima and R D S Mello, A workload-driven logical design approach for NoSQL document databases, *17th Int Conf Inf Integr Web-Based Appl Serv iiWAS 2015 - Proc*, no February 2019, 2015, DOI: 101145/28371852837218

[10] M A Rahman and L R Maghfiroh, Kajian Database Relasional dan Nonrelasional pada Data BPS dalam Menghadapi Era Big Data (Relational and Nonrelational Database Study on BPS-Statistics Indonesia Data in Facing Era of Big Data), *Proceedings of the National Seminar on Official Statistics* pp 25–30, 2019 https://doi.org/10.34123/semnasoffstat.v2019i1.170
[11] L R Maghfiroh and I G B B Nugraha, Survey data and metadata modeling using document-oriented NoSQL, *J Phys Conf Ser*, vol 971, no 1, 2018, DOI: 101088/1742-6596/971/1/012030
[12] Y. Li and S. Manoharan, A performance comparison of SQL and NoSQL databases, *2013 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM)*, 2013, pp. 15-19, doi: 10.1109/PACRIM.2013.6625441.
[13] L R Maghfiroh and I Santoso, NoSQL Model Data Warehouse Metadata Survei Dinamis. *Jurnal Aplikasi Statistika & Komputasi Statistik*, [S.l.], v. 11, n. 2, p. 55-74, sep. 2020. ISSN 2615-1367. Available at: <https://jurnal.stis.ac.id/index.php/jurnalasks/article/view/220>. Date accessed: 08 oct. 2021. doi: https://doi.org/10.34123/jurnalasks.v11i2.220.
[14] F G Tinetti, F Paez, L I Aita, and D Barry, Distributed Search on Large NoSQL Databases, *Proc PDPTA*, pp 685–690, 2011
[15] Y Gu, X Wang, S Shen, S Ji, and J Wang, Analysis of data replication mechanism in NoSQL database MongoDB, *2015 IEEE Int Conf Consum Electron - Taiwan, ICCE-TW 2015*, pp 66–67, 2015, doi: 101109/ICCE-TW20157217033
[16] D Mahajan, C Blakeney, and Z Zong, *Improving the energy efficiency of relational and NoSQL databases via query optimizations*, *Sustain Comput Informatics Syst*, *vol 22*, pp 120–133, 2019, DOI: 101016/jsuscom201901017
[17] P Cash, T Stanković, and M Štorga, *Experimental design research: Approaches, perspectives, applications*, no July 2016
[18] R Munir, *Matematika Diskrit* 2010
[19] A Koswara, "*Dokumentasi pengembang Web Portal INDAH*" 2021
[20] MongoDB, "Query Optimization" https://docsmongodbcom/manual/core/query-optimization/ [online], accesed August 25, 2021