Implementation of Heuristic Principles for Index Recommendations using Java in the MariaDB Database

Radityo P. Wibowo[1], Risa Perdana Sujanawati[1], Faizal Mahananto[1] and Prasasti Karunia Farista Ananto[1]

[1]*Information System Departement, Faculty of Information and Communication Technology, Institut Teknologi Sepuluh Nopember (ITS)*

Keywords: Index, Database Performance, Java, MariaDB.

Abstract: Almost all fields in life today use database and demanding the most optimal performance from a database. Therefore, a database must have the ability to optimize performance that can be obtained one of them by choosing an optimal index because the index plays an important role in determining the performance of queries that are executed against the system. The right index can increase speed in data processing. According to previous research, Index can improve database performance, namely the number of transactions per minute which increases three times and the cost is reduced by about 50% to 90%. That way index selection is a dilemma for database users. The output of the following research is a Java-based software that can be used in the MariaDB Database to obtain recommended indexes based on heuristic principles that have been used by previous researchers. The following software can evaluate database performance based on response time and Queries per Second or QPS. The recommendations from the following research can improve the performance of the Monitordom database by 43.65% and the Adventureworks 2012 database to 2.77%.

1 INTRODUCTION

The database is an integral part of our daily lives that we use unconsciously (Connolly and Begg, 2005). Database performance is often an important measurement related to user satisfaction. The performance of a database depends on the logical structure, physical structure and design of the database. So that the better the logical structure, physical structure and database design, the better the performance of the database (Osman and Knottenbelt, 2012). Almost all fields in life today have used databases and demanded the most optimal performance from a database. Therefore, a database must have the ability to optimize performance that can be obtained, one of which is by selecting an optimal index.

Index is an orderly arrangement that is logically used to sort rows in a table (Connolly and Begg, 2005). index plays an important role in determining the performance of queries executed against the system. (Chaudhuri et al., 2004). The right index can increase speed in data processing (Ameri et al., 2015). Misunderstanding often occurs, the more indexes the better, where it can increase database maintenance such as storage and does not provide optimal database performance results (Winand, 2012). The more indexes the more storage will be used, and this will have a negative effect on the performance of the database. The solution to overcome this is to choose the optimal index as proven in a database performance measured by transaction events per minute, can increase up to three times and can reduce the cost of a workload of 50% to 90% (Pedrozo and Vaz, 2014) (Valentin et al., 2000), so the index selection with a certain method needs to be done to get optimal performance improvement.

There are several index selection methods according to previous research, one of which is by using an algorithm developed based on the heuristic method and using the naïve probability method in the algorithm (S. Choenni, 1993) (Chaudhuri and Narasayya, 1997) (Chaudhuri et al., 2004). Almost all research for index recommendations is to use the heuristic method principle by developing an algorithm that is oriented towards optimal results.

Some DBMS have provided a default feature for index recommendations including SQL server (Chaudhuri et al., 2004). Various studies on index optimization with several other DBMS cases have also been developed such as AISIO for PostgreSQL

Wibowo, R., Sujanawati, R., Mahananto, F. and Ananto, P.
Implementation of Heuristic Principles for Index Recommendations using Java in the MariaDB Database.
DOI: 10.5220/0009906200920097
In *Proceedings of the International Conferences on Information System and Technology (CONRIST 2019)*, pages 92-97
ISBN: 978-989-758-453-4

8.4 (Pedrozo and Vaz, 2014), other recommendation index studies make MySQL 5.5 a trial and implementation. Even no exception for NoSQL databases such as MongoDB has become the object of research for index recommendations using mining algorithms (Ameri et al., 2015). However, open source such as Maria DB does not yet have a default feature index recommendation and there has not been much research on recommendations with the MariaDB case study. One way to overcome the problem of index selection on MariaDB can be developed by implementing heuristic principles using Java.

Java is an object-based programming language (Hermawan, 2004). Java is also multiplatform which can be interpreted for various operating systems so that java will be used by various operating system users. Java has a pattern matcher method such as Regular Expression which can filter the SQL you want to optimize.

This research is intended to make a software as a java-based index recommendation by implementing a heuristic method that has an index selection algorithm as previously done ((Chaudhuri et al., 2004)but what differs in the following research is in the index selection process using the General log or can be in the form of query as input and provide an output in the form of a recommended column name as an index as a result of the selection process carried out in the Maria DB database.

2 BACKGROUND

2.1 Index Selection

Logical and description, which is designed to meet the information needs of an organization (Connolly and Begg, 2005). Meanwhile, to control the database users can use a DBMS that is the Database Management System. (Connolly and Begg, 2005). According to Conolly, Database has three design phases namely Conceptual, Logical, and Physical. While the tools of this research are in the scope of the Physical Design phase where the output is intended to help the physical design database process. Performance measurement from the database can be done by assessing several aspects according to winand, as follows (Winand, 2012): Data Volume, System Load, and Response Time and Throughput. In this study will use measurements in the form of response time and throughput which can be interpreted as Query per Second (QPS) that calculate from average query execution in iteration of 60 seconds.

Index is a data structure that organizes data records in storage to optimize certain types of retrieval operations (Raghu and Gehrke, 2004). The right index can increase speed in data processing (Ameri et al., 2015). The basis of the algorithm in the following research is the AISIO heuristic principle (Pedrozo and Vaz, 2014) where in the selection of the index has four main processes but there are some details that are still not very clear, such as when making single and multi-level-index candidates so that they can use some other research references such as the index selection process on SQL server (Chaudhuri and Narasayya, 1997) is by doing permutations for column names, and obtaining heuristics that are in accordance with MariaDB and applied in the following research:

1. Parsing and Filter Queries. To get the initial index candidates in the form of the names of the columns contained in the input as did all research on the previous index recommendations.
2. Retrieving previous indexes. The following is an addition to this study because in previous studies the existing index was rarely considered, even though the index could be an optimal index.
3. Identifying index Configuration. After getting an index candidate in the form of a column name the next is to make an index configuration or in the form of a set of indexes based on a combination of column names so that a multi-level-index will appear like the main heuristic and making this configuration adheres to the details of making the configuration set in research on sql server (Chaudhuri and Narasayya, 1997) which found that the optimal combination is with the value j = 2, the combination is concerned with sequences so that it is done by permutation.
4. Configuration cost evaluation Cost evaluation is an evaluation activity with measurements in the form of QPS or Response time which is performed to find the best performance of all index configurations prior to enumeration as in previous studies (Chaudhuri and Narasayya, 1997).
5. Configuration enumeration. After getting a set of index configurations with the best performance, the enumeration was done as in the research for sql server (Chaudhuri and Narasayya, 1997), but the selection of the best enumeration candidates was not based on the most fit storage but used the selection factor by selecting the candidates who most often appeared in the workload based on other studies (Ameri, 2016). This selection factor in other references is considered a selectivity factor calculated using the naïve probabil-

ity (Choenni, Blanken and Chang, 1993). To obtain better correlation probabilities this research enhances the selectivity factor using naïve bayes probability.

6. Generate recommended index query After getting the selected index results, the recommendation will be generated to be made to create an index query that is ready to be implemented in the database.

2.2 Data Sample

The data used in the following research is Monitordom and Adventureworks 2012 which represent two different database types, namely a database with a simple table structure condition and an existing index only as a foreign key, and other database conditions that have a complex table structure with a number of tables of 70 and more than one supporting index for each.

3 METHODOLOGY

Adopting previous heuristic algorithm method of this research is in Figure 1. Index Recommendation and more detail explanation is conducted in this section

3.1 Settings

When making the initial encoding settings done with java and using the library in the form of keneksi owned by MariaDB with java which can be obtained on the official MariaDB website openly (MariaDB, 2017). Next is to prepare the page for log and query input because the database admin can only choose one of these inputs.

The main purpose of this coding is to be able to connect to the database and determine the parsing steps to be carried out and starting from here is the application of the heuristic algorithm (Chaudhuri and Narasayya, 1997) from previous studies to the process of selecting selectivity factors.

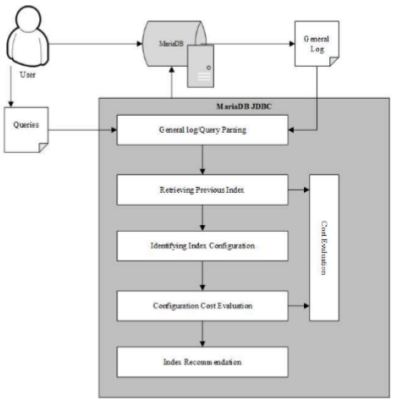

Figure 1: Index Recommendation

3.2 Parsing and Taking Index Beforehand

The process of retrieving column and table names when parsing is done using the JSqlParser library that is obtained openly on the GitHub forum (Binus university, 2016). Meanwhile, to display the previous index, a query is called which calls the index from informatics schema from the database.

3.3 Create Configuration Index and Selecting the Performance of the Best Configuration

ow to make configuration is to combine several column names that appear and the possibility of the index that occurs per configuration can be calculated with the following equation (1). Number of Configuration (Raghu and Gehrke, 2004):

$$\Sigma_i^n = 1\frac{n!}{(n! - j!)} \quad (1)$$

Where n is the number of columns that are index candidates and j is the number of different columns that you want to index. Due to the combination of index configurations that can be very numerous, as in the study of chauduri (Chaudhuri and Narasayya, 1997) that the most effective number of column combinations in an index is 2 so j = 2. Cost Evaluation itself is calculated based on Response time and throughput in the form of Queries per Second.

3.4 Configuration Enumeration

The enumeration configuration that is dividing the index candidates from the index configuration into units

and selecting candidate units using the naïve bayes method as in equation (2). Selectivity factor:

$$Sf(Ij|S) = \frac{P(S|Ij)P(Ij)}{P(S)} \quad (2)$$

Where S is the number of query statements contained in the workload and Ij is the probability of the index candidate appearing in the workload against other index candidates. Index candidates with the highest Sf will be the index to be recommended (Ameri, 2016). But beforehand it was determined in advance the type of cluster or non-clustered it.

3.5 Create Index Recommendation Query

Last is to display the index recommendation in the form of a query that has been made based on the results of the index recommendation. In addition to displaying the recommended index, this section will show the results of increasing the percentage of performance by calculating in equation (3). QPS Evaluation and equation (4). Response time Evaluation. The two assessments of improvement are different because for QPS the more the better and the less the Response time the better.

$$QPS = \frac{QPS(rec) - QP(initial)}{QPS(initial)x100\%} \quad (3)$$

$$Responsetime(RT) = \frac{RT(rec) - RT(initial)}{RT(initial)x100\%} \quad (4)$$

4 RESULT AND DISCUSSION

4.1 Implementation Environment

Implementation and result of this research depended on the environment since performance was influenced by hardware. This research was developed and implemented in the environment type Sony SVF14319SGB, Processor Intel Core i5, RAM 8GB, Hard Disk Drive 1000G. And software environment in this research is MariaDB 10.1.19 as Database and NetBeans IDE 8.2 for development process of application.

4.2 Monitordom Result

The index recommendation system uses response time, each query executed is iterated 50 times to get the stability of the average response time. Stability is tested by testing several times with the same environmental conditions. With 50 iterations, the system can provide consistent recommendations. Result of response time increase in average up to 30.84%. Ressult from monitordom in Figure 2 shows using this recommendation index, the responsetime decrease from average 0.020 to 0.015 second.

Queries in Monitordom are executed in only one table using similar pattern that is using type 'SELECT' and operator like 'GROUP BY', 'ORDER BY', 'DISTINC', and 'COUNT'. Result of Query Persecond (QPS) in this database sample give significant number increase after recommendation as shows in Figure 3.

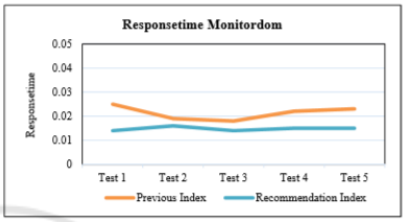

Figure 2: Responsetime result from Monitordom

	Tablename	Column in Query	Keyword	Using Previous Index QPS	Using Recommendation Index QPS
Q1	webstatus	1	ORDER BY, DESC, LIMIT	84.9	833.3
Q2	webstatus	1	COUNT, DISTINCT	59.2	128.2
Q3	webstatus	1	DISTINTC, ORDER BY, ASC, LIMIT	41.6	91.4

Figure 3: QPS Result Monitordom.

4.3 Adventureworks2013 Result

Testing conducted with same method as database sample before in Adventureworks2012 resulting increase on average response time 1.26% and detail each test shows in Figure 2. Tests also carried out using QPS measurements performance increase on average is 9% in queries that executed in AdventureWorks2012 that detailed in Figure 5. The queries is quite different from Monitordom because it has more complex operator and executed in seven tables that representing sales and marketing activities. The insignificant increase even decrease performance is due to AdventureWorks is a database that has an index that is in accordance with their needs. This is also evidenced in the AdventureWorks 2012 Data Dictionary documentation (sourceforge, 2009) regarding which indexes are already owned. Performance also found decreases in one of the response time test because the new index created does not actually help search

but burdens database work because the Adventureworks2012 database already has an index with column names as recommended by the following software.

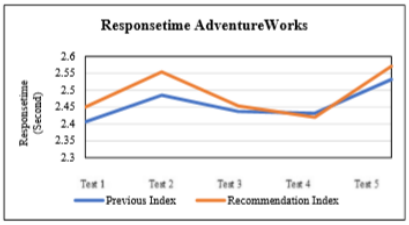

Figure 4: Responsetime result from AdventureWorks

	Tablename	Column in Query	Keyword	Using Previous Index QPS	Using Recommendation Index QPS
Q1	store, salesorderhead er	2	GROUP BY, ORDER BY, DESC	4,517	4,683
Q2	store, salesorderhead er	4	GROUP BY, ORDER BY, DESC	1,550	1,741
Q3	store, salesorderhead er, salesterritory	3	GROUP BY, ORDER BY, DESC	4,433	4,286
Q4	product, productmodel , productsubcat egory, productcateg ory, salesorderdet ail, salesorderhea der, store	5	GROUP BY, ORDER BY, ASC	0,050	0,067
Q5	store, salesorderhea der	2	GROUP BY, ORDER BY, ASC	5,867	6,142
Q6	store, salesorderhea der	2	COUNT, GROUP BY, ORDER BY, DESC	0,483	0,483

Figure 5: QPS Result AdventureWorks.

5 CONCLUSIONS

This section explains what the conclusion that was made from this research and author's suggestion for further research. After conducting the implementation and testing of the research "Implementation of Heuristic Principles for Index Recommendations using Java on MariaDB" using the exact environment explained, it can be concluded that performance of a database according to response time using the index of recommendations from the following studies can increase by around 1.26% in database that already indexed before and up to 30.84% in database that has not indexed before. according to QPS can increase by around 9% in database that already indexed before and up to four times faster in database that has not indexed before.

From the two sample databases used for implementation, it can be concluded that the selection of indexes with the following heuristic principles works better on databases that do not yet have indexes supporting operational activities and query type 'SELECT' that used operator like ''ORDER BY', 'GROUB BY', 'DISTINC' and 'COUNT'.

This study also found that the measurement of the response value of a query that was executed had the best value for 50 times. The results of these values are taken from the results of the experiment for several times to get consistent recommendations, so that with consistent recommendations it can be said that the average response time has been stable.

Author's suggestions that can be done for further research are develop an application by adding features such as combining QPS and response time assessments for example by using linear regression equations or other metrics assessments. Pay attention to the database structure such as the storage engine used in order to determine recommendations in more detail such as the type of index cluster or non-cluster.

Implement other algorithms in the selection of index recommendations such as Genetic algorithm to get better performance. This research could be reference to perform index selection algorithms implementation in other programming languages or other database objects.

Algorithm improvement also might improve by applying the configuration of the table name not only configuring column name in the step of creating candidate of index recommendations.

REFERENCES

Ameri, P. (2016). On a self-tuning index recommendation approach for databases. In *2016 IEEE 32nd International Conference on Data Engineering Workshops (ICDEW)*, pages 201–205. IEEE.

Ameri, P., Meyer, J., and Streit, A. (2015). On a new approach to the index selection problem using mining algorithms. In *2015 IEEE International Conference on Big Data (Big Data)*, pages 2801–2810. IEEE.

Binus university, W. (2016). Pengertian methode, class dan objek dalam oop.

Chaudhuri, S., Datar, M., and Narasayya, V. (2004). Index selection for databases: A hardness study and a principled heuristic solution. *IEEE Transactions on Knowledge and Data Engineering*, 16(11):1313–1323.

Chaudhuri, S. and Narasayya, V. R. (1997). An efficient, cost-driven index selection tool for microsoft sql server. In *VLDB*, volume 97, pages 146–155. Citeseer.

Connolly, T. M. and Begg, C. E. (2005). *Database systems: a practical approach to design, implementation, and management*. Pearson Education.

Hermawan, B. (2004). Menguasai java 2 & object oriented programming. *Yogyakarta: Penerbit Andi*.

MariaDB (2017). Mariadb connector/j 2.0.3 stable.

Osman, R. and Knottenbelt, W. J. (2012). Database system performance evaluation models: A survey. *Performance evaluation*, 69(10):471–493.

Pedrozo, W. G. and Vaz, M. S. M. G. (2014). A tool for automatic index selection in database management systems. In *2014 International Symposium on Computer, Consumer and Control*, pages 1061–1064. IEEE.

Raghu, R. and Gehrke, J. (2004). Sistem manajemen database edisi 3.

S. Choenni, H. B. o. T. C. (1993). Index selection relational database.

sourceforge (2009). Adventureworks database for mysql.

Valentin, G., Zuliani, M., Zilio, D. C., Lohman, G., and Skelley, A. (2000). Db2 advisor: An optimizer smart enough to recommend its own indexes. In *Proceedings of 16th International Conference on Data Engineering (Cat. No. 00CB37073)*, pages 101–110. IEEE.

Winand, M. (2012). *SQL performance explained: everything developers need to know about SQL performance;[covers all major SQL databases]*. M. Winand.