Index Effect on Data Manipulation Toward Database Performance

M Hayati\textsuperscript{1} and A Setyanto\textsuperscript{2}

\textsuperscript{1,2}Amikom Yogyakarta University, Yogyakarta, Indonesia.

E-mail: mardhiya_hayati@amikom.ac.id

Abstract. Database volume undergone an explosion due to intensive user and machine generated data. The volume growth slow down the processing speed. On the other hand the demand of real time information processing greatly desired. Therefore, speed up database processing becomes a key success factor for information processing. Indexing is a method to provide a sorted order of data content and a pointer to the physical location where data is stored. Effective indexing lead to better searching performance. This research focus on the observation of the effect of index to the performance of Data Manipulation Language statement (DML) on huge amount of data in the database. According to the evaluation, the test results shows a statistically significant difference between indexed compare to non-indexed database in all DML statement.

1. Introduction

Currently, the growth of data volume has achieve an incredible pace due to machine generated data and user involvement in data production. It causes a huge volume of data recorded in the database and as a consequence slower access speed. On the other side, the demands of providing a real time information processing become indispensable. Therefore, speed up of data operation in database processing is an unavoidable requirement for current database management systems. In general data processing in the database consists of inserting, deletion, updating, and data retrieving by using Structured Query Language (SQL). SQL\textsuperscript{[1]} is a set of commands used to access data in a relational database. SQL consists of several sub-types of commands, one of which is Data Manipulation Language (DML). DML is used to manipulate existing data in the database, DML commands consist of Insert, Update, Delete and Select.

The data access speed is indispensable in the presentation of an information.\textsuperscript{[2][3]. One of the important technique to optimize the processing speed is Indexing \textsuperscript{[4][5]. Indexing technique has been shown to improve database system performance in addition to an efficient database engine\textsuperscript{[6][7].}

There are several algorithms for index implementation such as B-tree, hB-Tree and T-tree. Previous research has been done by comparing execution time using 3 types of DML command (“Insert”, “Select” and “Delete”) command on B-tree and T-tree index algorithms. The results of
using B-tree algorithm is slower for small data than T-tree, but interestingly B-tree performs faster than T-Tree for large amount of data[8]. Another experiments [9] observed B-tree indexing technique using DML only for “Select” command. They tested some “select” statement with certain columns conditions in the query. The results that indexed columns as a condition of the “where” clause will perform better than the non-indexed column. Non-indexed column causes table scan so that execution time is longer, but not all indexed columns perform better, using “in” expression has a worse performance than the “between” However, does the index performance increase for the “select” command also has the same effect with other DML commands.

The purpose of this paper is to compare the query execution speed on indexed and non-indexed column. We are going to observe whether it causes a significant performance improvement or even decrease in DML (Data Manipulation Language) query operation. This research focuses on an evaluation of “insert”, “select”, “update”, “delete” command with search condition on indexed column, and it will be compared to non-indexed column.

2. Related Work

2.1. Basic Data Storage

A page or data block[9] is a storage place used in a database such as data rows including mapping the indexes used. Each page consists of 8KB (8192 bytes) of storage space and consists of 3 main components: page header, records, and offset array. All Page begins with a page header allocated a total of 96 bytes containing page information such as page numbers, type of page. In the second part, there is an area for storing data lines called records area (8060 bytes), the last part is the offset array area to store pointer position (36 bytes)[10].

Each added data row will occupy the recording area and the initial location of the page is stored in the last position of the offset array. As shown in figure 1

![Figure 1. Placement and Offset](image)

Page or data block has a limited storage capacity, therefore the data occupy multiple pages. Index Allocation Map (IAM) is in charge of organizing the existing pages.

2.2. About Index

Indexing aims to improve data search tasks in the database. By indexing technique, searching do not look at every single data row in the table, but searching process look up the data straight away to a given a key-index (pointer) on one certain columns. As a consequence, the searching process significantly faster to locate the desired data. To understand how indexes work, for example, the book searching in the library. First, you search book by book category. This section is analogous to a non-clustered index that serves as a guide where the book is located. Catalog of books are
placed by title and sorted in alphabetical order, instead of looking for a shelf with the title of the book beginning with the letter "A" better go straight to the shelf that you are looking for[11].

An index is an object in the database that serves to speed up data search. The index consists of column values and the key pointer. The key pointer will go directly to the place where the data is located. This technique enables to avoid full table-scan. The full table-scan process will search row by row in a table and stops if the desired data has been found. It takes a very long time, especially for large number of rows.

The indexing technique aims to reduce the computation complexity and I/O load on data retrieval process. However, in low-quality query design lead to infectivity of database engine to utilize the index. Instead of optimize the performance, bad query design lead the query performance getting worse[10].

Create an effective index has been done in various ways; create an index on the frequently used columns to sort the data, to group the data, the index is also given to the conditional expression column[2].

Table column can be alternatively used an index to improve performance but not all columns are effectively indexed, it will make decrease performance.

A Heuristic to guide the index selection process. A tool called AISIO (Automatic Index Selection Integrated into Optimizer) integrated to DBMS. AISIO will take SQL statement, analyzes them by heuristic rule, recommends potential index improve performance. AISIO tool has been integrated with PostgreSQL[12]

There are several components in the RDBMS that are used to optimize the "query" through the access management structure of indexes such as B-tree indexes, bitmap indexes, spatial indexes. Query Optimizer used to decide whether the use of the Index needs to be done or not[5]

2.3. B-tree

The B-tree is a generalized form of the Binary Search Tree data structure that has a node and can have more than two children [8]. The B-tree (balanced-tree) structure used for indexing pages has been organized into a hierarchical structure. [9]

The record storage hierarchy of the B-tree structure begins with root (root) [13] as the top part of a tree, the leaf as the lowest part of a tree, the part of the linked tree is called parent and child. The depth of a tree is indicated by \( n \) & \( (n + 1) \) in sequence. Each part of the record has been assigned a key value of a particular index and is always compared when adding a new record, the smaller index is always on the left and the larger index is on the right.

The number of records stored by B-tree with notification \( n = n=m^h-1 \), \( m \) is the number of keys owned by a block or page and \( h \) is the depth level of the tree. The number of children \( d = \lceil m / 2 \rceil \), thus yielding an equation of depth level of B-tree tree:

\[
h \leq \left\lfloor \log_m \left( \frac{n+1}{2} \right) \right\rfloor
\]
The number of records stored is exponentially proportional to the number of children owned and the length of the search process, addition, or deletion of records from a tree proportional to the depth level is denoted \( O(h) \) or \( O(\log_{(m/2)} ((n+1)/2 )) \) [8][9][13]

2.4. Database Time Measuring

The performance of the database system is measured by the transaction response time, the smaller the response time generated the performance of the database system the better. In database processing, a query is a job assigned by a user to a database via SQL syntax. Each time a SQL command is processed, the DBMS compiles, bind utility, and executes the query as shown in the following figure 2 [15]:

![Figure 2. State: SQL processing by DBMS](image)

There is a direct relationship between time and Tuples read (Tr). The more tuples the more time it takes to read the tuples (Tr)

2.5. Wilcoxon Test

Wilcoxon test is one of the non-parametric statistical tests in a population there are two independent samples, this test is applied if the measurement result is ordered ordinalally and possibly not normally distributed [13]. Non-parametric does not depend on rigid but more general and loose patterns or rules so suggest for data that is not normally distributed [14]. An independent sample used for testing comparative hypotheses or a test differences is also called test of significance. This test is to measure the level of significance in a case before treatment with after treatment, whether it has influence or not.

The Wilcoxon test takes into account the difference sign and the magnitude of the apparent difference between the two. [15] The value of the difference is sorted as a rank, if there is a similar value then the value is summed and taken an average. Wilcoxon test is calculated by the general equation [14]:

\[
T = \frac{(W - \mu_0)}{\sigma R} \\
\mu_0 = \frac{n(n+m+1)}{2} \\
\sigma R^2 = \frac{nm(n+m+1)}{12}
\]

\( W \) is the number of ratings of two samples that have been calculated, the result is selected the smallest value. \( \mu_0 \) dan \( \sigma R^2 \) is a population mean and standard deviation respectively. the \( n+m \) observations of the two independent samples are combined in a single dataset. The resulting \( T \) value is used to reject the null hypothesis (\( H_0 \)), the strength of rejection of \( H_0 \) by computing p-value is if and only if the p-value obtained is less than the selected level of significance [14].
3. Methodology

3.1. Experimental result
All these experiments were performed on Intel(R) Core (TM) i3CPU, clock processor 2.27Hz with 4.00Gb of main memory running in a Windows 10Pro operating system. All querys are executed on SQL server 2014.

3.2. Experimental methodology
The test is done by comparing the execution time of two tables i.e. table without index (T) with tables having index (Idx), each table has data of 5 million, 10 million, 15 million, 20 million, and 25 million records. Each table has 5 columns of the same data type "int", table without index (T) and table with index (Idx). Table structure with index (Idx) is shown in table 1.

Table 1. Idx table structure

| Column Name | Data Type | Index / No-Index |
|-------------|-----------|------------------|
| c1          | Int       | Index            |
| c2          | Int       | Index            |
| c3          | Int       | No-Index         |
| c4          | Int       | No-Index         |
| c5          | Int       | No-Index         |

4. Experiment and Result

4.1. Page count
It has obtained the data page owned by each table using sys.dm_db_index_physical_stats shown in table 2 below.

Table 2. Page count

| Number of Record | Index Level | Page Count |
|------------------|-------------|------------|
| 5.000.000 (T)    | 0           | 31.029     |
| 5.000.000 (Idx)  | 0           | 13.912     |
|                  | 1           | 44         |
|                  | 2           | 1          |
| 10.000.000 (T)   | 0           | 62.386     |
| 10.000.000 (Idx) | 0           | 27.939     |
|                  | 1           | 90         |
|                  | 2           | 1          |
| 15.000.000 (T)   | 0           | 93.048     |
| 15.000.000 (Idx) | 0           | 41.652     |
|                  | 1           | 132        |
|                  | 2           | 1          |
| 20.000.000 (T)   | 0           | 126.503    |
| 20.000.000 (Idx) | 0           | 56.611     |
The data stored stacked for each page into a table that does not have an index, and the table has a hierarchically arranged B-tree at the start with the root level (index level = 2), then the data is stored at the bottom of the leaf (index level = 0) is associated with parent or child in the middle level (index level = 1).

The table has no index has more pages than tables with index and only has a depth of 1 (only 1 level), but the table has an index of fewer pages and has more depth (3 levels), as shown in table 2.

### 4.2. DML clause Performance

Search data with scanning page system occurs in the table does not have an index, If the data has not been found then the possibility of accessing the entire page so it takes a long time.

Search by index starts from the root, from top to bottom, by accessing the indexed page of each record by comparing the indexed values searched with the current index position, the smaller indexs will be on the left and larger on the right, the number of pages accessed less because it has been through the process of comparing the position of the index value, then read the contents of the record, this process will save time because the number of pages accessed less than the search without index, execution time in the data search process shown in Figure 3 below.

|       |       |
|-------|-------|
| 25.000.000 (T) | 1      |
| 0     | 179    |
| 0     | 154.098|
| 25.000.000 (idx) | 0     |
| 0     | 69.305 |
| 1     | 218    |
| 2     | 1      |

**Figure 3. Query Performance**

The process of Changing and deleting data is almost identical to searching data, first doing a data search then changes or deletions. In the trial has done the data changed on the column that is not used an index with the search condition on the indexed column, so there is no rebalancing process in the tree. Execution time on data changes and data deletion is shown in figures 4 and 5 below.
Insert operations on tables that do not have index data arranged like a stack on a page, page capacity of 8kb, if the page is full, the data will save on the next page.

The insert process in the table has an index done by adding the index of a new record that will be stored into the tree. The insert process begins by searching for a vacant index position and placing it on a leaf node, the smaller index's value being left and larger on the right, the process completes if the number of indexes contained in the page is sufficient but otherwise will create pages and nodes new. The principle of this algorithm is to keep the tree a balanced condition and always reconstruct, this causes the insert process to be longer than the search data, this also applies to the update process whose data is the indexed column. The execution time of test results in some tables is shown in figure 6 below.
The hypothesis of this study is Is there a significant influence between the use of Index in the table to the execution time velocity for all data manipulation language process.

Wilcoxon test results in the speed of execution time of use of the index and without using the index in 4 categories of data manipulation that is inserted, update, delete and query are shown in the following table 3.

### Table 3. Wilcoxon Test Results

| Number of records | Query | Update | Delete | Insert |
|-------------------|-------|--------|--------|--------|
| 5,000,000         | 0.00  | 0.00   | 0.00   | 0.01   |
| 10,000,000        | 0.00  | 0.00   | 0.00   | 0.02   |
| 15,000,000        | 0.00  | 0.00   | 0.00   | 0.34   |
| 20,000,000        | 0.00  | 0.00   | 0.00   | 0.08   |
| 25,000,000        | 0.00  | 0.00   | 0.00   | 0.31   |

The real level of $\alpha=0.01$, the above data shows that $p$-value $<\alpha$ for category query, update, delete so $H_0$ rejected means there is a significant effect of index usage on response time, but the category insert receive $H_0$ means there is no effect of using the index to speed response time.

### 4.3. Wilcoxon test results

Figure 6. Insert Performance

### 5. Conclusion

The use of effective indexes increases response time for data retrieval process, and data modification by placing index columns as search condition column, but the use of the index is not effective on tables that often do data entry.
6. References

[1] N Soussi and M Bahaj 2017 *Semantics preserving SQL-to-SPARQL query translation for Nested Right and Left Outer Join* (Rev Mex Trastor Aliment) vol 15 no 5 pp 504-512

[2] M Wang, M Xiao, S Peng and G Liu 2017 “A hybrid index for temporal big data,” *Futur. Gener. Comput. Syst.*, vol. 72, pp. 264–272

[3] H Singh and S Bawa 2017 “A MapReduce-based scalable discovery and indexing of structured big data,” *Futur. Gener. Comput. Syst.*, vol. 73, pp. 32–43

[4] Nevarez bejamin 2015 *Query Tuning & Optimization*. united States of America: Mc Graw Hill

[5] T Morzy, R Wrembel, J Chmiel and A Wojciechowski 2012 *Time – HOBI: Index for optimizing star queries* *Inf Syst* vol 37 no 5 pp 412–429

[6] J Miguel, C D Barranco and O Pons 2017 *Indexing techniques to improve the performance of necessity-based fuzzy queries using classical indexing of RDBMS* (*Fuzzy Sets Syst*) vol 1 pp 1–18

[7] P S Patil 2012 *Customised Approach for Efficient Data Storing and Retrieving from University Database Using Repetitive Frequency Indexing* 2012 1st Int. Conf. Recent Adv. Inf. Technol. RAIT-2012 pp. 4–7

[8] W Guo and Z Hu 2010 *Memory database index optimization 2010* (Int. Conf. Comput. Intell. Softw. Eng CiSE 2010) pp 4 - 6

[9] C Qi 2016 *On index-based query in SQL Server database* *Chinese Control Conf CCC* vol 2016 - Augus pp 9519 - 9523

[10] J Strate and T Krueger 2012, *Expert Performance Indexing for SQL server 2012*. New York: Apress

[11] P Robert 2015 *Healthy SQL* New York: Apress

[12] W G Pedrozo and M S M G Vaz 2014 *A Tool for Automatic Index Selection in Database Management Systems* (2014 Int. Symp. Comput. Consum. Control) pp 1061–1064

[13] X Q Qiu and G S Jiang 2011 *Research about the index mechanism base on embedded database system* (Proc. - 2011 6th IEEE Jt. Int. Inf. Technol. Artif. Intell) Conf. ITAIC 2011 vol 1 pp 478-481

[14] J Perolat, I Couso, K Loquin, and O Strauss 2015 *Generalizing the Wilcoxon rank-sum test for interval data* (Int. J. Approx. Reason) vol 56 no PA pp 108-121