Journal of Computer Science

Case Reports

Query Optimization on Distributed Database Dengue Fever by Minimizing Attribute Involvement

[1]Slamet Sudaryanto Nurhendratno, [1]Sudaryanto, [1]Fikri Budiman and [2]Maryani Setyowati

[1]Faculty of Computer Science, Dian Nuswantoro University, Semarang, Indonesia
[2]Faculty of Health, Dian Nuswantoro University, Semarang, Indonesia

Article history
Received: 25-01-2018
Revised: 12-03-2018
Accepted: 05-04-2018

Corresponding Author:
Slamet Sudaryanto Nurhendratno
Faculty of Computer Science,
Dian Nuswantoro University,
Semarang, Indonesia
Email: slametalica301@dsn.dinus.ac.id

Abstract: Query optimization is an important task in a client/server environment of a distributed database, whereas a health epidemiologist data distribution based on DBD data on Geographic Information Systems (GIS). A proper method for a particular query process function is needed to generate query optimization on a distributed database. The query process requires important attention especially in distributed databases because the result of a cost-based query process is accessed by involving a number of attributes and visited sites. A query operation typically will search for data from various attributes in a scattered database table, although the processes do not require all table attributes. Query optimization requires minimum query operating costs (communication costs and access fees). The query cost can be optimized by separating attributes that are not required by the query. This can reduce the amount of communication and access time. The attributes should not be divided indiscriminately to obtain the best result of the query process and a vertical fragmentation method can be used to perform such attribute separation. In this research, attributes separation using vertical fragmentation method for a database health table is studied by comparing Bond Energy Algorithm (BEA) and Graphic Based Vertical Partitioning (GBVP) algorithm. The initial result of vertical fragmentation in both algorithms is the determination of types of attributes separated from a number of specific query process. The result of the separation of attributes from each algorithm is compared and evaluated using Partitioned Evaluator (PE) in order to achieve the access cost of several attributes. The results show that GBVP algorithm is more optimal for use in vertical table fragmentation process applied as query operation on distributed DBD database in a health field. The GBVP algorithm has less computational complexity, results a higher partition evaluator value and has lower query execution time than BEA.

Keywords: Query Distribution Process, Vertical Fragmentation, Optimization, BEA, GBVP, PE

Introduction

An increase of a large and complex database can decrease the performance and cost overruns of data access information system. Performance reduction and cost overruns occur due to function query accesses data retrieval from various attributes contained in a database table in which not all the attributes in the table are required. A distributed database can be implemented to improve the performance and reduce the cost of data access on a database. The process of designing a Distributed Databases is complex, so a data fragmentation (partitioning) scheme was used to facilitate a design process of a Distributed Databases (Al-Sayyed *et al*., 2014).

Fragmentation is a process of division or mapping of tables based on the columns and rows of data into the smallest unit of data. Data fragmentation is a process of division or mapping of a database where it is broken down by columns and rows stored in a computer site or a different unit in a data network, allowing for decisions to divided data (Abdalla and Amer, 2012). Data

fragmentation can be accomplished in several ways, including horizontal and vertical fragmentation. Horizontal fragmentation consists of a global fragment tuple subdivided or partitioned into several sub-sets. A blocking for this type is very useful in a distributed database, where each sub-sets can contain data that generally have a property. Vertical fragmentation subdivides the attributes of an available global fragment into several groups or subclass (Bhaskar and Sharma, 2012). The simplest form of vertical fragmentation is decomposition, where a row of unique-id can be included in each fragment to ensure and enable the reconstruction process through join operations. This fragmentation divides data into multiple tables which form interrelated attributes. This study was limited to test vertical fragmentation efficiency with Bond Energy Algorithm (BEA) and Graph-Based Vertical Partitioning (GBVP) algorithm. The main purpose of fragmentation is to minimize the number of access-related and share a relationship based on the efficiency of queries that are most frequently accessed (Al-Sayyed *et al*., 2014). To make the process of vertical fragmentation in the database to be tested, is based on the calculation of the algorithm Bond Energy (BEA) and the algorithm Graph-Based Vertical Partitioning (Rahimi and Riahi, 2015).

BEA is an algorithm used for a vertical fragmentation process. Information given about the use of attributes with initial transaction is converted into a square matrix referred to as the attribute affinity matrix which then is diagonalized by a cluster algorithm as the basis for calculating the bond energy algorithm. GBVP algorithm has less complexity in computing and produces meaning fragments with graphs. The affinity matrix is transformed into an affinity graph in order to partition the fragment based on defined rules and steps (Fung *et al*., 2002).

A vertical fragmentation design is initialized by building an Attribute Affinity matrix (AA). This matrix is used as the input generated from a multiplication of an attribute usage matrix with attribute query access matrix. The affinity matrix is then calculated using BEA to generate a clustered affinity matrix (Hoffer and Severance, 1975). The clustered affinity matrix determines attributes fragmentation. The calculation in GBVP algorithm has the same initial steps with BEA which is to perform an affinity matrix as the input. The matrix is then converted into a graph and the table is fragmented following the GBVP algorithm rules. Further, the rules for candidates identify a fragmentation of forming a cycle. The cycle can be extended to improve a decision fragmentation. The process runs until reaching at the end of a node. The results of the fragmentation of these algorithms are compared and evaluated using Partition Evaluators (PE) to determine which algorithm has performance that is more optimal.

In this research, an optimization of queries vertically generated on a fragmented table relationship using GBVP algorithm and BEA is analyzed and compared, where the implementation of BEA has been previously studied on a database in medical records management information system (Nurhendratno and Budiman, 2017).

Related Research

A clustering method based on vertical fragmentation to increase the system performance has become trend in a distributed database study, especially in determining the cost of query access. An implementation of vertical fragmentation by performing attributes clustering in a process of fragmentation in a distributed database was proposed by Rahimi and Haug (2010).

The method comprises two main algorithms. The first algorithm is used to place a set of data by simultaneously allocating the relevant elements and separating irrelevant elements. The second algorithm is used to cluster in which groups are created to determine a point to make pieces of a dataset. The main part of making vertical fragmentation in a distributed database is to find groups which contain relevant attributes in a relation table based on the affinity matrix value.

Affinity matrix contains a number of attributes with other attributes (the number of simultaneously accessing two attributes). The iteration in this algorithm is used and based on the grouping matrix $n \times n$ affinity matrix that will be used as the basic matrix in table fragmentation process that will be done (Rodríguez and Li, 2011). Initialization is conducted by downloading one column and placing it in the first column of the output matrix. Iteration step i n-i have a column on the left at the position $i + 1$ which allows the output matrix that will generate the greatest contribution to the calculation affinity calculations. Row ordering, at this step, the lines will be set the same as the column setting. Contributions from Ak column, which is placed between Ai and Aj. The next step is to calculate the number of accesses performed on each fragment is formed, then calculate the value maximize split quality (sq) of each fragment. The research has proven attributes in the cluster system will have a direct impact on the cost savings of storage and access costs. The study carried out by Hoffer and Severance (1975) can find a combination linearly with the cost of storage, retrieval and update the capacity restrictions for each file.

The fragments are separated into two-stage approach which are overlapping and non-overlapping fragments (Navate *et al*., 1984). The first stage is based on the empirical objective function and performs cost optimization by combining knowledge of the specific application environment in the second phase. Cornell and Yu (1990) proposed a model in a vertical partition problem as a programming problem of round number with the aim to minimize the number of disk accesses. This model uses certain physical factors related to the object files (attributes, length, selectivity and cardinality).

GBVP algorithm and BEA is analyzed and compared using the same affinity matrix, where the implementation of BEA has previously been studied on a database in medical records management information system by Nurhendratno and Budiman (2017), in which graph affinity was created by removing the existing value of 0 in the affinity matrix.

Analysis Comparison of Vertical Fragmentation

Bond Energy Algorithm

Bond Energy Algorithm (BEA) proposed by Hoffer, Severande and McCormick is an algorithm that can be used for vertical fragmentation process in a distributed database (Runeeanu, 2008). BEA Algorithm is divided into two steps, the first step is used to put a group of related data by allocating data elements simultaneously (elements who have no connection separated) and the second step can be used to form a group that is in charge of determining the point of a set of data (create cluster). The important thing in creating a vertical fragmentation in a distributed database is finding attributes which have been clustered in a relational table based on the affinity value in matrix of an attribute.

Affinity matrix is a matrix containing number of attributes which are mutually bound (number of access of two simultaneous attributes). BEA uses affinity matrix as an input to form clustered affinity matrix. Split function produces a clustered affinity matrix in the following steps: Initialization: Select and place one at random columns of the matrix into the matrix Clustered Affinity. The iteration step *i*: Place a column *n-i* at position *i* + 1 in the matrix Clustered Affinity. Rules contributions columns are illustrated in the following formula:

$$Count(Ai, Ak, Aj) = bond(Ai, Ak) + bond(Ak, Aj) - bond(Ai, Aj)$$

Graph-Based Vertical Partitioned Algorithm

Graph-Based Vertical Partitioning (GBVP) is different with the BEA. GBVP has a less computational complexity and produces fragments that have meaning by using a graph method. The input for GBVP algorithm is an affinity matrix considered a complete graph known as affinity graph where an edge value represents affinity between two attributes. A linearly connected spanning tree is then formed. This algorithm produces all fragments that have meaning in one iteration (Cornell and Yu, 1990). The steps of the algorithm in generating vertical fragments with affinity graph are:

1. Build an affinity graph from object attributes. Note that the matrix affinity is a sufficient data structure to represent the graph. No additional physical data storage is required
2. It can be started from any node
3. Select the edge that completes the conditions below:
    - It must be connected to the binary tree that is already established
    - It must have the greatest value among all existing edge selection
    - The iteration will end when all the nodes have already been used
4. If the next selected edge forms a primitive cycle:
    - If there is no node cycle, check all possibility cycles and if there is a possibility, mark the cycle as the affinity cycle. Return to step 3
    - If the there is an existing node cycle, remove the edge and continue to step 3
5. If the next selected edge does not form a cycle and there is a partition candidate, then:
    - If no former edge is found (selected edge is in between the last piece and a node cycle), check a possibility of a cycle extension of the new edge. If there is no possibility found, cut the edge and the cycle will be a partition. Return to step 3
    - If a former edge is found, change the cycle node and check the possibility of a cycle extension by the former edge. If there is no possibility found, cut the former edge and the cycle will be a partition. Return to step 3

Partition Evaluator

Partition Evaluator (PE) is a function to compare and evaluate different algorithms, using the same input on the process of designing a database. In the process of PE, the input used is an accessing attributes matrix followed by designing an *Evaluator* used to evaluate in finding the better partition or fragmentation (Lisbeth and Li, 2011). There are two common terms in PE which are *"irrelevant local attribute access cost"* and *"relevant attribute remote access cost"*.

Irrelevant local attribute access costs measures the cost of the transaction process which due to irrelevant attributes and assumes that all needed data fragments by a transaction are locally available. *Irrelevant local attribute access cost* is described by the following formula:

$$E_M^2 = \sum_{i=1}^{M} \sum_{t=1}^{M} q_t^2 * |R_{itk}| * \left\{1 - \frac{|R_{itk}|}{n_{itk}^r}\right\}$$

where, $|R_{itk}|$ is the number of attributes that are relevant in a fragment. While the *relevant remote attribute access cost* measures a remote processing costs caused by the relevant attributes of accessed data fragments. *Relevant remote attribute access cost* illustrated by the equation formula:

$$E_R^2 = \sum_{t=1}^{T} \min \left\{ \sum_{i=1}^{M} q_t^2 * \left| R_{itk} \right| * \frac{\left| R_{itk} \right|}{n_{itk}^r} \right\}$$

where, $\left| R_{itk} \right|$ is the number of attributes that are relevant in another fragments. While the function of *PE* is:

$$PE = E_M^2 + E_R^2$$

The definition and notation used in the PE functions are:

T = The number of transactions that are below the consideration
Q_t = Transaction frequency t, for $t = 1, 2, \ldots T$
M = The number of fragments of a partition
n_{itk}^r = The number of attributes that are accessed k fragments and fragments associated with the transaction of t
R_{itk} = The number of relevant attributes in fragment k accessed and related with fragment I by transaction t

Comparison Process of BEA and GBVP

In this research, the author proposed a procedure completion that will be done as the main purpose of this research. The process below is an example of vertical fragmentation process in a certain case:

A = (ICD, patient_name, address, gender, date_of_birth) are the attributes of the patient table and a query used is:
q1 = Select ICD, address from Patient
q2 = Select ICD, from Patient where date_of_birth = value
q3 = Select ICD, patient_name from Patient where gender = value
q4 = Select gender, address from Patient where date_of_birth = value

where, A1 = ICD, A2 = Patient_name, A3 = Address, A4 = Gender, A5 = Date_of_birth.

A matrix of the use of the attributes from attributes and query above is.

Next is calculating the frequency of each query on the entire web.

Next is building an affinity matrix resulted from multiplication of matrix of the use of the attributes with matrix of attributes and query access.

Approach with BEA

After forming affinity matrix, next is creating a cluster matrix from several attributes using split function.

The BEA uses affinity matrix as inputs to form a clustered affinity matrix. The contribution is calculated by randomly selecting two columns in the affinity matrix. A sorting result from the process produces the maximum value contribution which is [A3, A1, A5, A4, A2].

The next step is calculating the number of the accesses of each existing fragment by calculating the quality split value in each fragment:

1. Split at: [A1, A2, A3, A5] | [A4]
   Access fragment1 = 51
   Access fragment2 = 0
   Aksesfragment1 and fragment2 = 31
   Split quality = (51×0) – (〖31〗 ^2) = –961
2. When fragmentation is done at: [A1, A2, A5] | [A4, A3]
   Access fragment1 = 29
   Access fragment2 = 0
   Accessfragment1 and fragment2 = 53
   Split quality = (29×0)-(〖53〗 ^2) = -2809
3. When fragmentation is done at: [A1, A5] | [A3, A4, A2]
   Access fragment1 = 0
   Access fragment2 = 0
   Accessfragment1 and fragment2 = 82
   Split quality = (0×0)–(〖82〗 ^2) = –6724
4. When fragmentation is done at: [A1] | [A2, A3, A4, A5]
   Access fragment1 = 0
   Access fragment2 = 11
   Accessfragment1 and fragment2 = 71
   Split quality = (0×11)– 〖71〗 ^2 = –5041
5. When fragmentation is done at: [A1, A3, A4, A5] | [A2]
   Access fragment1 = 62
   Access fragment2 = 0
   Accessfragment1 and fragment2 = 20
   Split quality = (32×0)– 〖20〗 ^2 = –400

Based on the results, it can be concluded that the fragmentation with the maximum quality split is $sq = -400$ on the fragmentation done at [A1, A3, A4, A5] | [A2].

Approach With GBVP

GBVP algorithm uses the same affinity matrix with previous that generated by BEA as describe in Table 3. The graph affinity is made by removing value 0 in the affinity matrix. The graph can be seen in Fig. 1 where this process is conducting by starting from the node 1 (step 2) and follow by selecting the edge1-5 (step 3) and choosing the edge 5-3 edge as the next edge and forming a candidate to be partitioned (step 4). Note that the node 1 is a node cycle.

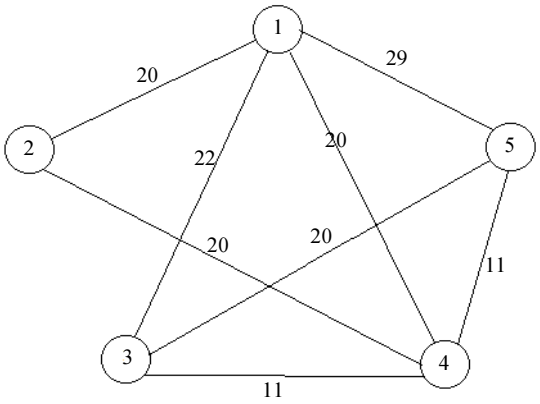

Fig. 1: Graph of affinity

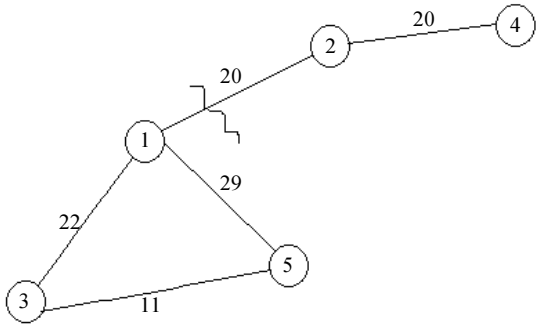

Fig. 2: Results fragmentation GBVP Algorithm, starting from node 1

The process is continued by selecting the edge 3-1, (check step 3), so the cycle of 1, 3, 5 is considered as a partition for the edge 1-2 and 2-4 are not eligible contained in step 4.2 and 5.1. Both of the edges cannot forming a cycle and the existence of a partition candidate and cycle node appear on the graph. The results of the process of the algorithm above are shown in Fig. 2 that the GBVP algorithm produces two affinity cycles separated by edge 1, 2 and two fragments which are (1, 3, 5) and (2, 4).

Two fragments which are (1, 3, 4, 5) and (2) are resulted from BEA and two fragments which are (1, 3, 5) and (2, 4) are resulted from GBVP algorithm. Partition Evaluator (PE) is use to compare and evaluate the resulted fragments from these algorithms. The inputs used in the process of PE are attribute accessing matrix (Table 1).

PE Calculations (Using BEA Algorithm)

1) Calculate *Irrelevant local attribute access cost*:

$$\begin{aligned} E_M^2 &= \left\{\left(1^2 * 2 * (1-2/4)\right) + \left(1^2 * 2 * (1-2/4)\right)\right. \\ &\quad \left. + \left(1^2 * 2 * (1-2/4)\right) + \left(1^2 * 3 * (1-3/4)\right)\right\} \\ &\quad + \left\{\left(1^2 * 1 * (1-1/1)\right)\right\} \\ &= 3,75 + 0 \\ &= 3,75 + 0 \end{aligned}$$

2) Calculate *relevant remote attribute access cost*:

Value	Value Minimum
Q1 on fragment 1	1^2*0*(0/1) = 0
Q1 on fragment 2	0^2*2*(2/4) = 0; (0)
Q2 on fragment 1	1^2*0*(0/1) = 0
tQ2 on fragment 2	0^2*2*(2/4) = 0; (0)
Q3 on fragment 1	1^2*1*(1/1) = 1
Q3 on fragment 2	1^2*2*(2/4) = 1; (1)
Q4 on fragment 1	1^2*0*(0/1) = 0
Q4 on fragment 2	0^2*3*(3/4) = 0; (0)

$$E_R^2 = 0 + 0 + 1 + 0 = 1$$

So, $PE = E_M^2 + E_R^2 = 3,75 + 1 = 4,75$

PE Calculations (Using GBVP Algorithm)

1) Calculate *Irrelevant local attribute access cost*

$$\begin{aligned} E_M^2 &= \left\{\left(1^2 * 2 * (1-2/3)\right) + \left(1^2 * 2 * (1-2/3)\right)\right. \\ &\quad \left. + \left(1^2 * 1 * (1-1/3)\right) + \left(1^2 * 2 * (1-2/3)\right)\right\} \\ &\quad + \left\{\left(1^2 * 2 * (1-2/2)\right) + \left(1^2 * 2 * (1-1/2)\right)\right\} \\ &\quad + (0,667 + 0,667 + 0,667) + (0 + 0,5) \\ &= 3,168 \end{aligned}$$

2) Calculate *relevant remote attribute access cost*:

Value	Value Minimum
Q1 On fragment 1	1^2*0*(0/2) = 0
Q1 On fragment 2	0^2*2*(2/3) = 0; (0)
Q2 On fragment 1	1^2*0*(0/2) = 0
Q2 On fragment 2	0^2*2*(2/3) = 0; (0)
Q3 On fragment 1	1^2*2*(2/2) = 2
Q3 On fragment 2	1^2*1*(1/3) = 1/3 = 0,33;(0,33)
Q4 On fragment 1	1^2*1*(1/2) = 1/2 = 0,5
Q4 On fragment 2	1^2* 2*(2/3) = 4/3 = 1,3; (0,5)

$$E_R^2 = 0 + 0 + 0,33 + 0,5 = 0,83$$

So, $PE = E_M^2 + E_R^2 = 3,168 + 0,83 = 3,998$

Complexity Analysis

A good algorithm is an efficient algorithm, the efficiency of the algorithm is measured by the amount of time and memory space needed to run the algorithm. An efficient algorithm is an algorithm that minimizes time and space requirements. The algorithm can be said to be good or efficient is it requires formal criteria used to assess the algorithm that is with its complexity.

There are two kinds of algorithm complexity, namely the complexity of time and space. The time complexity of the algorithm is to measure the number of computations performed by a computer when solving a problem using an algorithm. The size in question refers to the number of calculation steps and processing time of the processing. The time complexity of the algorithm contains the number expressions and the number of steps required as a function of the size of the problem. The complexity of space relates to the system memory required in program execution. Table 1 shows the algorithmic group based on the time complexity asymptotically.

The time and space requirements of an algorithm depend on the size of the input, typically the amount of data being processed. The size of the input is symbolized by *n*. After setting the input size, the next step in measuring the time complexity is to calculate the number of operations performed by the algorithm so that the notation of the time complexity in function n is $f(n)$.

Implementation and Comparison

In order to establish affinity matrix, there are several steps that must be completed, in the vertical fragmentation of activities. In the process of vertical fragmentation, we do a comparison results using 10 tables in the DBD health database (Dummy Data), using 70 queries to fragmentation table vertically. The measures that we use in executing the research outline is as follows: In the method of BEA, the first step is the formation of affinity matrix by classifying attributes based on the affinity (AA). The next perform matrix multiplication using Attributes (AU) with a matrix of Query Access (QA) thus that the contribution of each attribute value obtained to get a Split tilapia Quality (SQ) as a determinant of the result of fragmentation. To evaluate the value of the access cost, then after the obtained values of table fragmentation results from both methods, the next step is to compare the values of these fragments, by calculating the Partition Evaluator (PE).

From the utilization of the above two algorithms, generate some fragments of the BEA and GBVP algorithms. Furthermore, using Partition Evaluator (PE) results from both fragments the algorithm will compare and evaluate. The input used in this process is the PE matrix that can accesses the attributes.

PE Calculation (Using BEA and GBVP)

After using the BEA and GBVP methods, then calculate the cost to access the data from the calculation table with the proposed method. The results of fragmentation of the calculation table using BEA and GBVP methods in the table below:

The table above shows the result of the fragmentation of each table that uses GBVP and BEA algorithms. The results of the two algorithms above fragmentation display different results due to the fragmentation of the rules already established on the algorithm used. Results fragmentation by each of the methods will be tested by calculating the cost of data access using Partition Evaluator.

Access Cost

The results of the calculation of the cost of access to data by using Partition Evaluator (PE) show differences Partition Evaluator value of both BEA and GBVP algorithms which are shown in the table.

From the results of the experiments conducted, indicating that the GBVP algorithm produces a better fragmentation rather than BEA algorithms that can be seen from the partition Evaluators (PE) resulted from the total cost of access for relevant attributes and attribute minimal access cost is irrelevant. Where the greater value of Partition Evaluator (PE) produced which partition or fragmentation is better.

Comparison between BEA and GBVP

The results of the proposed algorithm show the comparison of query execution time on tables fragmented by using BE and GBVP algorithms. Execution time comparison results obtained from the implementation of the results table fragmentation generated by both algorithms when design of ProSIARS Distributed Databases. Comparison of the execution time is shown in the figure below.

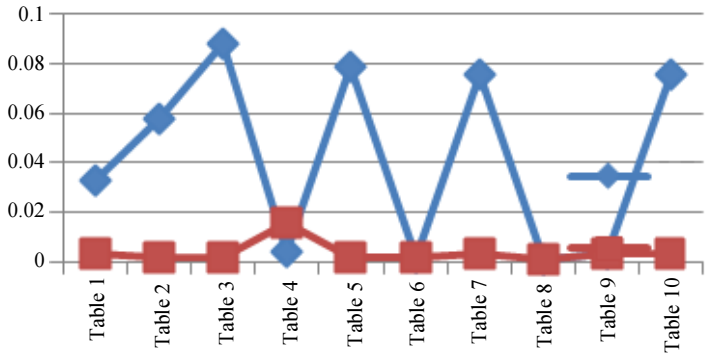

Fig. 3: Comparison graph algorithm execution time BEA and GBVP

Figure 3. shows the difference in access time of the query (execution) of the experiment using the BEA algorithm and the GBVP algorithm. Figure 3 shows that 6 of the 10 Tables (fragmented using the BEA algorithm) have a higher execution time than the GBVP algorithm (Tables 1, 2, 3, 5, 7 and 10). The graph also shows 3 tables having the same execution time (Tables 6, 8 and 9). While Table 4 is the only one fragmentation with BEA algorithm which has lower execution time compared to GBVP algorithm. For Table 10 and 11 are comparison of the space complexity of BEA and GBVP. There are the difference space of complexity are BEA and GBVP that is in the number of iterations (i) and space of complexcity $O(I*k*m*n)$.

Table 1: The use of any attribute matrix

	A1	A2	A3	A4	A5
q1	1	0	1	0	0
q2	1	0	0	0	1
q3	1	1	0	1	0
q4	0	0	1	1	1

Table 2: Matrix of query access on every site

	Site1	Site2	Site3	Amount
q1	10	7	5	22
q2	20	9	0	29
q3	3	12	5	20
q4	0	5	6	11

Table 3: Affinity matrix

	A1	A2	A3	A4	A5
A1	71	20	22	20	29
A2	20	20	0	20	0
A3	22	0	33	11	11
A4	20	20	11	31	11
A5	29	0	11	11	40

Table 4: Cluster affinity matrix

	A3	A1	A5	A4	A2
A3	33	22	11	11	0
A1	22	71	29	20	20
A5	11	29	40	11	0
A4	11	20	11	31	20
A2	0	20	0	20	20

Table 5: Query access matrix BEA

A1	A3	A4	A5	A2
1	1	0	0	0
1	0	0	1	0
1	0	1	0	1
0	1	1	1	0
Fragment 1			Fragment 2	

Table 6: Query access matrix GBVP

A1	A3	A5	A2	A4
1	1	0	0	0
1	0	1	0	0
1	0	0	1	1
0	1	1	0	1
Fragment 1			Fragment 2	

Table 7: Algorithmic group

Algorithm group	Name
$O(1)$	Constant
$O(\log n)$	Logarithmic
$O(n)$	Linear
$O(n \log n)$	$n \log n$
$O(n^2)$	Quadratic
$O(n^3)$	Cubic
$O(2^n)$	Exponential
$O(n!)$	Factorial

Table 8: Result of fragmentation

	Fragmentation results table	
Tables	BEA Method	GBVP Method
ms_pasien	[A2 A3 A1 A4] | [A5]	[A1 A2 A3] | [A4 A5]
tb_tindakan	[A3 A1 A5 A7 A2 A6] | [A4]	[A1 A2 A5 A6] | [A7 A3 A4]
tb_rekam_medis	[A9 A10 A6 A4 A3 A2 A1 A5 A7] | [A8]	[A1 A2 A3 A4] | [A6 A5 A8 A7 A9] | [A10]
ms_paramedis	[A6 A4 A2 A1 A3 A5] | [A7]	[A1 A2 A3] | [A6 A7] | [A4 A5]
ms_wilayah	[A3 A2 A4] | [A5 A1]	[A1 A2 A5] | [A3 A4]
ms_unit_surveilans	[A3 A2 A1] | [A4]	[A1 A2 A3] | [A4]
tb_resume	[A4 A5] | [A1 A2 A3]	[A1 A2 A3] | [A4 A5]
tb_rujukan	[A3 A2 A1] | [A4]	[A1 A2 A3] | [A4]
tb_kejadian	[A3 A5 A4 A2 A6 A7 A1] | [A8]	[A5 A2 A4] | [A6 A7 A8] | [A1] | [A3]
tb_kelas	[A4 A1 A3 A2] | [A5]	[A1 A2 A3] | [A4 A5]

Table 9: Partition evaluator value

	Value partition evaluator	
Tables	BEA	GBVP
ms_pasien	4,75	4,98
tb_tindakan	7,41	10,41
tb_rekam_medis	17,85	17,95
ms_paramedis	10,98	8,34
ms_wilayah	3,49	4,64
ms_unit_surveilans	4,34	4,34
tb_resume	5,32	5,32
tb_rujukan	3,72	3,72
tb_kejadian	15,28	10,3
tb_kelas	12,5	14,07

Table 10: Space complexity of BEA

Tables name	*Number of points* (*m*)	*Number of* attributes (*n*)	Number of iterations (*i*)	Space complexity $O(I*k*m*n)$
ms_pasien	60	5	10	3.000
tb_tindakan	30	7	21	4.410
tb_rekam_medis	63	10	44	27.720
ms_paramedis	51	7	21	7.497
ms_wilayah	19	5	10	9.500
ms_unit_surveilans	41	4	6	744.000
tb_resume	88	5	10	4.400
tb_rujukan	58	4	6	1.392
tb_kejadian	27	8	28	6.048
tb_kelas	9	5	10	450.000
Average Space Complexity with 10 tables relation 74 queries				5.661

Table 11: Space complexity of GBVP

Tables name	*Number of points* (*m*)	*Number of* attributes (*n*)	Number of iterations (*i*)	Space complexity $O(I*k*m*n)$
ms_pasien	60	5	6	1.800
tb_tindakan	30	7	8	1.680
tb_rekam_medis	63	10	11	6.930
ms_paramedis	51	7	8	2.856
ms_wilayah	19	5	10	950.000
ms_unit_surveilans	41	4	5	820.000
tb_resume	88	5	6	2.640
tb_rujukan	58	4	5	1.160
tb_kejadian	27	8	9	1.944
Tb_kelas	9	5	6	270.000
Average Space Complexity with 10 tables relation 74 queries				2.105

Time Complexity Reduction

The result of time complexity analysis of BEA and GBPV algorithm can be proved as follows:

1. Step 1 (initialization) of the Prim algorithm takes at most n operations. So the complexity of this step is $O(n)$
2. Step 2, is an iteration step, requires at most n-1 testing (since one node is selected in step 2), so the complexity of this step is $O(n)$
3. Step 3 run exactly n times. Each time you perform step 3, define the edge with the smallest weights of the unassigned node set (F) to the set of connected nodes (T), with at most n operations. So this step 3 has a complexity of $O(n)$

After the node in the most recent T set is marked, it is necessary to update the node list in the set F. For each node in the set F, there is a comparison of the weights of the smallest side of the node in the set F to the node of the set T, to determine the side with the smallest weights connected any node in the set F to any node in the set T. The partition renewal process is in $O(n)$ operation, none of this part 3 requires more than $O(n)$ operation, then the complexity of step 3 is $O(n)$, remember that $O(n) + O(n) = O(n)$. Since step 3 is implemented n times, then the operations performed is $(n)(O(n)) = O(n^2)$. So the complexity of all iteration steps (step 2 and step 3) is $O(n) + O(n^2) = O(n^2 + n) = O(n^2)$. From calculating the complexity of each of the above steps, the complexity of the GBPV algorithm is the complexity of step 1 + the complexity of the steps of all iterations (step 2 and step 3). the time complexity in function n is s:

$$\begin{aligned} f(n) &= O(n) + O(n^2) \\ &= O(n^2 + n) \\ &= O(n^2) \end{aligned}$$

So it can be concluded that GBVP time complexity is quadratic.

Then the time required is nothing more than $d + \sum_{I=1}^{n-1}\left[c + b + a(n+1)\right]$, which can be simplified into $((a/2)*n^2+(b+c-a/2)*n+(d-b))$. The function is dominated by $(a/2)*n^2$. Then according to the definition of big-O can be written $f(n) = n^2$, $c = a/2$ and $n^0 = 0$. It can be said that the algorithm has the time complexity in quadratic, $O(n^2)$.

The exponential function $T(n) = 2^{n+k}$, where k is a constant, is $O(2^n)$ because 2^{n+k} is $2^k 2^n$ for all n. Generally, $T(n) = m^{n+k}$ is $O(l^n)$; $l \geq m > 1$, because $m^{n+k} \leq l^{n+k} = l^k l^n$ for any constant k.

Below is a split bond energy algorithm that can be seen in Fig. 4 where the input is an Cluster Affinity Matrix (CAM) and the output is F Set of two Fragments.

Split of Bond Energy Algorithm	
Input : Cluster Affinity Matrix (CAM) Output: *F* Set of two Fragments	
	Begin {Initialization of variables} *X*[1,1..*N*]; *Y*[1,1..*Y*];
1.	{Ditermine Split Point} For *I* = 1 to *n* do
2.	If (*i* == 1) then
3.	*Y*[1,*i*] = CAM(1,i);
4.	Else
5.	*Y*[1,*i*] = CAM(1,*i*)-CAM(1,*i*–1);
6.	End-If
7.	*X*[1,*i*] = *i*;
8.	End-For
9.	Plot(*X*,*Y*);
10.	Smallest = *Y*[1,1];
11.	Split-Point = 1;
12.	For *i* = 2 to n
13.	If (Smallest< *Y*[1, *i*] then
14.	Split-point is recorded as *X*[1, *i*]
15.	Smallest = *Y*[1, *i*]
16.	End-If
17.	End-For
18.	End-Begin
$T(n) = d + \sum_{I=1}^{n-1}\left[c + b + a(n-1)\right]$	

Fig. 4: Split of Bond Energy Algorithm (BEA).

Space Complexity Reduction

The space requirements for BEA are modest because only the data points and centroids are stored. Specifically, the storage required is:

$$O\left((m+k)n\right)$$

where, *m* is the number of points and n is the number of attributes. The time equirements for BEA are also modest-basically linear in the number of data points. In particular, the time required is:

$$O(I*k*m*n)$$

where, *I* is the number of iterations required for convergence, as mentioned *I* is often small and can usually be safely bound as most changes typically occur in the first few iterations which can be seen in Fig. 5 split of GBVP algorithm. Therefore BEA is linear in *m*, the number of points and is efficient as well as simple provided that *K*, the number of cluster is significantly less than m (Rahimi and Riahi, 2015).

Split of GBVP	
Input : $F^{j,}$ Q1,Q2…Q_M Output : Rec-ID^j Cluster-NO^j [1:N/K]	
Begin {Initialzation steps}	
1.	For each F^j do in parallel
2.	For p = 1 to N/K do
3.	Read F^j (p);
4.	Cluster -NO^j [p] = 0;
5.	REC-ID^j [p] = (j-i) * N/K + p;
6.	For i = 1 to M do
7.	If F^j [p] satisfies Q_i then
8.	Cluster – NO^j [p] = Cluster – NO^j [p] +2^{M-I}
9.	End-If
10.	End-For
11.	End-For
$T(n) = 2^{nxk}$	

Fig. 5: Split of GBVP algorithm

Conclusion

The purpose of conducting this study is to know the impact on the response time while moving from centralized to distributed databases with BEA and GBVP Algorithms.

Experiments fragmentation vertically done using BEA and GBVP algorithms at 10 tables by using a total of 74 queries and input an affinity matrix resulted in the fragmentation of the different tables. Based on the results of trials that have been done show that GBVP algorithm is an algorithm that is more optimal for use in the process of fragmentation of the table vertically. The statement was supported by the results of the analysis of algorithms GBVP who have less computational complexity and generate value Partition Evaluator higher and has a query execution time is lower compared with the results of fragmentation using BEA algorithm. Distributed databases have many aspects and every organization has certain preferences. For the public health DBD (Dummy Data), the response time is prioritized. Our experiment showed that the average response time is decreased if we switch from centralized database to distributed database. In distribution, we put the data to the site where it is used most frequently. This locality of data reduces the response time. In the distributed database, data is fragmented. These fragments are short compared to the full database (centralized database contains maximum columns). However, when we need data from multiple sites for a query (report queries), the response time is increased. Accessing data from multiple remote sites and then joining those takes a long time. But in the centralized database, since data is at one place so, it is easy and fast to search it. The purpose of conducting this study is to know the impact on the response time while moving from centralized to distributed databases using vertical fragmentation. Experiment results showed that the response time is decreased in distributed databases. Due to fragmentation dataset for the single site contains fewer records than the centralized database, therefore the response time is less. In algorithm performance (time and space complexity) GBVP algorithm has better space complexity than BEA (50% better). For Time complexity the BEA algorithm has a group of quadratic functions of $O(n^2)$. The GBVP algorithm has an exponential algorithm group $O(n^2)$.

Acknowledgement

This research was provided by the Research and Technology Ministry of Higher Education, sponsored under a grant budget of private colleges compete coordinator VI Central Java, Indonesia.

Funding Information

All funding for conducting this research comes from the research grant scheme of the ministry of research and higher education of the Republic of Indonesia.

Author's Contributions

Slamet Sudaryanto Nurhendratno: Designed and analysed data, performed experiments and co-wrote the paper.

Sudaryanto: Performed experiments BEA and GBVP, simulation and evaluation.

Maryani Setyowati: Designed experiments and work supervision.

Ethics

This article is the original contribution of the authors and is not published elsewhere. There is no ethical issue involved in this article.

References

Abdalla, H. and A. Amer, 2012. Dynamic horizontal fragmentation, replication and allocation model in DDBSs. Proceedings of the International Conference on Information Technology and e-Services, Mar. 24-26, IEEE Xplore Press, Sousse, Tunisia, pp: 1-7. DOI: 10.1109/ICITeS.2012.6216603

Al-Sayyed, R., F. Al Zaghoul, D. Suleiman, M. Itriq and I. Hababeh, 2014. A new approach for database fragmentation and allocation to improve the distributed database management system performance. J. Softw. Eng. Applic., 7: 891-905. DOI: 10.4236 / jsea.2014.711080

Bhaskar, R. and R. Sharma, 2012. An analysis of vertical splitting algorithm. Int. J. Comput. Applic., 52: 30-36. DOI: 10.5120/8304-1767

Cornell, D.W. and P.S. Yu, 1990. A vertical partitioning algorithm for relational database. Proceedings of the International Conference on Data Engineering, (CDE' 90), Los Angeles, California, pp: 30-35.

Fung, C., K., Karlapalem and Q. Li, 2002. An evaluation of vertical class partitioning for query processing in object-oriented databases. IEEE Trans. Knowl. Data Eng., 14: 1095-1118. DOI: 10.1109/TKDE.2002.1033777

Hoffer, J.A and D.G. Severance, 1975. The use of cluster analysis in physical database design. Proceedings of the 1st International Conference on Very Large Database, (VLD' 75), pp: 69-86.

Lisbeth, R. and X. Li, 2011. A vertical partitioning algorithm for distributed multimedia databases. Proceedings of the International Conference on Database and Expert Systems Applications, (ESA' 11), Springer, Berlin, Heidelberg, pp: 544-558. DOI : 10.1007 / 978364223 091248

Navate, S.B., S. Ceri, G. Wiederhold and J. Dour, 1984. Vertical partitioning algorithms for database design. ACM Trans. Database Syst., 9: 680-710. DOI: 10.1145/1994.2209

Nurhendratno, S.S. and F. Budiman, 2017. Design model integration and syncronization between surveillance units to support data warehouse epidemiology. J. Theoretical Applied Informa. Technol., 95: 498-505.

Rahimi, H. and D. Riahi, 2015. Hierarchical simultaneous vertical fragmentation and allocation using modified Bond Energy Algorithm in distributed databases. Applied Comput. Informat. Saudi Comput. Society, King Saud Uni. DOI: 10.1016/j.aci.2015.03.001

Rahimi, S.K. and F.S. Haug, 2010. Query Optimization. In: Distributed Database Management Systems: A Practical Approach, Rahimi, S.K. and F.S. Haug (Eds.), John Wiley and Sons, Inc., NJ. Hoboken, USA., ISBN-10: 0470602368.

Rodríguez, L. and X. Li, 2011. A support-based vertical partitioning method for database design. Proceedigns of the 8th International Conference on Electrical Engineering Computing Science and Automatic Control, Oct. 26-28, IEEE Xplore Press, Mexico, pp: 1-6. DOI: 10.1109/ICEEE.2011.6106682

Runeeanu, R., 2008. Fragmentation in Distributed Databases. In: Innovations and Advanced Techniques in System, Computing Science and Software Engineering, Elleithy, K. (Ed.), Springer Science, Business Media B.V, ISBN-10: 9781402052620, pp: 57-62.