Query Optimization on Distributed Health Database DBD by Minimizing Attribute Involvement

Slamet Sudaryanto N[1], Sudaryanto[2], Maryani S[3]
[1,2]Computer Science Faculty, Dian Nuswantoro University
Central Java, Indonesia
[1]slametalica301@dsn.dinus.ac.id
[2]msdr8047@gmail.com

[3]Health Faculty, Dian Nuswantoro University
Central Java, Indonesia
watiek_ms@yahoo.com

JDIM
Journal of Digital Information Management

ABSTRACT: *Query optimization is an important task in the client / server environment of a distributed database, where large data locations are widely distributed, such as distribution of health epidemiology data based DBD on geographic information systems (GIS). In order to generate query optimization on a distributed database it is necessary to have a proper method on a particular query process function. The query process requires important attention especially in distributed databases, because the result of a cost-based query process (access fees and communication costs) is affected by the involvement of the number of attributes and sites visited. If a query can be decomposed into subqueries that require operations on a separate database (geographically) and can determine the exact site access sequence of a query process query circuit then the operating costs for the query process will be minimal. When a query process in a distributed database occurs, queries operations will search for data from various attributes in a scattered database table, whereas query processes often do not require all the attributes of the table. Therefore, in order to optimize the query required minimum query operation cost (communication cost and access cost). One way to minimize the cost of queries is to separate attributes that are not required by a query, thereby reducing the amount of time communication and access. Can not make mistakes in the separation of attributes, attributes should not be split indiscriminately, because if not appropriate it will result in the amount of access costs are getting larger and ultimately reduce the performance of the query process itself. To perform such attribute separation can be done by Vertical Fragmentation method. In this experiment will be conducted by comparing the results of separation attributes. Separation of attributes will be done by using Vertical Fragmentation method to source health database tables (database testing), while the algorithm used for attribute separation is Bond Energy Algorithm (BEA) and Graphic Based Vertical Partitioning (GBVP). The initial result of vertical fragmentation in both algorithms is the determination of what attributes will be separated from a number of specific query processes. The result of separation of attributes from each algorithm will be compared and evaluated using Partitioned Evaluator (PE). The purpose of evaluation of the result of separation of attribute is to know the amount of access cost of some attributes from each algorithm. Thus it will be known the most optimal algorithm in the operation of the query process. Algorithms that have better performance are algorithms that have the lowest execution time. The purpose of this research is to perform query optimization by applying the correct algorithm, by way of framing the method of fragmentation between BEA algorithm with GBVT to then concluded algorithm which has query optimization performance from the most optimal query process so it is suitable to be applied as query operation on distributed database in field health.*

Subject Categories and Descriptors
[H.2.4 Systems]; Query processing: **[C.2.4 Distributed Systems];** Distributed databases

General Terms: Distributed Databases, Query Optimization

Keywords: GIS, Proses Query Distribute, Vertical Fragmentation, Ooptimasi Query, BEA, GBVP, PE.

Received: 11 December 2017, Revised 24 January 2018, Accepted 31 January 2018.

DOI: 10.6025/jdim/2018/16/3/105-113

Nomenclature

AA: Attribute Affinity
AU: Attribute Usage
BEA: Bond Energy Algorithm
CA: Cluster Affinity
GBVP: Graph-Based Vertical Partitioning
SQ: Split Quality
PE: Partition Evaluator

1. Introduction

An increasingly large and complex databases in a system can cause a decrease in performance and cost overruns on a system of data access information. Performance reduction and cost overruns occur due to the function query will do the access data retrieval from various attribute that contained in the database table, the table is accessed while, not all the attributes required. One way to improve performance and reduce the cost of data access at database can be overcome by designing a Distributed Database. But in the process of designing a Distributed Databases are complex, so the scheme is used fragmentation (partitioning) of data to facilitate the design process of Distributed Databases [1].

Fragmentation is a process of division or the 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 database where the database is broken down by columns and rows are then stored in a computer site or a different unit in a data network, allowing for decisions to data which has been divided [2]. Fragmentation of data can be accomplished in several ways, including horizontal fragmentation, vertical fragmentation. Horizontal fragmentation consists of a tuple of global fragment is then subdivided or partitioned into several sub-sets. Blocking this type is very useful in a distributed database, where each sub-sets can contain data that has the property in general. Vertical fragmentation will subdivide the attributes of the available global fragment into several groups or subclass [3]. The most simple 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 a join operation. In other words, this kind of fragmentation will divide the data into multiple tables that are interrelated attributes. In this study will only test the vertical fragmentation efficiency with the approach Bond Energy Algorithm (BEA) and Graph-Based Vertical Partitioning (GBVP).

The main purpose of fragmentation is done to minimize the number of access-related and share a relationship based on the efficiency of queries that are most frequently accessed [1]. 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 (GBVP) [4].

BEA is one of the algorithms used in the process of Vertical fragmentation, information given about the use of attributes with traksasi initially converted into a square matrix, referred to as the attribute affinity matrix. The next step will be in this matrix is diagonalized by the algorithm cluster as the basis for calculating the bond energy algorithm[5]. While GBVP an algorithm that has a complexity of computing less and produce fragments that everything has a meaning by using graph (graph) that transform matrix affinity into a graph affinity for partitioning the fragment in accordance with the rules and the steps that have been defined in the algorithm GBVP ([4],[6])

The first step in designing the study vertical fragmentation, is to build an attribute affinity matrix (AA). This affinity matrix as input generated from the use of matrix multiplication and matrix attribute query access. Affinity matrix is then calculated using BEA algorithm that generates a clustered affinity matrix [7]. Clustered affinity matrix will determine on which attributes fragmentation will do. Calculations in GBVP algorithm also has the same initial steps with BEA that perform input an affinity matrix, the matrix next converted into a graph, then the table will be fragmented following the rules of the algorithm GBVP. Where rules for candidates identified fragmentation of forming cycles. This cycle can be extended to improve decision fragmentation. The process runs until all nodes run out. The result of the fragmentation of the two algorithms are then compared and evaluated using Partition Evaluator (PE) to determine the fragmentation which is more optimal algorithm.

Based on this background, the authors analyze and compare how the optimization of queries generated on the table relationships fragmented vertically by using algorithms and algorithms Energy Bond Graph-Based Vertical Partitioning that have done on Database Management Information System Hospital Medical Record.

2. Related Work

The concept of vertical fragmentation to increase performance by dividing attributes into groups (clusters)

of each attribute of global fragment has often become liletratur, especially in relation to the cost of access (cost-based). In designing the process of fragmentation (in a distributed database) has been studied previously by ([6] - [10]), they have been discussing the implementation of vertical fragmentation by performing clustering attributes. The method generally comprises two main algorithms. The first algorithm is used to place a set (set) of data by allocating the most pertinent elements together (and separated by a set of elements that are not related). The second is an algorithm used to create the group, which determines the point to make the pieces of the data set (create clusters). The main part is done in making vertical fragmentation in the distribution database is to find a grouping of attributes in a relation table based on the values to values affinity attribute affinity matrix. Affinity matrix is a matrix containing the attachment between the number one attribute with other attributes (the number of simultaneously accessing two attributes). Repeat partitioning method (iteration) in this algorithm was used [9] and [10] 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. Initializaion download one column and place it in the first column of the matrix output. Iteration step i, $n - i$ have a column on the left at the position $i + 1$ which allows the output matrix that will cause 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 maximaze split quality (sq) of each fragment. They have 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 [7] can find a combination linearly with the cost of storage, retrieval and update the capacity restrictions for each file.

The method presented by Navate et al [9], with a two-stage approach in separating the fragments into fragments overlapping and non-overlapping. The first stage is based on the empirical objective function and then perform cost optimization by combining knowledge of the specific application environment in the second phase. Cornell and Yu [11] proposed a model in a vertical partition problems as a programming problem bilanga round with the aim to minimize the number of disk accesses. This model uses certain physical factors related to the object files (attributes, length and selectifity, cardinality).

In this paper we will use an algorithm to cluster database that Bond Energy Algorithm (BEA). And to compare the results, use the same affinity matrix generated Earlier in the BEA algorithm to be done using algorithms Graph Based Partitioned Vertical Partitioning (GBVP). Graf affinity are made by removing the existing value of 0 in the affinity matrix.

3. Analisys Comparison of Vertical Fragmentation

3.1 Bond Energy Algorithm

Bond Energy algorithms or Bond Energy Algorithm (BEA) is an algorithm that can be used for vertical fragmentation process in a distributed database [5]. Proposed by the BEA, Hoffer, Severande and McCormick. Algorithms BEA is divided into two steps, the first algorithm is used to put a group of related data by allocating data elements simultaneously (elements who have no connection separated), while the second algorithm can be used to form a group that is in charge of determining the point of a set of data (cluster made).

To create vertical fragmentation in a distributed database, the main thing to note is finding attributes are already organized in a relational table based on the affinity that existed at an attribute affinity matrix. Affinity is a matrix containing attachment number one attribute with another attribute that (number of two attributes simultaneously).

BEA uses Affinity Matrices as an input to form Clustered Affinity Matrix. Split function to produce a matrix Clustered Affinity with the following steps: Initialization: select and place one at random columns of the matrix into the matrix Clustered Affinity. Iterasi step i: place a column n-i at position i + 1 in the matrix Clustered Affinity. Rules contributions columns illustrated with the following formula:

$$Cont(Ai, Ak, Aj) = bond(Ai, Ak) + bond(Ak, Aj) - bond(Ai, Aj).$$

3.2 Algorithm Graph-Based Verical Partitioned

Unlike the Bond Energy algorithms, algorithms Graph-Based Vertical Partitioning (GBVP) is an algorithm that has a less computational complexity and produce fragments that everything has a meaning by using graph (graph).

Input from GBVP algorithm is Matrix Affinity. In GBVP algorithm, Matrix Affinity considered a complete graph is called regular or affinity graph where the edge value represents the affinity between two attributes. Then, forming a spanning tree connected linearly, this algorithm generates all the fragments that have meaning in one iteration [11,5]. The algorithm for generating vertical fragment with affinity graph, will be explained below by using the 5 steps [12,2]:

1. Build affinity graph of object attributes. Note that the own matrix affinity is sufficient data structure to represent this graph. No additional physical data storage is required.

2. Can be started from node anywhere.

3. Choose the *edge* that complete the condition below :

- Must be connected to the binary tree is established.
- Must have the greatest value among all existing edge selection.
- This iteration will end when all the nodes are already in use.

4. When the edge is selected subsequently form a primitive cycle:

• If no node cycle, check all the possibilities cycles and if there is a possibility, mark the cycle as the cycle of affinity. Return to step 3.

• If the existing node cycle, waste edge and proceed to step 3

5. When the edge is selected next does not form a cycle and there is a candidate partition, then:

• If no former edge (edge selected which are among the last piece and node cycle), check the possible extension of new edge cycle. If it is not found possible, cut edge and cycle will be a partition. Return to step 3.

• If found former edge, change node cycle and check the possibility of an extension of the cycle by the former edge. If it is not found possible, cut former edge and cycle will be a partition. Return to step 3.

3.3 Partitioned 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 input is used matrices Accessing Attributes, followed by designing an Evaluator used to evaluate the partition or fragmentation which is better [13]. PE has two terms that are often used first is "irrelevant local attribute access cost" or local access charges attributes irrelevant and "relevant attribute remote access cost" or long-distance access charges relevant attributes.

Irrelevant local access attribute costs measure the cost of the transaction process resulting from the attributes are not relevant, assume that all the fragments of data needed by local. Irrelevant available transaction attribute access cost is described by the formula:

$$E_M^2 = \sum_{i=1}^{M} \sum_{t=1}^{T} q_t{}^2 * |R_{itk}| * \left(1 - \frac{|R_{itk}|}{n^r_{itk}}\right)$$

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

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

Where $|R_{itk}|$ in this formula is the number of attributes that are relevant in some other fragments. While the function of PE are:

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

Below are definitions and notation used in the PE functions:

T = The number of transactions that are under consideration.

Qt = transaction frequency t, for $t = 1, 2, \ldots .., T$.

M = the number of fragments of a partition.

n^r_{itk} = The number of attributes that are accessed k fragments and fragments associated with the transaction t.

R_{itk} = The number of relevant attributes k accessed in fragments and fragments associated with the transaction t.

3.4 Comparison Process BEA and GBVP

In carrying out the study, the authors have proposed related to the settlement procedures would be conducted so that the study runs in accordance with the original purpose of the study. The process below is an example of vertical fragmentation process on a specific case:

A = (ICD, patient_name, address, gender, date_of_birth) are the attributes of the patient table, and a query that is 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 in A1= ICD, A2= patient_name, A3=address, A4=gender, A5=date_of_birth.

Then generated a matrix of Use attributes and query attributes of the above, namely:

	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 1. The use of any attribute matrix

The next calculate the frequency of each query on the entire web.

The next step to build affinity matrix resulting from the use of matrix multiplication matrix Attributes and Query Access.

	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 2. Matrix access query every site

	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 3. Affinity matrix

3.5 Approach With BEA

After forming Affinity Matrices, then creates a matrix cluster of several attributes using split function. BEA uses Affinity Matrices as an input to form Clustered Affinity Matrix. The next contributions calculated by selecting two columns at random affinity matrix column. Sequencing results obtained from the calculation process that produces the greatest value contribution is: [A3, A1, A5, A4, A2].

	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 4. Cluster affinity matrix

After treatment is completed, count the number of accesses each fragment that is, calculate the value split quality in each fragment:

1. Split at: [A1, A2, A3, A5] | [A4]

Access fragment1 = 51

Access fragmen2 = 0

Aksesfragmen1 and fragmen2 = 31

Split quality = (51 × 0) - ([31] ^ 2) = -961

2. When fragmentation is done at: [A1, A2, A5] | [A4, A3]

Access fragmen1 = 29

Access fragmen2 = 0

Aksesfragmen1 and fragmen2 = 53

Split quality = (29 × 0) - ([53] ^ 2) = -2809

3. When fragmentation is done at: [A1, A5] | [A3, A4, A2]

Access fragment 1 = 0

Access fragmen2 = 0

Aksesfragmen1 and fragmen2 = 82

Split quality = (0 × 0) - ([82] ^ 2) = -6724

4. When fragmentation is done at: [A1] | [A2, A3, A4, A5]

Access fragmen1 = 0

Access fragmen2 = 11

Aksesfragmen1 and fragmen2 = 71

Split quality = (0 × 11) - [71] ^ 2 = -5041

5. When fragmentation is done at: [A1, A3, A4, A5] | [A2]

Access fragmen1 = 62

Access fragmen2 = 0

Aksesfragmen1 and fragmen2 = 20

Split quality = (32 × 0) - [20] ^ 2 = -400

From the above results, it can be concluded that the fragmentation of the split is the most optimal quality *sq* = - 400 on fragmentation is done at [A1, A3, A4, A5] | [A2]

3.6 Approach with GBVP

GBVP algorithm uses the same affinity matrix generated with previous BEA algorithm in Table 3 Graf affinity are made by removing the existing value of 0 in the affinity matrix.

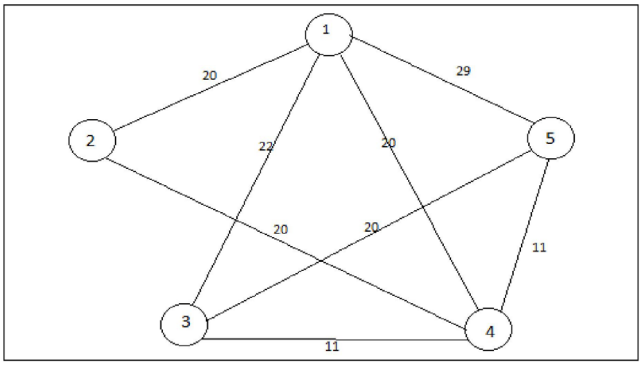

Figure 1. Graf Affinity

Let us start from the node 1 (step 2), then edge1-5 selected (step 3). Then the 5-3 edge was chosen to be the next edge and forming a candidate to be partitioned (step 4). Note that node-1 is node cycle. Then proceed with the process of selecting edge3-1, check 1,3,5 step 3. So the cycle is considered as a partition for edge 1-2 and 2-4 are not eligible contained in step 4,2 and 5,1 for the second step of the edge can not be formed a cycle and the cycle of a candidate partitions and nodes that appear on a graph. The results of the above algorithm is shown in Figure 2 As shown in Figure 2, the algorithm produces two cycles of affinity GBVP separated by edge1,2. This algorithm produces two fragments, namely (1,3,5) and (2,4).

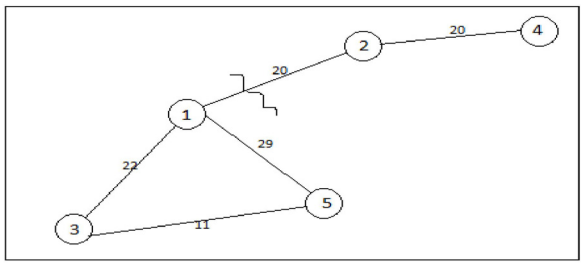

Figure 2. Results fragmentation GBVP algorithm, starting from node to-1

From the above two algorithms, produce the two fragments of BEA algorithm that fragments (1, 3, 4, 5) (2). While GBVP algorithm produces two fragments, namely (1, 3, 5) and (2, 4). The next, by using Partition Evaluator (PE) results of both algorithms fragments will be compared and evaluated. Inputs used in the process of PE is Accessing Attribute Matrix (Table 1).

3.7 PE Calculations (using BEA algorithm)

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	

Figure 3. Query Access Matrix BEA

1) Calculate *Irrelevant local attribute access cost*

$$E^2_M = \{(1^2 * 2 * (1-2/4)) + (1^2 * 2 * (1-2/4) + (1^2 * 2 * (1-2/4)) + (1^2 * 3 * (1-3/4))\} + \{(1^2 * 1 * (1-1/1))\}$$

$$= 3,75 + 0$$

$$= 3,75$$

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	$1^2 * 0 * (0/1)=0$	
Q2 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

$$E^2_R = 0 + 0 + 1 + 0 = 1$$

So, $\mathbf{PE} = E^2_M + E^2_R = 3,75 + 1 = 4,75$

3.8 PE Calculations (using GBVP algorithm)

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	

Figure 4. Query Access Matrix GBPV

1) Calculate *Irrelevant local attribute access cost*

$$E^2_M = \{(1^2 * 2 * (1-2/3)) + (1^2 * 2 * (1-2/3)) + (1^2 * 1 * (1-1/3)) + (1^2 * 2 * (1-2/3))\} + \{(1^2 * 2 * (1-2/2)) + (1^2 * 2 * (1-1/2))\}$$

$$= (0,667 + 0,667 + 0,667 + 0,667) + (0 + 0,5)$$

$$= 3,168$$

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	$1^2 * 1 * (1/2)=1/2=0,5$	0,5
Q4 On fragment 2	$1^2 * 2 * (2/3)=4/3=1,33$	

$$E^2_R = 0 + 0 + 0,33 + 0,5 = 0,83$$

So, $\boldsymbol{PE} = E^2_M + E^2_R = 3,168 + 0,83 = 3,998$

4. Implementation and Comparison

In order to establish affinity matrix, there are several steps that must be done, in the vertical fragmentation of activities. In the process of vertical fragmentation, we do a comparison results using 10 tables in the database Pro SIARS, 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) so that the contribution of each attribute value obtained to get a split tilapia quality (SQ) as a determinant of the result of fragmentation. In order 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 fragmetasi, by calculating the partition evaluator (PE).

From the above two algorithms, the algorithm produced a

few fragments of the BEA and GBVP. Furthermore, by using Partition Evaluator (PE) results of both algorithms fragments will be compared and evaluated. Inputs used in the process is the PE matrix Accessing Attributes.

4.1 PE Calculations (using BEA algorithm)

After getting the results of the calculations have been done, the next step is to compare the cost of access to the data from the calculation of the fragmentation of the table with the proposed method and the method BEA GBVP method. The results of the calculation table fragmentation using BEA and GBVP methods shown in the table below:

No	Table	Fragmentation results table	
		BEA Method	GBVP Method
1	ms_pegawai	[A2 A3 A1 A4] | [A5]	[A1 A2 A3] | [A4 A5]
2	tb_absensi	[A3 A1 A5 A7 A2 A6] | [A4]	[A1 A2 A5 A6] | [A7 A3 A4]
3	tb_nilai	[A9 A10 A6 A4 A3 A2 A1 A5 A7] | [A8]	[A1 A2 A3 A4] | [A6 A5 A8 A7 A9] | [A10]
4	ms_siswa	[A6 A4 A2 A1 A3 A5] | [A7]	[A1 A2 A3] | [A6 A7] | [A4 A5]
5	ms_prodi	[A3 A2 A4] | [A5 A1]	[A1 A2 A5] | [A3 A4]
6	ms_mapel	[A3 A2 A1] | [A4]	[A1 A2 A3] | [A4]
7	tb_siswakelas	[A4 A5] | [A1 A2 A3]	[A1 A2 A3] | [A4 A5]
8	tb_walikelas	[A3 A2 A1] | [A4]	[A1 A2 A3] | [A4]
9	tb_jadwal	[A3 A5 A4 A2 A6 A7 A1] | [A8]	[A5 A2 A4] | [A6 A7 A8] | [A1] | [A3]
10	tb_kelas	[A4 A1 A3 A2] | [A5]	[A1 A2 A3] | [A4 A5]

Table 5. Results of fragmentation

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

4.2 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 below:

From the results of experiments conducted, show that the algorithm produces fragmentation table GBVP better than the BEA algorithm seen from the Partition Evaluator (PE) resulting from the sum of the cost of access to the relevant attributes and attribute the minimum access charges that are not relevant. Where the greater value of Partition Evaluator (PE) produced will show the partition or fragmentation which one is better.

No	Tabel	Value Partition Evaluator	
		BEA	GBVP
1	ms_pegawai	4,75	4,98
2	tb_absensi	7,41	10,41
3	tb_nilai	17,85	17,95
4	ms_siswa	10,98	8,34
5	ms_prodi	3,49	4,64
6	ms_mapel	4,34	4,34
7	tb_siswakelas	5,32	5,32
8	tb_walikelas	3,72	3,72
9	tb_jadwal	15,28	10,3
10	tb_kelas	12,5	14,07

Table 6. Partition evaluator value

From the results of experiments conducted, show that the algorithm produces fragmentation table GBVP better than the BEA algorithm seen from the Partition Evaluator (PE) resulting from the sum of the cost of access to the relevant attributes and attribute the minimum access charges that are not relevant. Where the greater value of Partition Evaluator (PE) produced will show the partition or fragmentation which one is better.

4.3 PE Calculations (using BEA algorithm)

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

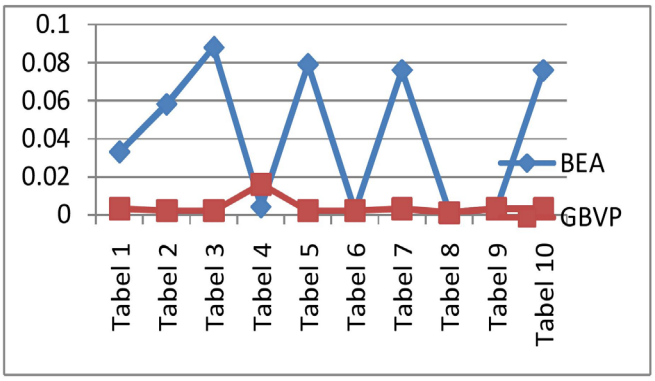

Figure 5. Comparison graph algorithm execution time BEA and GBPP

GBVP algorithm in table 1, 2, 3, 5, 7 and 10. The graph also shows in table 6, 8 and 9 both algorithms has the same execution time.

5. 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 algorithm and PBVP Algorithm.

Experiments fragmentation vertically done using BEA algorithms and algorithms GBVP 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 academic sector (ProSIARS), 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 data-base (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 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 data set for single site contains less records than centralized data-base, so response time is less.

Acknowledgements

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

References

[1] Gope, D. C. (2012). Dynamic Data Allocation Methods in Distributed Database System, *American Academic & Scholarly Research Journal*, 4 (6) (November).

[2] Ashraf, Imran., Khokhar, A. S. (2010). Principles for Distributed Databases in Telecom Environment., Sweden.

[3] Gupta, S., Panda, S. (2012). Vertical Fragmentation, Allocation and Re-Fragmentation in distributed Object Relational Database Systems-(Update Queries Included), *International Journal of Engineering Research and Development*, 4 (7) 45-52, (November).

[4] Rahimi, H., Riahi, D. (2015). Hierarchical simultaneous vertical fragmentation and allocation using modiûed Bond Energy Algorithm in distributed databases.

[5] Adrian Runceanu, Towards Vertical Fragmentation in Distributed Databases.

[6] Karlapalem, K., Li. (1995). *Partitioning Schemes for Object Oriented Database. In:* 5th InternationalWorkshop on Research Issues on Data Engineering : Distributed Object Management.

[7] Hoffer, A., Severance, D. G. (1975). *The Use of Cluster Analysis in Physical Database Design. In Proceedings of 1st VLDB Conference,* Mass.

[8] Karlapalem, K., Navathe, S. B., Morsi, M. M. A. (1994). Issues in *Distribu-tion design. of object-oriented databases*, in Distributed Object Management, Morgan Kaufmann Publishers.

[9] Navathe, S. B., Ceri, S. Wiederhold, G., Dou, J. (1984). Vertical partition-ing algorithms for database design.*In:* ACM TODS 9 (4).

[10] Marir, Farhi., Najjar, Yahiya., Mahmoud, Y., AlFaress, Hassan, I., Abdalla. An Enhanced Grouping Algorithm for Vertical Partitioning Problem in DDBs.

[11] Cornell, D. W., Yu, P. S. (1990). An Effective Approach to Vertical Partitioning for Physical Design of Relation Database, *IEEE Transactions on Software Engineering, 16-2.*

[12] Lee, S., Lim, H. (1997). *Extension of Vertical Technical Conference on Circuits/systems, Computers and Communications,* Japan.

[13] Mitchell, C. Components of a Distributed Database.

[14] Ceri, S., Pelagatti, G. (1984). *Distributed Databases Principles and Systems*. NY, McGraw Hill.

[15] Ezeife, C. I., Barker, K. (1993). *Vertical Class Fragmentation in a Distri-buted Object Based Svstem.* TR 94-03, Univ. of Manitoba DeRt. of Computer Science.

[16] Huang, Y. -F., Chen , J. -H. (2001). Fragment Allocation in Distributed Database Design.

[17] Jonker, W. (2000). *Databases in telecommunications: international workshop co-located with VLDB-99*, Edinburgh, Scotland, UK, September 6th 1999:

proceedings. Springer, Berlin.

[18] Ray, Chhanda. (2009). Distributed Database System, India: Dorling Kindersley, 2009.

[19] Z. Wei, C., Chi, Steen, M. (2008). Service-Oriented Data Denormalization for Scalable Web Applications, *In:* International World Wide Web Conference, Beijing.

Author's Information

Slamet Sudaryanto has received his S.T and M.Com from Computer Engineering Department, Yogyakarta Institute of Science & Technology (IST "AKPRIND" Yogyakarta) in 1997 and 2001 respectively. He is a lecturer in Computer Science Department at Dian Nuswantoro University, Semarang Indonesia. He is member of Indonesian Computer Electronics and Instrumentation Support Society (IndoCEISS). His research interest in distribution and integration database of public health.

Sudaryanto has received his S. Kom and M.Com from Computer Engineering Department, Dian Nuswantoro University) in 1996 and 2005 respectively. He is a lecturer in Computer Science Department at Dian Nuswantoro University, Semarang Indonesia. He is member of Indonesian Computer Electronics and Instrumentation Support Society (IndoCEISS). His research interest in Information System and Web Services.

Maryani Setyowati has received her S.KoSm and M.Kes from Public Health Department, Diponegoro University (UNDIP), Semarang, Indonesia in 2003 and 2007 respectively. Lecturer in Public Health Department, Dian Nuswantoro University, Semarang, Indonesia. Since 2010. She is head of medical record and health information. Her research interest in governance of public health data.