Applied Computing and Informatics (2015) **xxx**, xxx–xxx

Saudi Computer Society, King Saud University

Applied Computing and Informatics

(http://computer.org.sa)
www.ksu.edu.sa
www.sciencedirect.com

ORIGINAL ARTICLE

Hierarchical simultaneous vertical fragmentation and allocation using modified Bond Energy Algorithm in distributed databases

Hossein Rahimi, Fereshteh-Azadi Parand *, Davoud Riahi

Math and Computer Science Department, Allameh Tabataba'i University, Ahmad Ghasir St., Beheshti Av., Tehran, Iran

Received 31 December 2014; revised 2 March 2015; accepted 3 March 2015

KEYWORDS

Bond Energy Algorithm;
Distributed database system;
Data allocation and fragmentation;
Clustering

Abstract Designing an efficient Distributed Database System (DDBS) is considered as one of the most challenging problems because of multiple interdependent factors which are affecting its performance. Allocation and fragmentation are two processes which their efficiency and correctness influence the performance of DDBS. Therefore, efficient data fragmentation and allocation of fragments across the network sites are considered as an important research area in distributed database design. This paper presents an approach which simultaneously fragments data vertically and allocates the fragments to appropriate sites across the network. Bond Energy Algorithm (BEA) is applied with a better affinity measure that improves the generated clusters of attributes. The algorithm simultaneously generates clusters of attributes, calculates the cost of allocating each cluster to each site and allocates each cluster to the most appropriate site.

* Corresponding author. Tel.: +98 2188713160.
E-mail addresses: s.rahimi@atu.ac.ir (H. Rahimi), parand@atu.ac.ir (F.-A. Parand), d.riahi@atu.ac.ir (D. Riah).

☆ Peer review under responsibility of King Saud University.

ELSEVIER Production and hosting by Elsevier

http://dx.doi.org/10.1016/j.aci.2015.03.001

Please cite this article in press as: H. Rahimi et al., Hierarchical simultaneous vertical fragmentation and allocation using modified Bond Energy Algorithm in distributed databases, Applied Computing and Informatics (2015), http://dx.doi.org/10.1016/j.aci.2015.03.001

Results show more efficient clustering and allocation which gives better performance.

1. Introduction

Distributed databases reduce cost and increase performance and availability, but the design of Distribute Database Management Systems (DDBMS) is complicated. To make this process feasible it is divided into two steps: Fragmentation and Allocation. Fragmentation tries to split data into fragments, which should be allocated to sites over the network in the allocation stage. The process of fragmentation falls into two categories: Vertical Fragmentation and Horizontal Fragmentation. Vertical Fragmentation (VF) is partitioning relation R into disjoint sets of smaller relations while Horizontal Fragmentation (HF) is partitioning relation R into disjoint tuples. The allocation problem involves finding the optimal distribution of fragmentation to set F on site set S. There are four data allocation strategies applicable in a distributed relational database: centralized, fragmentation (partition), full replication, and partial replication (selective) [10]. When data is allocated, it might either be replicated or maintained as a single copy. So, fragment allocation can be either non-redundant or redundant. Under a non-redundant allocation scheme, exactly one copy of each fragment will exist across all the sites, while under redundant allocation schema, more than one copy of each fragment will exist across all the sites [12]. In this work, we combine fragmentation with partial replication of some clusters of attributes.

Allocation and fragmentation are interdependent and efficient data fragment allocation requires considering allocation constraints in the process of fragmentation, but in the most previous works these two steps are separated.

There are two general approaches toward solving the partitioning problem. One is to find the efficient solution by considering some of the constraints. In Hoffer [13] the storage capacity and retrieval cost constraints are the role factors. Each of these factors is weighted based on their amount of effect. The objective was to minimize the value of overall cost. The weights are calculated using linear programming approach so that the sum of the weights is equal to 1.

$$min(c_1 * storage\ cost + c_2 * retrieval\ cost) \quad (1)$$

Another good example of first set of approaches is proposed in Schkolnick [21]. The method tries to cluster records within an Information Management System (IMS) type hierarchical structure. The generated hierarchical tree is linear in the number of nodes. Heuristic grouping is used by the method presented in Hammer and Niamir [3]. It starts by assigning attributes to different positions. All potential types of grouping are considered and the one which represents the

Please cite this article in press as: H. Rahimi et al., Hierarchical simultaneous vertical fragmentation and allocation using modified Bond Energy Algorithm in distributed databases, Applied Computing and Informatics (2015), http://dx.doi.org/10.1016/j.aci.2015.03.001

greatest improvement over the current grouping candidate becomes the new candidate. Grouping and regrouping are iterated until no further improvement is likely. The main issue is the direction of movement, which has a dominant effect on the efficiency of the algorithm. Another heuristic approach is presented in Ma et al. [5] uses a cost model and targets at globally minimizing these costs. The major objective is to fragment based on efficiency of the most frequent queries. In Hoffer and Severance [14] clusters of similar attributes are generated using the affinity measure between pairs of attributes in conjunction with Bond Energy Algorithm (BEA). One of the major weaknesses is that the number of attributes in clusters are not decidable, and since it only considers pairwise attribute similarity, it is improper for larger numbers of attributes. Vertical fragmentation could also be done in more than one phase. This method is presented in Navathe et al. [23]. A two-phased approach separates fragments into overlapping and non-overlapping fragments. The first phase is based on empirical objective function and then it performs cost optimization by incorporating the knowledge of a specific application environment in the second phase. The method presented in Latiful and Shahidul [6] is a methodology for the design of distributed object databases that includes an analysis phase to indicate the most adequate fragmentation technique, a horizontal class fragmentation algorithm, and a vertical class fragmentation algorithm. The analysis phase is responsible for driving the choice between the horizontal and the vertical partitioning techniques, or even the combination of both, in order to assist distribution designers in the fragmentation phase of object databases. Baiao et al. [8] presents a three phased methodology for the design of distributed database that contains analysis phase, horizontal fragmentation algorithm phase, and vertical class fragmentation phase. The method illustrated in Abuelyaman [7] experimentally shows that moving an attribute that is loosely coupled in a partition improves hit ratio of attribute in partition.

A method for synchronized horizontal fragmentation and allocation is proposed in Abdalla [4]. This method introduces a heuristic cost model to find optimal fragment and allocation. Fragmentation is based on a set of simple predicates, and optimal allocation is the one which minimizes the cost function. An adaptable vertical partitioning method is presented in Jin and Myoung [15]. This article reviews Binary Vertical Partitioning (BVP) [18] and compares its results with the presented adaptable vertical partitioning (AVP) which uses a hierarchical method of fragmentation, creates a tree of partitions and finally selects the best result. A heuristic method is implemented in Adrian Runceanu [1]. It applies the approach of formulating an objective function, named Partition Evaluator [2], before developing (heuristic) algorithms for the partitioning problem. This approach enables studying the properties of algorithms with respect to an agreed upon objective function, and also to compare different algorithms for goodness using the same criteria for distributed database vertical fragmentation. A new heuristic algorithm which is based on a decomposition technique is developed in Mahmoud and Roirdon [16] that greatly reduces the computational complexity

Please cite this article in press as: H. Rahimi et al., Hierarchical simultaneous vertical fragmentation and allocation using modified Bond Energy Algorithm in distributed databases, Applied Computing and Informatics (2015), http://dx.doi.org/10.1016/j.aci.2015.03.001

Table 1 Model notations.

AFF	Attribute Affinity matrix
QA	Query Access matrix
CA	Clustered Affinity matrix
DM	Distance Matrix
AU	Attribute Usage matrix
TSC	Total Storage Cost
V	Volume of data allocation measured in characters
SC_{ij}	Storage cost of fragment i in site j
$aff(A_i, A_j)$	The affinity of attributes A_i and A_j
$freq_l(q_k)$	Access frequency of a query k on site l
$acc_l(q_k)$	Access per execution of query k on site l
S_{ij}	Similarity measure between A_i and A_j
MQA	Minimized Query Access
SC	Storage Cost
IIC	Iteration Input Cluster(is fed to next iteration)
LC	Leaf Cluster

of the problem of file allocation and capacity assignment in a fixed topology distributed network. Although using a partial replication scheme increases database efficiency, this benefit comes with some costs. This cost, which could potentially be high, consists of total storage cost, cost of local processing, and communication cost [19]. Some fragmentation methods along with query optimization , distribution optimization, and join optimization are covered in Haroun Rababaah, Hakimzadeh [9]. Here we take into account communication and local processing costs in combination with query access and calculate total storage cost separately.

Fragmentation and allocation are usually performed separately while these two steps of Distributed DBMS design are closely related to each other. The reason for separating the distribution design into two steps is to better deal with the complexity of the problem [17].

Here we present a method for VF, which applies BEA hierarchically with a modified similarity measure and simultaneously allocates the fragments to the most appropriate site. The model notations are listed in Table 1.

The rest of this article is organized as follows. Methods and different influencing factors are discussed in Section 2. The algorithm is described in details in Section 3. Section 4 draws comparative results of applying both the classic BEA and the presented method on one database. Finally, conclusion and future work are discussed in Section 5.

2. Methods

Allocation and fragmentation are interdependent problems where solving them simultaneously is difficult but results in better performance of applications. To the best of our knowledge, BEA is not applied to simultaneous fragmentation and allocation. Since in vertical partitioning attributes which are usually accessed

Please cite this article in press as: H. Rahimi et al., Hierarchical simultaneous vertical fragmentation and allocation using modified Bond Energy Algorithm in distributed databases, Applied Computing and Informatics (2015), http://dx.doi.org/10.1016/j.aci.2015.03.001

together are placed in one fragment, defining a precise measure of togetherness is critical. BEA uses affinity of attributes to create clusters of attributes, which are the most similar. It starts with Attribute Usage (AU) and Query Access (QA) matrices generates Attribute Affinity matrix (AFF) and finally creates Clustered Affinity matrix (CA) by positioning and re-positioning columns and rows of attributes. The *affinity* measure is too simple. The proposed affinity measure in BEA is basically based on simultaneous access of attribute A_i and attribute A_j of relation $R(A_1, A_2, \ldots, A_n)$ by query q_k for every query in $Q = (q_1, q_2, \ldots, q_q)$. In other words, Two attributes are considered similar if they are accessed by the same query. This is indicated in AU by $A_{ij} = 1$ and $A_{ik} = 1$ simultaneously for attributes j and k accessed by query i Considering the affinity of attributes A_i and A_j as $aff(A_i, A_j)$, access frequency of a query k on site l as $freq_l(q_k)$, and access per execution of query k on site l as $acc_l(q_k)$, the equation for affinity presented is as below [14].

$$aff(A_i, A_j) = \sum_{k|use(q_k,A_i)=1 \wedge use(q_k,A_j)=1} \sum_{\forall S_l} freq_l(q_k) * acc_l(q_k) \tag{2}$$

After generating AFF using the described affinity measure, clusters of attributes are created using the split function. The $Split(AFF)$ takes as input the AFF matrix, permutes its rows and columns, and generates a CA matrix. The permutation is done in such a way to maximize the following global measure.

$$\sum_{i=1}^{n}\sum_{j=1}^{n} aff_{ij}[aff_{i,j-1} + aff_{i,j+1} + aff_{i-1,j} + aff_{i+1,j}] \tag{3}$$

where

$$aff_{i,0} = aff_{0,j} = aff_{i,n+1} = aff_{n+1,j} = 0 \tag{4}$$

The last set of conditions takes care of the cases where an attribute is being placed in CA to the left of the leftmost attribute or to the right of the rightmost attribute during column permutations, and prior to the topmost row and following the last row during row permutations. In the process of splitting the bond between two attributes i and j and the net contribution to the global affinity measure of placing the attribute k between i and j play key roles. The bond between attributes i and j is defined as

$$bond(A_i, A_j) = \sum_{k=1}^{n} aff(A_k, A_i) aff(A_k A_j) \tag{5}$$

The net contribution of placing the attribute k between i and j is defined as

$$\begin{aligned} cont(A_i, A_k, A_j) \quad &= 2bond(A_i, A_k) + 2bond(A_k, A_j) \\ &\quad -2bond(A_i, A_j) \end{aligned} \tag{6}$$

The split function generates the Clustered Affinity Matrix in two steps:

Algorithm 1. Simultaneous VF and allocation

Require:
Communication Cost Matrix
Attribute Usage Matrix (AU)
Query Access Matrix (QA)
Number of attributes
Output:
Clustered Attribute Matrices as a Tree
Allocated clusters to sites
1: Optimizing Communication Cost Matrix and generating DM
2: Generating Minimized Query Access matrix (MQA)
$MQA = \sum_i \sum_k DM * QA$
3: $IIC \leftarrow AU$
4: **while** *Number of attributes in* $IIC > 3$ **do**
Run Modified BEA Algorithm (IIC,MQA)
Add LC and IIC to Tree
5: **end while**
6: Calculate Storage Cost
$\sum_{i=1}^{m} X_{ij} * SC_{ij} * V$
7: Allocate each cluster to site with minimum cost

- **Initialization:** Place and fix one of the columns of AFF matrix arbitrarily into CA matrix.
- **Iteration:** Pick each of the remaining $n - i$ columns where i is the number of columns already placed in CA and try to place them in the remaining $i + 1$ positions in the CA. Choose the placement that makes the greatest contribution to the global affinity measure described above. Continue this until no more columns remain to be placed.

Since the clustering result of BEA is the split border between two sets of attributes, BEA does not work efficiently for larger databases. Therefore, we need a better approach to identify more partitioning candidates. As we infer similarity of two attributes when they have concurrent occurrence in a query, concurrent absence of them for the same query could also be considered as a weighted measure of similarity. Furthermore, single occurrence of each attribute could be considered as a weighted measure of dissimilarity. Consider n_{00}, n_{11}, and n_{01} and n_{10} be the number of simultaneous absence of attributes, presence of attributes, and single occurrence of each attribute for one query in the Affinity Usage (AU) matrix, respectively. Similarity measure S_{ij} which is described in Xu and Wunsch [20] uses n_{11} and n_{00} in the nominator of the fraction to show similarity and n_{10} and n_{01} in the denominator to indicate dissimilarity.

Please cite this article in press as: H. Rahimi et al., Hierarchical simultaneous vertical fragmentation and allocation using modified Bond Energy Algorithm in distributed databases, Applied Computing and Informatics (2015), http://dx.doi.org/10.1016/j.aci.2015.03.001

$$S_{ij} = \frac{n_{11} + n_{00}}{n_{11} + n_{00} + w_1(n_{01} + n_{10})} \tag{7}$$

This measure computes the match between two objects directly. Unmatched pairs are weighted based on their contribution to the similarity. If one considers simple matching similarity w_1 equals to one. In constrained-means clustering [24] the coefficient is considered equal to 2. Gower [11] suggests w_1 to be equal to 1/2. It can be concluded, choosing an appropriate value for the weight w_1 depends the approach and also on the structure and definition of the database itself.

Each one of queries can be accessed different times on each site. The frequency of query access on each site is described in the Query Access (QA) matrix. The entry QA_{ij} indicates the number of times in which query i is accessed in site j. On the other hand communication costs between sites of a distributed database play a key role in the performance of a distributed DB. Distance Matrix (DM) is the asymmetric square matrix that reflects these costs which can be minimized using the method described in Bentley and Dittman [25]. Multiplying DM in QA generates a new matrix in which the influence of communication costs between sites and query access per site is considered simultaneously and since DM is minimized distance matrix then the resulted matrix will be the Minimized Query Access (MQA) matrix.

$$MQA = \sum_i \sum_k DM * QA \tag{8}$$

The Total Storage Cost (TSC) of each attribute in each site depends on storage cost for one item and the total volume of that site.

$$TSC = \sum_{i=1}^{m} X_{ij} * SC_{ij} * V \tag{9}$$

where

$$X_{ij} = 1 \text{ if fragment } i \text{ is allocated to site } j \tag{10}$$

The attribute with minimum storage cost for each site will be allocated to that site. Eq. 9 is also applied to the remaining attributes and sites with minimum cost value allocate the attributes.

3. Algorithm

The algorithm (**Algorithm 1**) works with communication cost between network sites, QA matrix, AU matrix, and attributes count as inputs and generates the tree of clustered attributes along with allocating them to sites.

Please cite this article in press as: H. Rahimi et al., Hierarchical simultaneous vertical fragmentation and allocation using modified Bond Energy Algorithm in distributed databases, Applied Computing and Informatics (2015), http://dx.doi.org/10.1016/j.aci.2015.03.001

The Algorithm works hierarchically and gradually creates a cluster tree. In each iteration it generates two sets of attributes. The larger set of more similar attributes which we call it Iteration Input Cluster (IIC) is used as input for the next iteration. The other smaller set is called Leaf Cluster (LC) since it is separated and placed as leaf node in the tree. In the first step DM is generated by optimizing Communication Cost matrix using Whitten et al. [25]. Then MQA is generated by multiplying QA in DM matrix. The next step is to initialize IIC by AU matrix. The algorithm continues with iterating on the modified BEA algorithm, which will be explained later, until attribute count in IIC is equal to 3. Since in each iteration the most similar attributes group in one IIC, we assume after this number of iterations, the resulted IIC contains the most similar attributes of all therefore there is no need to go further. Next, the storage cost for each attribute on each site is calculated and finally based on these costs, each cluster of attributes is allocated to the most appropriate site. The last IIC is allocated to all sites.

The modified BEA algorithm is actually modifies the affinity measure in the original BEA. As it is mentioned before BEA is simply using the concurrent occurrence of attributes to create AFF matrix. In the modified BEA presented here, other possibilities are considered. With S_{ij} borrowed from Xu and Wunsch [20], taking co-absence into account, and calculating n_{00}, n_{01}, and n_{10} the new affinity measure S_{ij} is

$$S_{ij} = \frac{n_{11} + w_1 n_{00}}{n_{11} + w_1 n_{00} + coef} \tag{11}$$

The weights of w_1 and w_2 are between 0 and 1 since n_{00}, n_{01}, and n_{10} have less positive effect on similarity in comparison to n_{11}. Furthermore, it can be inferred that w_1 should be greater than w_2. The approaches to calculate the value of each weight are dependent on the structure and definition of the table and their relations in the database. Gower and Legendre measure [11] and Rogers and Tanimoto measure [22] are some methods to calculate values of weights. Each of the weights is calculated considering the structure and definition of the database and queries. The structure of the database gives us some information regarding to the relations of different attributes. Therefore, by considering the queries, initial values of the weights are inferred and after generating the elementary results, the weights are slightly changed in such a way that results reflect the true expected relations between attributes with consideration to the structure of the database.

As we mentioned earlier simultaneous absence of attributes can give us some sense of similarity of

Please cite this article in press as: H. Rahimi et al., Hierarchical simultaneous vertical fragmentation and allocation using modified Bond Energy Algorithm in distributed databases, Applied Computing and Informatics (2015), http://dx.doi.org/10.1016/j.aci.2015.03.001

Algorithm 2. Modified BEA algorithm

Require:
Attribute Query Matrix
Query Access Matrix
Result:
AFF Matrix
1: $S \leftarrow MQA$
2: **for** *each attribute number i* **do**
3: $QS_i \leftarrow sum(S_{ij})$
4: **end for**
5: **for** *each attribute number i* **do**
6: **for** *each attribute number j* **do**
7: *initialize* $n_{00}, n_{11}, n_{01}, n_{10}$ *by* 0
8: **if** $(i == j)$ **then**
9: $AFF_{ij} \leftarrow sum(A_j) * QS$
10: **else**
11: **for** *each query number k* **do**
12: *calculating* $n_{00}, n_{11}, n_{01}, n_{10}$
13: **if** $(n_{01} == 0$ *and* $n_{10} > 0)$ *or* $(n_{10} == 0$ *and* $n_{01} > 0)$ **then**
14: $coef \leftarrow (-1)(n_{01} + n_{10}) * w_1$
15: **else**
16: $coef \leftarrow (|n_{01} - n_{10}|) * w_1$
17: **end if**
18: $S_{ij} \leftarrow (n_{11} + w_2 * n_{00})/(n_{11} + w_2 * n_{00}) + coef$
19: **end for**
20: **end if**
21: $AFF_{ij} \leftarrow S_i * QS_i$
22: **end for**
23: **end for**
24: *call Function Split*(*AFF*)

the attributes. On the other hand, since this effect is marginal in comparison to the effect of simultaneous presence, n_{00} has some weighted effect on the affinity measure and therefore w_1 have a value between 0 and 1.

The variable *coef* in the denominator is reflecting the effect of n_{01} and n_{10}. There are four different possibilities. When $n_{01} > 0$ and $n_{10} = 0$, it indicates that for two attributes of A_i and A_j, all queries which access A_i do not access A_j. This means that these attributes have some level of similarity. As a result the S_{ij} should get greater values so the weighted measure in the denominator, w_2, should be negative. This is shown in Lines 12 and 13 of **Algorithm 2**. The same is for the case in which $n_{10} > 0$ and $n_{01} = 0$. Other possibility is that both n_{01} and n_{10} are greater than 0. This condition means A_i and A_j do not have the same behavior upon different queries which are accessing them. This has negative effect on the similarity, so the weighted measure in the denominator, w_2, should be positive. This is shown in Line 15. After calculating the AFF matrix, the algorithm calls the split function which we described in Section 2 and creates clusters of attribute.

Please cite this article in press as: H. Rahimi et al., Hierarchical simultaneous vertical fragmentation and allocation using modified Bond Energy Algorithm in distributed databases, Applied Computing and Informatics (2015), http://dx.doi.org/10.1016/j.aci.2015.03.001

4. Case study

In order to estimate the amount of improvement and correctness of our algorithm, we applied both the classic BEA and our algorithm on database of Terminal Management System (TMS). TMS is a server which is connected to stores' (or supermarkets') terminal with a unique serial number. Depending on the terminal it can download or update terminal information or operating system. Since stores are located in different places, TMS can obviously work better with distributed database. Each terminal has a unique serial number, one task is defined for each terminal group. These tasks contain one or more files which can be associated to a group of terminals. Each Terminal, group of terminal and task has one table. A simple schema of tables and their relations is illustrated in Fig. 4. After reviewing the transactions, eight most frequent transactions and considering the relations of tables in TMS, eight attributes (illustrated in Table 2) for distributing in seven sites were selected.

The AU, QA, and DM matrices are as shown in Figs. 1–3. The query access input for both algorithms were MQA. The weights w_1 and w_2 in our algorithm were set to 0.7 and 0.3, respectively. The resulted clustering tree for each algorithm is shown in Fig. 5. As it can be observed, both algorithms behave the same until the fourth iteration. The classic BEA separates attribute number 2 and puts

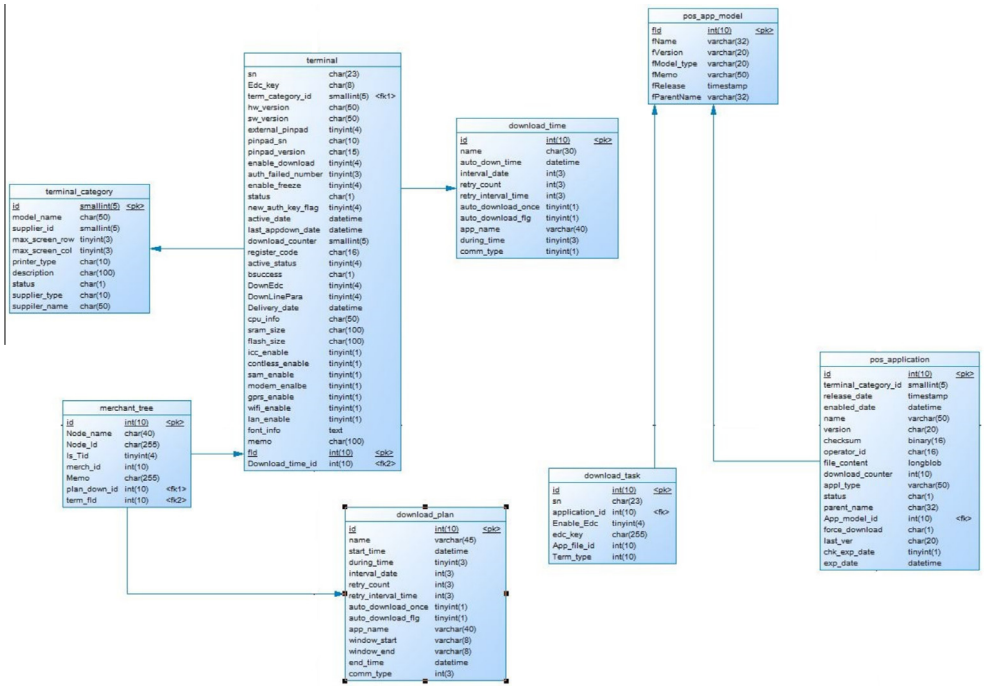

Fig. 4 Table relations in TMS.

Please cite this article in press as: H. Rahimi et al., Hierarchical simultaneous vertical fragmentation and allocation using modified Bond Energy Algorithm in distributed databases, Applied Computing and Informatics (2015), http://dx.doi.org/10.1016/j.aci.2015.03.001

Table 2 List of attributes and their related tables.

No.	List of attributes	Related table
1	*model – name*	*terminal – category*
2	*term – fid*	*terminal*
3	*hw – version*	*terminal*
4	*pinpad – version*	*terminal*
5	*flash – size*	*terminal*
6	*app – name*	*download – time*
7	*interval – date*	*download – time*
8	*start – time*	*download – plan*

	S_1	S_2	S_3	S_4	S_5	S_6	S_7
q_1	28	44	45	31	29	32	10
q_2	47	31	29	33	0	1	7
q_3	35	50	43	37	6	31	9
q_4	29	26	37	45	43	18	2
q_5	41	24	29	50	24	2	32
q_6	44	40	12	39	43	24	14
q_7	50	11	33	29	10	9	27
q_8	0	25	4	47	28	6	35

Fig. 1 Query Access matrix (QA) for seven sites.

	S_1	S_2	S_3	S_4	S_5	S_6	S_7
S_1	0	7	12	8	4	9	4
S_2	7	0	11	7	8	10	3
S_3	12	11	0	10	16	13	8
S_4	8	7	10	0	12	6	4
S_5	4	8	16	12	0	9	8
S_6	9	10	13	6	9	0	7
S_7	4	3	8	4	8	7	0

Fig. 2 Communcation Cost Distance Matrix (DM) for seven sites.

	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8
q_1	1	0	1	1	0	1	1	0
q_2	1	1	1	1	0	1	1	0
q_3	0	1	1	0	1	0	0	0
q_4	1	1	0	1	1	1	1	1
q_5	1	1	1	1	0	1	1	1
q_6	1	0	1	0	0	1	1	1
q_7	0	1	1	1	0	0	0	1
q_8	1	1	0	0	1	0	1	0

Fig. 3 Attribute Usage matrix (AU) for 8 queries.

Please cite this article in press as: H. Rahimi et al., Hierarchical simultaneous vertical fragmentation and allocation using modified Bond Energy Algorithm in distributed databases, Applied Computing and Informatics (2015), http://dx.doi.org/10.1016/j.aci.2015.03.001

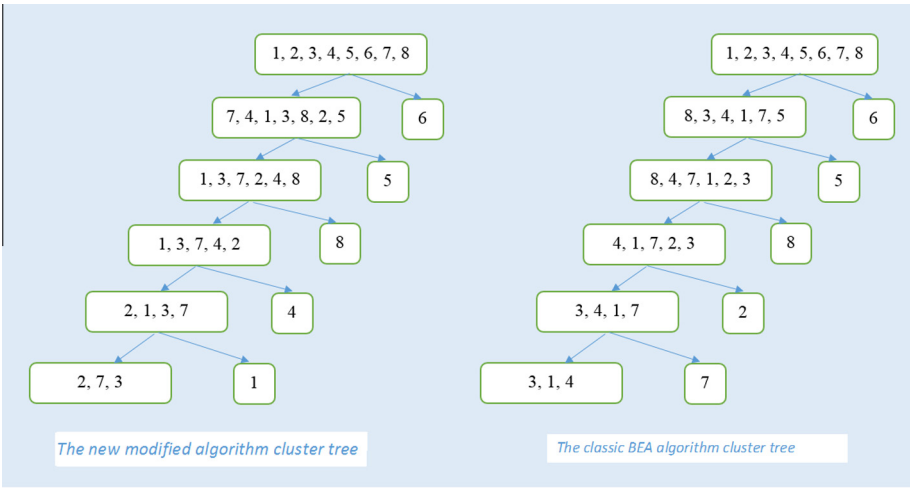

Fig. 5 Hierarchical attribute clustering tree.

Table 3 The Similarity of attributes.

Attributes	n_{11}	n_{00}	n_{10}	n_{01}	*coef*	S_{ij}
A_2 *and* A_1	4	0	2	2	0	1
A_2 *and* A_3	4	0	2	2	0	1
A_2 *and* A_7	4	0	2	2	0	1
A_4 *and* A_1	4	1	2	1	0.35	0.93
A_4 *and* A_3	4	1	2	1	0.35	0.93
A_4 *and* A_7	4	1	2	1	0.35	0.93
A_1 *and* A_3	4	0	2	2	0	1
A_1 *and* A_7	6	2	0	0	0	1
A_3 *and* A_7	4	0	2	2	0	1

attributes number 3, 4, 1, and 7 in a cluster. On the other hand the modified algorithm separates attribute number 4 and clusters attributes number 2, 1, 3, and 7. Considering the conditions applied in our algorithm, the *coef* and S_{ij} are calculated and illustrated in Table 3. It is obvious that A_4 is less similar to other attributes than A_2 therefore it has been separated correctly. We can conclude that the new algorithm considers better measure and clusters the attributes much better.

5. Conclusion

Distributed databases reduce cost of update and retrieval of information and increase performance and availability, but the design of DDBMS is more complicated than designing centralized database. One of the major challenges which greatly affects DDBS performance is fragmentation and allocation of fragments to sites. Allocation and fragmentation can logically be merged and done

Please cite this article in press as: H. Rahimi et al., Hierarchical simultaneous vertical fragmentation and allocation using modified Bond Energy Algorithm in distributed databases, Applied Computing and Informatics (2015), http://dx.doi.org/10.1016/j.aci.2015.03.001

simultaneously. In this paper we proposed a method that merges vertical fragmentation and allocation. To achieve this goal we applied Bond Energy Algorithm with a modified affinity measure in a hierarchical process and simultaneously calculated the cost of data allocation for each site and assigned fragment to the appropriate site. The use of the hierarchical process resulted in clustering sets of more similar attributes and better data fragmentation. On the other hand, by performing simultaneous cost calculation we took interdependency of data fragmentation and allocation into account.

An extension to the work could cover optimizing the cost function for data allocation considering the retrieval and update frequency for each attribute and applying better approach to calculate weights for similarity measure.

References

[1] Adrian Runceanu, Fragmentation in distributed databases, Innovations and Advanced Techniques in Systems, Computing Sciences and Software Engineering, 2008, pp. 57–62.
[2] G.S. Chinchwadkar, A. Goh, An overview of vertical partitioning in object oriented databases, Comput. J. 42 (1) (1999).
[3] M. Hammer, B. Niamir, A heuristic approach to attribute partitioning, in: Proceedings ACM SZGMOD International Conference on Management of Data, Boston, Mass., ACM, New York, 1979.
[4] Hassan I. Abdalla, A synchronized design technique for efficient data distribution, Comput. Human Behav. 30 (2014) 427–435.
[5] H. Ma, K.D. Schewe, M. Kirchberg, A heuristic approach to vertical fragmentation incorporating query information, in: Proc. 7th International Baltic Conference on Databases and Information Systems (DB and IS), 2006, pp. 69–76.
[6] Latiful A.S.M. Hoque, Shahidul Islam Khan, A New Technique for Database Fragmentation in Distributed Systems, International Journal of Computer Applications 5(9):2024, August 2010. Published By Foundation of Computer Science.
[7] E.S. Abuelyaman, An optimized scheme for vertical partitioning of a distributed database, Int. J. Comput. Sci. Netw. Sec. 8 (1) (2008).
[8] F. Baiao, M. Mattoso, G. Zaverucha, A distribution design methodology for object DBMS, Distrib. Parallel Databases 16 (1) (2004) 4590.
[9] Haroun Rababaah, H. Hakimzadeh, Distributed Databases: Fundamentals and research, Advanced Database B561, Spring 2005.
[10] N. Iacob, Data replication in distributed environments, Annals of the Constantin Brncusi, University of Trgu Jiu, Economy Series, Issue 4, 2010.
[11] J. Gower, A general coefficient of similarity and some of its properties, Biometrics 27 (1971) 857872.
[12] J.O. Hauglid, N.H. Ryeng, DYFRAM: dynamic fragmentation and replica management in distributed database systems, Distrib. Parallel Databases 28 (2010) 157185.
[13] J.A. Hoffer, An integer programming formulation of computer database design problems, Znf. Sci. 11 (July 1976) 29–48.
[14] J.A. Hoffer, D.G. Severance, The use of cluster analysis in physical database design, in: Proceedings 1st International Conference on Very Large Databases, Framingham, Mass., 1975.
[15] Jin Hyun Son, Myoung Ho Kim, An adaptable vertical partitioning method in distributed systems, J. Syst. Softw. 73 (2004) 551–561.
[16] S. Mahmoud, J.S. Roirdon, Optimal allocation of resources in distributed information networks, ACM Trans. Database Syst. 1 (1976) 1.
[17] M.T. Ozsu, P. Valduriez, Principles of Distributed Database Systems, Alan Apt, New Jersey, 1999.
[18] S. Navathe, S. Ceri, G. Wiederhold, J. Dou, Vertical partitioning algorithms for database design, ACM Trans. Database Syst. 9 (4) (1984) 680710.
[19] S.K. Rahimi, F.S. Haug, Distributed Database Management Systems, A John Wiley and Sons Inc. Publication, IEEE Computer Society, 2010.

Please cite this article in press as: H. Rahimi et al., Hierarchical simultaneous vertical fragmentation and allocation using modified Bond Energy Algorithm in distributed databases, Applied Computing and Informatics (2015), http://dx.doi.org/10.1016/j.aci.2015.03.001

[20] Rui Xu, Donald Wunsch II, Survey of clustering algorithms, IEEE Trans. Neural Netw. 16 (3) (2005).
[21] M. Schkolnick, A clustering algorithm for hierarchical structure, ACM Trans. Database Syst. 2 (1977) 1.
[22] Pang-Ning Tan, Michael Steinbach, Vipin Kumar, Introduction to Data Mining, 2005, ISBN: 0-321-32136-7.
[23] Shamkant Navathe, Stefano Ceri, Gio Wiederhold, Jinglie Dou, Vertical partitioning algorithms for database design, ACM Trans. Database Syst. 9 (4) (December 1984).
[24] K. Wagstaff, S. Rogers, S. Schroedl, Constrained -means clustering with background knowledge, in: Proc. 8th Int. Conf. Machine Learning, 2001, pp. 577–584.
[25] J. Whitten, L. Bentley, K. Dittman, Systems Analysis and Design Methods, sixth ed., McGraw-Hill, 2004.

Please cite this article in press as: H. Rahimi et al., Hierarchical simultaneous vertical fragmentation and allocation using modified Bond Energy Algorithm in distributed databases, Applied Computing and Informatics (2015), http://dx.doi.org/10.1016/j.aci.2015.03.001