

Saudi Computer Society, King Saud University

Applied Computing and Informatics

([http://computer.org.sa)](http://computer.org.sa/) [www.ksu.edu.sa](http://www.ksu.edu.sa/) [www.sciencedirect.com](http://www.sciencedirect.com/science/journal/22108327)

ORIGINAL ARTICLE

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



Hossein Rahimi, Fereshteh-Azadi Parand [\*](#_bookmark0), 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

Available online 23 April 2015

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 perfor- mance. Allocation and fragmentation are two processes which their efficiency and correctness influ- ence 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 simultane- ously generates clusters of attributes, calculates the cost of allocating each cluster to each site and allocates each cluster to the most appropriate site. Results show more efficient clustering and allo- cation which gives better performance.

© 2015 The Authors. Production and hosting by Elsevier B.V. on behalf of King Saud University. This is

KEYWORDS

Bond Energy Algorithm; Distributed database system; Data allocation and frag- mentation;

Clustering

an open access article under the CC BY-NC-ND license (<http://creativecommons.org/licenses/by-nc-nd/4.0/>).

1. Introduction

Distributed databases reduce cost and increase performance and availability, but the design of Distribute Database Man- agement Systems (DDBMS) is complicated. To make this pro- cess feasible it is divided into two steps: Fragmentation and

\* Corresponding author. Tel.: +98 2188713160.

E-mail addresses: [s.rahimi@atu.ac.ir](mailto:s.rahimi@atu.ac.ir) (H. Rahimi), [parand@atu.ac.ir](mailto:parand@atu.ac.ir) (F.-A. Parand), [d.riahi@atu.ac.ir](mailto:d.riahi@atu.ac.ir) (D. Riahi).

q Peer review under responsibility of King Saud University.

**Production and hosting by Elsevier**

Allocation. Fragmentation tries to split data into fragments, which should be allocated to sites over the network in the allo- cation stage. The process of fragmentation falls into two cate- gories: Vertical Fragmentation and Horizontal Fragmentation. Vertical Fragmentation (VF) is partitioning relation R into disjoint sets of smaller relations while Horizon- tal 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 rela- tional database: centralized, fragmentation (partition), full replication, and partial replication (selective) [[10]](#_bookmark21). 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

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

2210-8327 © 2015 The Authors. Production and hosting by Elsevier B.V. on behalf of King Saud University. This is an open access article under the CC BY-NC-ND license (<http://creativecommons.org/licenses/by-nc-nd/4.0/>).

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]](#_bookmark14). In this work, we combine fragmentation with partial replication of some clusters of attributes.

Allocation and fragmentation are interdependent and effi- cient 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 parti- tioning problem. One is to find the efficient solution by consid- ering some of the constraints. In Hoffer [[13]](#_bookmark14) 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*) (1)

Another good example of first set of approaches is proposed in Schkolnick [[21]](#_bookmark15). The method tries to cluster records within an Information Management System (IMS) type hierarchical structure. The generated hierarchical tree is linear in the num- ber of nodes. Heuristic grouping is used by the method pre- sented in Hammer and Niamir [[3]](#_bookmark16). It starts by assigning attributes to different positions. All potential types of grouping are considered and the one which represents the 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 direc- tion of movement, which has a dominant effect on the effi- ciency of the algorithm. Another heuristic approach is presented in Ma et al. [[5]](#_bookmark16) uses a cost model and targets at glob- ally minimizing these costs. The major objective is to fragment based on efficiency of the most frequent queries. In Hoffer and Severance [[14]](#_bookmark14) 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 weak- nesses is that the number of attributes in clusters are not decid- able, and since it only considers pairwise attribute similarity, it is improper for larger numbers of attributes. Vertical fragmen- tation could also be done in more than one phase. This method is presented in Navathe et al. [[23]](#_bookmark17). A two-phased approach sep- arates fragments into overlapping and non-overlapping frag- ments. The first phase is based on empirical objective function and then it performs cost optimization by incorporat- ing the knowledge of a specific application environment in the second phase. The method presented in Latiful and Shahidul

[[6]](#_bookmark16) is a methodology for the design of distributed object data- bases that includes an analysis phase to indicate the most ade- quate 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]](#_bookmark18) 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 Abuelya- man [[7]](#_bookmark16) 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]](#_bookmark16). This method introduces a heuristic cost model to find optimal fragment and allocation. Fragmentation is based on a set of simple predicates, and opti- mal allocation is the one which minimizes the cost function. An adaptable vertical partitioning method is presented in Jin and Myoung [[15]](#_bookmark14). This article reviews Binary Vertical Parti- tioning (BVP) [[18]](#_bookmark14) 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 imple- mented in Adrian Runceanu [[1]](#_bookmark16). It applies the approach of for- mulating an objective function, named Partition Evaluator [[2]](#_bookmark16), 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 fragmenta- tion. A new heuristic algorithm which is based on a decompo- sition technique is developed in Mahmoud and Roirdon [[16]](#_bookmark14) that greatly reduces the computational complexity of the prob- lem of file allocation and capacity assignment in a fixed topol- ogy 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 communica- tion cost [[19]](#_bookmark14). Some fragmentation methods along with query optimization , distribution optimization, and join optimization are covered in Haroun Rababaah, Hakimzadeh [[9]](#_bookmark19). 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 sepa- rately while these two steps of Distributed DBMS design are closely related to each other. The reason for separating the dis- tribution design into two steps is to better deal with the com- plexity of the problem [[17]](#_bookmark14).

Here we present a method for VF, which applies BEA hier- archically with a modified similarity measure and simultane- ously allocates the fragments to the most appropriate site. The model notations are listed in [Table 1](#_bookmark1).

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

*SCij* Storage cost of fragment *i* in site *j aff*(*Ai*; *Aj*) The aﬃnity of attributes *Ai* and *Aj freql* (*qk*) Access frequency of a query *k* on site *l*

*accl*(*qk*) Access per execution of query *k* on site *l Sij* Similarity measure between *Ai* and *Aj*

MQA Minimized Query Access

SC Storage Cost

IIC Iteration Input Cluster(is fed to next iteration)

LC Leaf Cluster

The rest of this article is organized as follows. Methods and different influencing factors are discussed in Section [2](#_bookmark2). The

*cont*(*Ai*; *Ak*; *Aj*) = 2*bond*(*Ai*; *Ak*)+ 2*bond*(*Ak*; *Aj*)

(6)

—2*bond*(*A* ; *A* )

*i j*

algorithm is described in details in Section [3](#_bookmark6). Section [4](#_bookmark7) 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](#_bookmark9).

1. Methods

Allocation and fragmentation are interdependent problems where solving them simultaneously is difficult but results in bet- ter performance of applications. To the best of our knowledge, BEA is not applied to simultaneous fragmentation and alloca- tion. Since in vertical partitioning attributes which are usually accessed together are placed in one fragment, defining a precise measure of togetherness is critical. BEA uses affinity of attri- butes 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* mea- sure is too simple. The proposed affinity measure in BEA is basi- cally based on simultaneous access of attribute *Ai* and attribute

*Aj* of relation *R*(*A*1; *A*2; .. . ; *An*) by query *qk* for every query in

*Q* = (*q*1; *q*2; .. . ; *qq*). In other words, Two attributes are consid-

ered similar if they are accessed by the same query. This is indi- cated in AU by *Aij* = 1 and *Aik* = 1 simultaneously for attributes *j* and *k* accessed by query *i* Considering the affinity

of attributes *Ai* and *Aj* as *aff*(*Ai*; *Aj*), access frequency of a query

*k* on site *l* as *freq* (*q* ), and access per execution of query *k* on site

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

*MQA* = P*i* P*kDM* \* *QA*

3: *IIC* ← *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

2: Generating Minimized Query Access matrix (MQA)

6: Calculate Storage Cost

P

*m*

*i*=1

*X* \* *SC* \* *V*

*ij ij*

7: Allocate each cluster to site with minimum cost

* + Initialization: Place and fix one of the columns of AFF matrix arbitrarily into CA matrix.

*l k* ● Iteration: Pick each of the remaining *n* — *i* columns where *i*

*l* as *accl*(*qk*), the equation for affinity presented is as below [[14]](#_bookmark14).

*aff*(*Ai*; *Aj*) = X X*freql* (*qk*)\* *accl*(*qk*) (2)

*k*|*use*(*qk* ;*Ai* )=1Λ*use*(*qk*;*Aj* )=1 6*Sl*

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.

*n n*

XX

*affij*[*affi*;*j*—1 + *affi*;*j*+1 + *affi*—1;*j* + *affi*+1;*j*] (3)

*i*=1 *j*=1

where

*affi*;0 = *aff*0;*j* = *affi*;*n*+1 = *affn*+1;*j* = 0 (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 attri- bute 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 split- ting the bond between two attributes *i* and *j* and the net con-

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. Fur-

thermore, single occurrence of each attribute could be consid- ered as a weighted measure of dissimilarity. Consider *n*00; *n*11, and *n*01 and *n*10 be the number of simultaneous absence of attri- butes, presence of attributes, and single occurrence of each

attribute for one query in the Affinity Usage (AU) matrix, respectively. Similarity measure *Sij* which is described in Xu and Wunsch [[20]](#_bookmark14) uses *n*11 and *n*00 in the nominator of the frac- tion to show similarity and *n*10 and *n*01 in the denominator to indicate dissimilarity.

tribution to the global affinity measure of placing the

attribute *k* between *i* and *j* play key roles. The bond between *S* =

*n*11

+ *n*00

(7)

*ij n* + *n*

+ *w* (*n* + *n* )

attributes *i* and *j* is defined as

11 00

1 01 10

*bond*(*Ai*; *Aj*) =

X*n*

*k*=1

*aff*(*Ak*; *Ai*)*aff*(*AkAj*) (5)

This measure computes the match between two objects directly. Unmatched pairs are weighted based on their contri- bution to the similarity. If one considers simple matching sim-

The net contribution of placing the attribute *k* between *i*

and *j* is defined as

ilarity *w*1 equals to one. In constrained-means clustering [[24]](#_bookmark22) the coefficient is considered equal to 2. Gower [[11]](#_bookmark23) suggests *w*1 to be equal to 1/2. It can be concluded, choosing an appro-

priate 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 *QAij* indicates the

to create AFF matrix. In the modified BEA presented here, other possibilities are considered. With *Sij* borrowed from Xu and Wunsch [[20]](#_bookmark14), taking co-absence into account, and cal-

culating *n*00; *n*01, and *n*10 the new affinity measure *Sij* is

*n*11 + *w*1*n*00

number of times in which query *i* is accessed in site *j*. On the other hand communication costs between sites of a distributed

*Sij* =

11

*n*

+ *w*1*n*00

+ *coef* (11)

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]](#_bookmark24). 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 dis- tance matrix then the resulted matrix will be the Minimized Query Access (MQA) matrix.

*MQA* = XX*DM* \* *QA* (8)

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]](#_bookmark23) and Rogers and Tanimoto measure [[22]](#_bookmark20) are some methods to calculate values of weights. Each of the weights is calculated considering the structure and definition of the data- base and queries. The structure of the database gives us some information regarding to the relations of different attributes.

*i k* Therefore, by considering the queries, initial values of the

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.

*m*

X

*TSC* = *Xij* \* *SCij* \* *V* (9)

*i*=1

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

where

*Xij* = 1 *if fragment i is allocated to site j* (10)

The attribute with minimum storage cost for each site will be allocated to that site. Eq. [9](#_bookmark4) is also applied to the remaining attributes and sites with minimum cost value allocate the attributes.

1. Algorithm

The algorithm ([Algorithm 1](#_bookmark3)) 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.

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 Iter- ation Input Cluster (IIC) is used as input for the next iteration. The other smaller set is called Leaf Cluster (LC) since it is sep- arated 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]](#_bookmark24). 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 sim- ilar attributes group in one IIC, we assume after this number of iterations, the resulted IIC contains the most similar attri- butes 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 affin- ity measure in the original BEA. As it is mentioned before BEA is simply using the concurrent occurrence of attributes

Algorithm 2. Modified BEA algorithm

Require:

Attribute Query Matrix Query Access Matrix Result:

*AFF Matrix*

1: *S* ← *MQA*

2: for *each attribute number i* do

3: *QSi* ← *sum*(*Sij*)

4: end for

5: for *each attribute number i* do

6:

7:

8:

9:

10:

11:

12:

13:

then

14:

15:

16:

17:

18:

19:

20:

21:

22:

for *each attribute number j* do

*initialize n*00; *n*11; *n*01; *n*10 *by* 0

if (*i* == *j*) then

*AFFij* ← *sum*(*Aj*)\* *QS*

else

for *each query number k* do

*calculating n*00; *n*11; *n*01; *n*10

if (*n*01 == 0 *and n*10 > 0) *or* (*n*10 == 0 *and n*01 > 0)

*coef* ←(—1)(*n*01 + *n*10)\* *w*1

else

*coef* ←(|*n*01 — *n*10|) \* *w*1

end if

*Sij* ← (*n*11 + *w*2 \* *n*00)/(*n*11 + *w*2 \* *n*00)+ *coef*

end for end if

*AFFij* ← *Si* \* *QSi*

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.

1. Case study





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



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



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

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 *Ai* and *Aj*, all queries which access *Ai* do not access *Aj*. This

means that these attributes have some level of similarity. As a result the *Sij* should get greater values so the weighted mea- sure in the denominator, *w*2, should be negative. This is shown in Lines 12 and 13 of [Algorithm 2](#_bookmark5). 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 *Ai* and *Aj* 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](#_bookmark2) and creates clusters of attribute.

In order to estimate the amount of improvement and correct- ness 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 super- markets’) 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 data- base. 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](#_bookmark10). After reviewing the transactions, eight most frequent transactions and considering the relations of tables in TMS, eight attributes (illustrated in [Table 2](#_bookmark11)) for distributing in seven sites were selected.

The AU, QA, and DM matrices are as shown in [Figs. 1–3](#_bookmark8). 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](#_bookmark13). As it can be observed, both algorithms behave the same until the fourth iteration. The classic BEA separates attribute number 2 and puts 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 *Sij* are calculated and illustrated in [Table 3](#_bookmark12). It is obvi- ous 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.

1. 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 cen- tralized database. One of the major challenges which greatly affects DDBS performance is fragmentation and allocation of fragments to sites. Allocation and fragmentation can logi- cally be merged and done simultaneously. In this paper we proposed a method that merges vertical fragmentation and allocation. To achieve this goal we applied Bond Energy Algo- rithm 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.

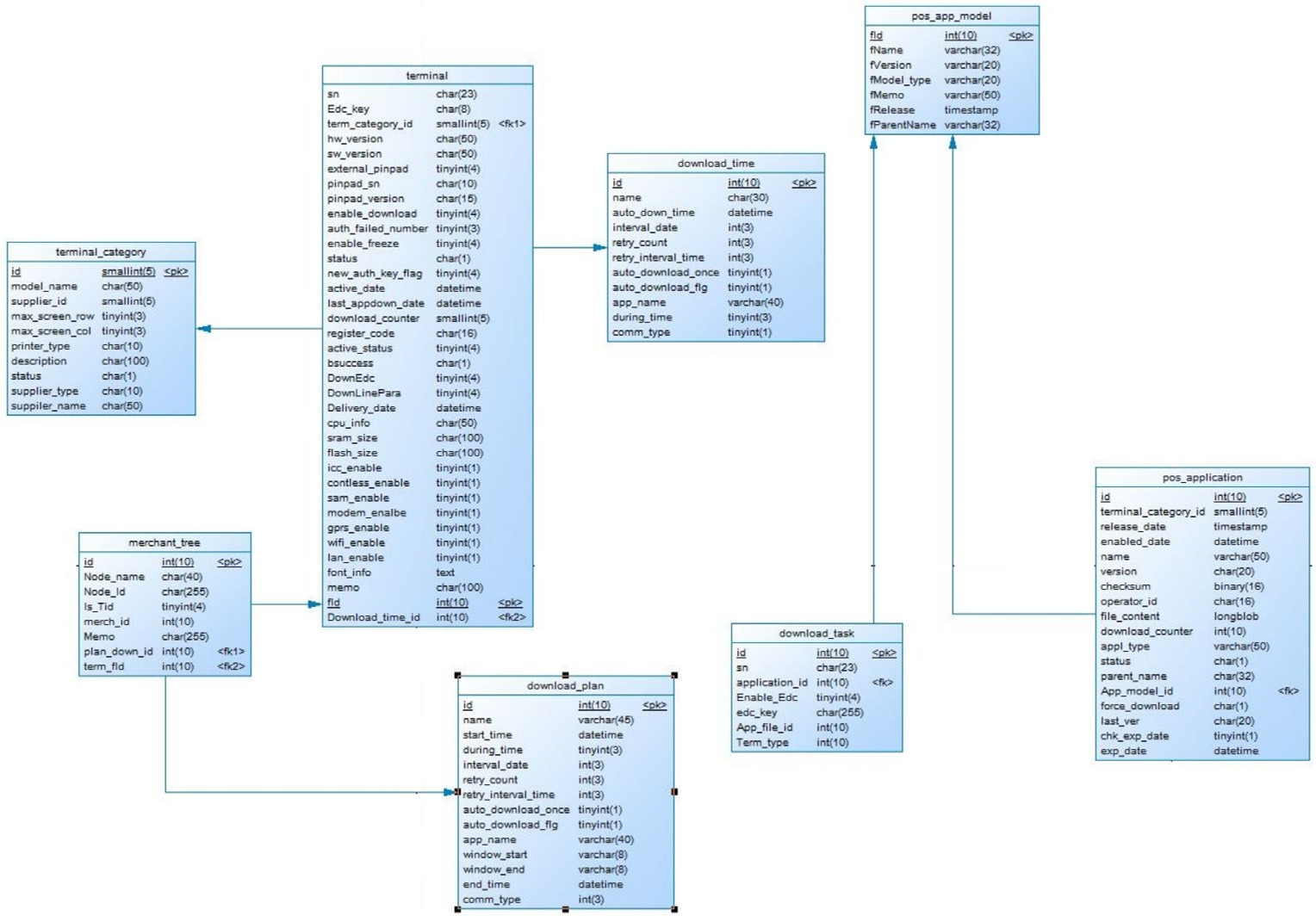


Fig. 4 Table relations in TMS.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Table 2 List of attributes and their related tables. | | |  | Table 3 The Similarity of attributes. | | | | | | |
| No. | List of attributes | Related table |  | Attributes | *n*11 | *n*00 | *n*10 | *n*01 | *coef* | *Sij* |
| 1 | *model* — *name* | *terminal* — *category* |  | *A*2 *and A*1 | 4 | 0 | 2 | 2 | 0 | 1 |
| 2 | *term* — *fid* | *terminal* |  | *A*2 *and A*3 | 4 | 0 | 2 | 2 | 0 | 1 |
| 3 | *hw* — *version* | *terminal* |  | *A*2 *and A*7 | 4 | 0 | 2 | 2 | 0 | 1 |
| 4 | *pinpad* — *version* | *terminal* |  | *A*4 *and A*1 | 4 | 1 | 2 | 1 | 0.35 | 0.93 |
| 5 | *flash* — *size* | *terminal* |  | *A*4 *and A*3 | 4 | 1 | 2 | 1 | 0.35 | 0.93 |
| 6 | *app* — *name* | *download* — *time* |  | *A*4 *and A*7 | 4 | 1 | 2 | 1 | 0.35 | 0.93 |
| 7 | *interval* — *date* | *download* — *time* |  | *A*1 *and A*3 | 4 | 0 | 2 | 2 | 0 | 1 |
| 8 | *start* — *time* | *download* — *plan* |  | *A*1 *and A*7 | 6 | 2 | 0 | 0 | 0 | 1 |
|  |  |  |  | *A*3 *and A*7 | 4 | 0 | 2 | 2 | 0 | 1 |
|  |  |  |  |  |  |  |  |  |  |  |

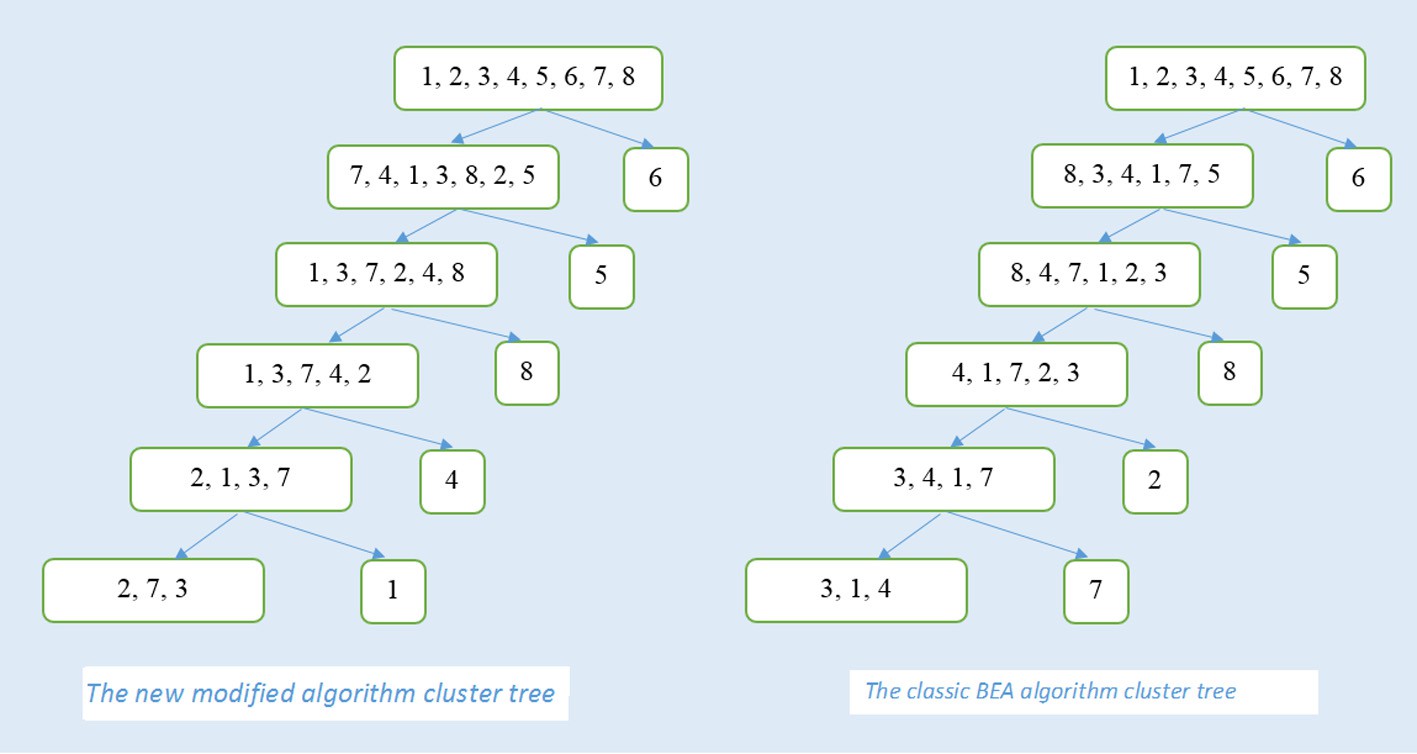


Fig. 5 Hierarchical attribute clustering tree.

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](http://refhub.elsevier.com/S2210-8327(15)00005-8/h0010) [partitioning in object oriented databases, Comput. J. 42 (1)](http://refhub.elsevier.com/S2210-8327(15)00005-8/h0010) [(1999)](http://refhub.elsevier.com/S2210-8327(15)00005-8/h0010).
3. [M. Hammer, B. Niamir, A heuristic approach to attribute](http://refhub.elsevier.com/S2210-8327(15)00005-8/h0015) [partitioning, in: Proceedings ACM SZGMOD International](http://refhub.elsevier.com/S2210-8327(15)00005-8/h0015) [Conference on Management of Data, Boston, Mass.,](http://refhub.elsevier.com/S2210-8327(15)00005-8/h0015) [ACM,](http://refhub.elsevier.com/S2210-8327(15)00005-8/h0015) [New York, 1979](http://refhub.elsevier.com/S2210-8327(15)00005-8/h0015).
4. [Hassan I. Abdalla, A synchronized design technique for efficient](http://refhub.elsevier.com/S2210-8327(15)00005-8/h0020) [data distribution, Comput. Human Behav. 30 (2014) 427–435](http://refhub.elsevier.com/S2210-8327(15)00005-8/h0020).
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](http://refhub.elsevier.com/S2210-8327(15)00005-8/h0035) [of a distributed database, Int. J. Comput. Sci. Netw. Sec. 8 (1)](http://refhub.elsevier.com/S2210-8327(15)00005-8/h0035) [(2008)](http://refhub.elsevier.com/S2210-8327(15)00005-8/h0035).
8. [F. Baiao, M. Mattoso, G. Zaverucha, A distribution design](http://refhub.elsevier.com/S2210-8327(15)00005-8/h0040) [methodology for object DBMS, Distrib. Parallel Databases 16](http://refhub.elsevier.com/S2210-8327(15)00005-8/h0040)

[(1) (2004) 4590](http://refhub.elsevier.com/S2210-8327(15)00005-8/h0040).

1. Haroun Rababaah, H. Hakimzadeh, Distributed Databases: Fundamentals and research, Advanced Database B561, Spring 2005.
2. N. Iacob, Data replication in distributed environments, Annals of the Constantin Brncusi, University of Trgu Jiu, Economy Series, Issue 4, 2010.
3. [J. Gower, A general coefficient of similarity and some of its](http://refhub.elsevier.com/S2210-8327(15)00005-8/h0055) [properties, Biometrics 27 (1971) 857872](http://refhub.elsevier.com/S2210-8327(15)00005-8/h0055).
4. [J.O. Hauglid, N.H. Ryeng, DYFRAM: dynamic fragmentation](http://refhub.elsevier.com/S2210-8327(15)00005-8/h0060) [and replica management in distributed database systems,](http://refhub.elsevier.com/S2210-8327(15)00005-8/h0060) [Distrib. Parallel Databases 28 (2010) 157185](http://refhub.elsevier.com/S2210-8327(15)00005-8/h0060).
5. [J.A. Hoffer, An integer programming formulation of computer](http://refhub.elsevier.com/S2210-8327(15)00005-8/h0065) [database design problems, Znf. Sci. 11 (July 1976) 29–48](http://refhub.elsevier.com/S2210-8327(15)00005-8/h0065).
6. 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.
7. [Jin Hyun Son, Myoung Ho Kim, An adaptable vertical](http://refhub.elsevier.com/S2210-8327(15)00005-8/h0075) [partitioning method in distributed systems, J. Syst. Softw. 73](http://refhub.elsevier.com/S2210-8327(15)00005-8/h0075) [(2004) 551–561](http://refhub.elsevier.com/S2210-8327(15)00005-8/h0075).
8. [S. Mahmoud, J.S. Roirdon, Optimal allocation of resources in](http://refhub.elsevier.com/S2210-8327(15)00005-8/h0080) [distributed information networks, ACM Trans. Database Syst.](http://refhub.elsevier.com/S2210-8327(15)00005-8/h0080) [1 (1976) 1](http://refhub.elsevier.com/S2210-8327(15)00005-8/h0080).
9. [M.T. Ozsu, P. Valduriez, Principles of Distributed Database](http://refhub.elsevier.com/S2210-8327(15)00005-8/h0090) [Systems,](http://refhub.elsevier.com/S2210-8327(15)00005-8/h0090) [Alan Apt, New Jersey, 1999](http://refhub.elsevier.com/S2210-8327(15)00005-8/h0090).
10. [S. Navathe, S. Ceri, G. Wiederhold, J. Dou, Vertical](http://refhub.elsevier.com/S2210-8327(15)00005-8/h0095) [partitioning algorithms for database design, ACM Trans.](http://refhub.elsevier.com/S2210-8327(15)00005-8/h0095) [Database Syst. 9 (4) (1984) 680710](http://refhub.elsevier.com/S2210-8327(15)00005-8/h0095).
11. [S.K. Rahimi, F.S. Haug, Distributed Database Management](http://refhub.elsevier.com/S2210-8327(15)00005-8/h0100) [Systems,](http://refhub.elsevier.com/S2210-8327(15)00005-8/h0100) [A John Wiley and Sons Inc. Publication, IEEE](http://refhub.elsevier.com/S2210-8327(15)00005-8/h0100) [Computer Society, 2010](http://refhub.elsevier.com/S2210-8327(15)00005-8/h0100).
12. [Rui Xu, Donald Wunsch II, Survey of clustering algorithms,](http://refhub.elsevier.com/S2210-8327(15)00005-8/h0110) [IEEE Trans. Neural Netw. 16 (3) (2005)](http://refhub.elsevier.com/S2210-8327(15)00005-8/h0110).
13. [M. Schkolnick, A clustering algorithm for hierarchical structure,](http://refhub.elsevier.com/S2210-8327(15)00005-8/h0115) [ACM Trans. Database Syst. 2 (1977) 1](http://refhub.elsevier.com/S2210-8327(15)00005-8/h0115).
14. Pang-Ning Tan, Michael Steinbach, Vipin Kumar, Introduction to Data Mining, 2005, ISBN: 0-321-32136-7.
15. [Shamkant Navathe, Stefano Ceri, Gio Wiederhold, Jinglie Dou,](http://refhub.elsevier.com/S2210-8327(15)00005-8/h0125) [Vertical partitioning algorithms for database design, ACM](http://refhub.elsevier.com/S2210-8327(15)00005-8/h0125) [Trans. Database Syst. 9 (4) (December 1984)](http://refhub.elsevier.com/S2210-8327(15)00005-8/h0125).
16. K. Wagstaff, S. Rogers, S. Schroedl, Constrained -means clustering with background knowledge, in: Proc. 8th Int. Conf. Machine Learning, 2001, pp. 577–584.
17. [J. Whitten, L. Bentley, K. Dittman, Systems Analysis and](http://refhub.elsevier.com/S2210-8327(15)00005-8/h0135) [Design Methods, sixth ed., McGraw-Hill, 2004](http://refhub.elsevier.com/S2210-8327(15)00005-8/h0135).