A New Technique For Multiple Vertical Fragmentations Of Datasets Using Affinity Matrix In Distributed Environment Using Matlab

Sunil Kumar
Research Scholar – Himalayan University, Itanagar, Arunachal Pradesh (India)

Prof.(Dr.) Neelendra Badal
Dept. of Computer Science & Engineering, KNIET Sultanpur UP

Abstract—Distribution design involves making decisions on the fragmentation and allocation of data across the sites of a computer network. Multiple vertical splitting is the process of subdividing the attributes of a relation to generate fragments. In this paper, we propose an analysis for multiple vertical splitting algorithms using multiple vertical fragmentation of datasets using affinity matrix in distributed environment. This approach starts from the attribute affinity matrix and generates initial clusters based on the affinity values between attributes. Then, it uses the database according to optimal splitting solution to produce final groups that will represent the fragments. Then we analyzed these fragments according to their contribution level. The result is generated that shows how to find optimal solutions

In multiple vertical fragmentation of datasets using affinity matrix in distributed environment

Index Terms—*BEA, Multiple Vertical Fragmentation, Clustered Affinity Matrix, Attribute Usage Matrix, Frequency Matrix, Distributed Database System, Slop Based Partitioning Algorithm.*

I. INTRODUCTION

In distributed computing environments, each unit of data (item) which is accessed at the station, (site) is not usually a relationship but part of the relationship. Therefore, to optimize the performance of the query, the relations of global schema are fragmented into items.

The primary concern of distributed database systems is to design the fragmentation and allocation of the underlying database. The distribution design involves making decisions on the fragmentation and placement of data across the sites of a computer network. The first phase of the distribution design in a top-down approach is the fragmentation phase, which is the process of clustering into fragments the information accessed simultaneously by applications. The fragmentation phase is then followed by the allocation phase, which handles the physical storage of the generated fragments among the nodes of a computer network, and the replication of fragments.

A Distributed database is a database that is under the control of a central database management system (DBMS) in which storage devices are not all attached to a common CPU. It may be stored in multiple computer located in the same physical location, or may be dispersed over a network of interconnected computers. There are multiple sites (computers) in a distributed database so if one site fails then system will not be useless, because other sites can do their job because same copy of data is installed on every location

II. FRAGMENTATION

Fragmentation is a design technique to divide a single relation or class of a database into two or more Partitions such that the Combination of the partitions provides the original database without any loss of information .This reduces the amount of irrelevant data accessed by the applications of the database, thus reducing the number of disk accesses.

Further a new algorithm is being described for fragmentation of clusters of attributes of database table. Vertical partitioning is used during design of a database to enhance the performance of query execution. In order to obtain improved performance, fragments must closely match the requirements of the query workload. Vertical partitioning involves splitting the relations along the columns into partitions all having equal number of rows. Each partition now acts as a separate relation but we preserve the row ordering in all the partitions as it was in the original relation. It should be noted that each partition may contain more than one column. However, when columns in multiple partitions are accessed instead of join we just need to do pasting of columns. The advantages of vertical partitioning are as follows: If query involves only few columns then we avoid unnecessary fetching of other columns. This saves the I/O bandwidth and avoids unnecessary processing. Moreover data in a column belongs to the same domain e.g., values in salary column will be numeric within some range. This similarity in data can be well exploited by compression algorithms and better compression ratios can be achieved.

MULTIPLE VERTICAL FRAGMENTATIONS

Multiple Vertical fragmentation is the collective decay properties of the relational schema R into the sub schema R1, R2,..,Rm, such that each attribute in these sub schemas is often accessed together.

To show how often the same queries together, Hoffer and Severance introduced the concept attribute affinity [11].

If Q = {q1, q2, ..,qm} is a set of applications, R(A1, A2, .., An) is a relational schemas. The relationship between qi and attributes Ajis determined by using the values:

$$use(q_i, A_j) = \begin{cases} 1, & A_j \text{ is engaged in } q_i \\ 0, & A_j \text{ is not engaged in } q_i \end{cases}$$

Put (Ai, Aj) = {qεQ | use(q, Ai). use(q, Aj) = 1}. Attribute affinity between Ai and Ajis:

$$Aff(A_i, A_j) = \sum_{q \in Q(A_i, A_j) \forall S_l} \left(\sum ref_l(q) * acc_l(q) \right)$$

In particular, refl (q): the number of pairs of attributes (Ai,Aj) is referenced in the application q at station Sl; accl(q): frequency of access to applications q in station Sl. BEA algorithm consists of two main phases:

1) Permutations row, column affinity matrix of attribute to obtain the cluster affinity matrix (CA) which has global affinity measure AM (global affinity measure) [1] is the largest.

2) Find the partition of the set of attributes from the matrix CA by exhaustive method, so that:

Z= CTQ *CBQ – COQ2 is the maxima, with:

$$CTQ = \sum_{q \in TQ \, \forall S_j} \sum ref_i(q_j) acc_j(q_i))$$

$$COQ = \sum_{q \in OQ \, \forall S_j} \sum ref_j(q_j) acc_j(q_i))$$

	A_1	A_2	..	A_i	A_{i+1}	..	A_n
A_1							
..			TA				
A_i							
A_{i+1}							
						BA	
A_n							

In which,

$$AQ(q_i) = \{A_j | use(q_i, A_i) = 1\};$$

$$TQ = \{q_i | AQ(q_i) \subseteq TA\};$$

$$BQ = \{q_i | AQ(q_i) \subseteq BA\};$$

$$OQ = Q \backslash \{TQ \cup BQ\}$$

The complexity of the algorithm is proportional to n2.

III. FRAGMENTATION PROCEDURE

In this section SBPA, used for Vertical Partitioning of relation, is discussed in detail. Firstly using the AUM and FM, Clustered Affinity Matrix (CAM) is calculated. After calculation of CAM, SBPA is used to make the fragments of the relation.

A. ATTRIBUTE USAGE MATRIX

The Attribute Usage Matrix is used to show the attributes of relation used by a query. For each query QI and each attribute AJ, an Attribute Usage Value 0 or 1 is associated in AUM. The associated value is 1 if the attribute AJ is used by query QI otherwise the value associated is 0.

$$use(Q_i, A_j) = \begin{cases} 1, & \text{if Attribute } A_j \text{ is used by Query } Q_l \\ 0, & \text{Otherwise} \end{cases}$$

Each row of AUM refers the attributes used by the corresponding query. The "1" entry in a column indicates that the query "uses" the corresponding attribute.

Table 1 is an example of Attribute Usage Matrix in this paper.

Table 1: Attribute Usage Matrix

Query	Attribute			
	A1	A2	A3	A4
Q1	1	0	1	0
Q2	0	1	1	0
Q3	0	1	0	1
Q4	1	0	0	1

B. FREQUENCY MATRIX

The frequency matrix represents the number of time a query is fired from one or more sites. Table2 is an example of the Frequency Matrix used in this paper.

Table 2: Frequency Matrix

Query	Site		
	S1	S2	S3
Q1	10	12	15
Q2	7	0	0
Q3	30	25	20
Q4	5	0	0

C. ATTRIBUTE AFFINITY MATRIX

Attribute affinity value measures the strength of an imaginary bond between the two attributes. It is predicated on the fact that attributes are used together by the query. Attribute affinity value represents the number of times two attributes are accessed together at all sites.

Attribute affinity value between two attributes AI and AJ of a relation R[A1,A2…AN]with respect to the set of queries Q={Q1, Q2…QQ}is defined as follows.

Attribute affinity value between AI and AJ is represented as aff (AI, AJ).

$$Aff(A_I, A_J) = \sum_{All\ Queries\ that\ access\ AI\ and\ Aj} Query\ A$$

Where,

Query access=$\sum_{all\ site} access\ frequency\ of\ a\ query * \frac{access}{execution}$

Query access (Q1) = 37

Query access (Q2) = 7

Query access (Q3) = 75

Query access (Q4) = 5

aff (A1,A3) =ΣQ1 query access=37

In the same way the whole Attribute affinity value is calculated.

Table 3: Attribute Affinity Matrix

Attribute	Attribute			
	A1	**A2**	**A3**	**A4**
A1	42	0	37	5
A2	0	82	7	75
A3	37	7	44	0
A4	5	75	0	8

D. CLUSTERED AFFINITY MATRIX

For the fragmentation of attributes in a relation, firstly attributes must be clustered. Clustering problem is widely researched in databases, data mining and statistics communities [8], [9], [10], [11], [12], [13]. Hoffer and Severance in [2] has suggested that the Bond Energy Algorithm (BEA) should be used for this purpose. The Bond Energy Algorithm takes Attribute Affinity Matrix as input, changes the order of its rows and columns, and produces a Clustered Affinity Matrix (CAM). Bond Energy Algorithm makes the cluster of those attributes which have high Attribute affinity value.

Placement of attributes in CAM

Placement of A1 and A2:

In the initialization step first and second columns of AAM is placed to the first and second column of CAM respectively.

Attribute A1 is placed at position 1 in CAM: [A1]

Attribute A2 is placed at position 2 in CAM: [A1, A2]

Placement of A3:

Contribution of attribute A3 at position 1 in CAM= 6364

Contribution of attribute A3 at position 2 in CAM = 6860

Contribution of attribute A3 at position 3 in CAM = 1764

Attribute A3 is placed at position 2 in CAM: [A1, A3, A2]

Placement of A4:

Contribution of attribute A4 at position 1 in CAM = 1220

Contribution of attribute A4 at position 2 in CAM = - 3724

Contribution of attribute A4 at position 3 in CAM = 23956

Contribution of attribute A4 at position 4 in CAM =24300

Attribute A4 is placed at position 4 in CAM: [A1, A3, A2, A4]

Hence in Clustered Affinity Matrix, the order of placing the attributes in rows and columns are given below:

[A1, A3, A2, A4]

Table 4: Clustered Affinity Matrix

Attribute	Attribute			
	A1	**A3**	**A2**	**A4**
A1	42	37	0	5
A3	37	44	7	0
A2	0	7	82	75
A4	5	0	75	80

IV. BOND ENERGY ALGORITHM

Bond Energy Algorithm (BEA) has been usedfor clustering of entities. BEA finds anordering of entities (in our case attributes)such that the global affinity measure ismaximized.

$$AM = \sum_i \sum_j \left(affinity\ of\ A_i\ and\ A_j\ with\ their\ neighbors\right)$$

- The bond energy algorithm (BEA) was developed and has been used in the database design area to determine how to group data and how to physically place data on a disk.
- It can be used to cluster attributes based on usage and then perform logical or physical design accordingly. With BEA, the

affinity (bond) between database attributes is based on common usage.

- This bond is used by the clustering algorithm as a similarity measure. The actual measure counts the number of times the two attributes are used together in a given time. To find this, all common queries must be identified.
- The idea is that attributes that are used together form a cluster and should be stored together. In a distributed database, each resulting cluster is called a vertical fragment and may be stored at different sites from other fragments.
- The basic steps of this clustering algorithm are:

i. Create an attribute affinity matrix in which each entry indicates the affinity between the two associate attributes. The entries in the similarity matrix are based on the frequency of common usage of attribute pairs.

ii. The BEA then converts this similarity matrix to a BOND matrix in which the entries represent a type of nearest neighbor bonding based on probability of co-access. The BEA algorithm rearranges rows or columns so that similar attributes appear close together in the matrix.

iii. Finally, the designer draws boxes around regions in the matrix with high similarity.

The resulting matrix, modified from, is illustrated in Figure 1. The two shaded boxes represent the attributes that have been grouped together into two clusters.

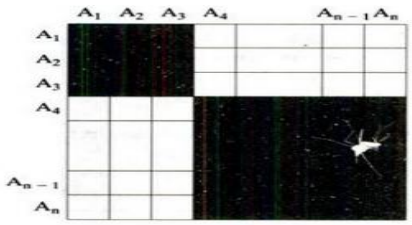

Figure 1: Clustered Affinity Matrix for BEA

Two attributes Ai and Aj have a high affinity if they are frequently used together in database applications. At the heart of the BEA algorithm is the global affinity measure. Suppose that a database schema consists of n attributes {A1, A2, , An}. The global affinity measure, AM, is defined as

$$AM = \sum_{i=1}^{n} (bond(A_i, A_{i-1}) + bond(A_i, A_{i+1}))$$

Vertical fragmentation based on knowledge- oriented clustering technique-

To illustrate the vertical fragmentation algorithm based on knowledge-oriented clustering techniques. We use the assumption of examples about vertical fragmentation problem based on BEA algorithm is presented in [1], [8]:

The set of attributes At= {A1, A2, A3, A4}

The set of transactions Q = {q1, q2, q3, q4}. The matrix used:

	A_1	A_2	A_3	A_4
q_1	1	0	1	0
q_2	0	1	1	0
q_3	0	1	0	1
q_4	0	0	1	1

The frequency of application execution with a set of transactions {q1, q2, q3, q4}, and F = {f1, f2, f3, f4} = {45, 5, 75, 3}.

From the assumption, we have the reference feature vectors:

	q_1	q_2	q_3	q_4
VA_1 =	45	0	0	0
VA_2=	0	5	75	0
VA_3=	45	5	0	3
VA_4=	0	0	75	3

The similar matrix S4x4= (s(Ak, Al)) k=1,4; l=1,4

	A_1	A_2	A_3	A_4
A_1	1	0	0.9918	0
A_2		1	0.0073	0.9970
A_3			1	0.0026
A_4				1

The result of the vertical fragmentation algorithm based on the clustering algorithm towards knowledge-oriented.

Cluster	**Set of attribues**
1	$\{A_1, A_3\}$
2	$\{A_2, A_4\}$

This fragmentation results correlate with the results of the multiple vertical fragmentation by algorithm BEA.

V. PRAPOSED SBP ALGORITHAM

The objective of Slop Based Partitioning Algorithm is to find a set of attributes that are frequently accessed by distinct set of queries. Using the Slop Based Partitioning Algorithm, the user makes the fragments of a relation on the basis CAM, which is calculated by AUM and FM. The first row of CAM is taken for fragmenting the clusters from a relation. The point between the neighbors attributes of the CAM is considered as Split-point

if slop diminishes between the se attributes very rapidly .The pseudo code for the SBPA is given below:

Algorithm: SBPA

Y [1, 2] =CAM (1, 2)-CAM (1, 1) = 37-42= -5, X [1, 2] = 2

Y [1, 3] =CAM (1, 3)-CAM (1, 2) = 0-37= -37, X [1, 3] =3

Y [1, 4] = CAM (1, 4)-CAM (1, 3) =5-0 =5, X [1, 4] =4

The Plot command in pseudo code plots the following graph. The graph shows the slop value Y [1, i] at point i.

Vertical partition between two fragmentation

- *Comparison:* In the last step the user finds the smallest value of Y [1, i] which represents the rapid diminishing of slop. The index i at which value of Y [1, i] is the smallest the corresponding value of X [1, i] is considered as Split-point. The following Calculation is performed with referenced to CAM.

Y [1, 1] =CAM (1, 1) =45, X [1, 1] =1

Y [1, 2] =CAM (1, 2)-CAM (1, 1) = 0-45= -45, X [1, 2] = 2

Y [1, 3] =CAM (1,3)-CAM (1,2)= 45-0= 45, X [1, 3] =3

Y [1, 4] = CAM (1, 4)-CAM (1, 3) =0-45 =-45, X [1, 4] =4

Y [1, 5] = CAM (1, 5)-CAM (1, 4) =-0-0 =0, X [1, 5] =5

Y [1, 6] = CAM (1, 6)-CAM (1, 5) =45-0 =45, X [1, 6] =6

Y [1, 7] = CAM (1, 7)-CAM (1, 6) =0-45 =-45 X [1, 7] =7

Y [1, 8] = CAM (1, 8)-CAM (1, 7) =0-0 =0, X [1, 8] =8

Y [1, 9] = CAM (1, 9)-CAM (1, 8) =0-0 =0, X [1, 9] =9

Y [1, 10] = CAM (1, 10)-CAM (1, 8) =0-0 =0, X [1,10] =10

The Plot command in pseudo code plots the following graph. The graph shows the slop value Y [1, i] at point i.

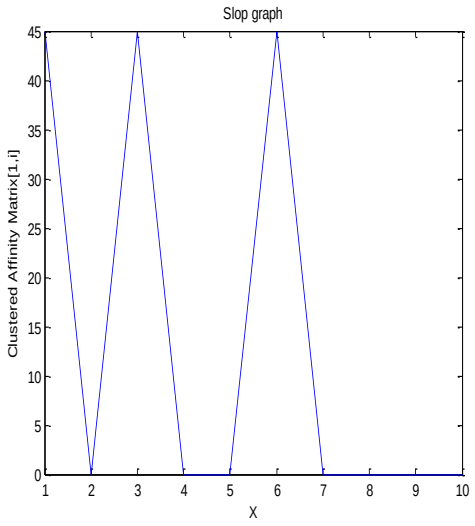

Figure2: Multiple vertical Fragmentations

Graph between CAM [1, i] and X[1,i]

The above graph shows the slop diminishes very rapidly between X [1, i] =2 and X [1, i] =3. So the Split-point is recorded between second and third attribute of CAM. So the above Clustered Affinity Matrix can be divided into two fragments. One fragment contains the attribute {A1, A3} and second fragment contains the attributes {A2, A4}.

So for the fragmentation of relation R [A1, A2, A3, A4] has done as below:

[A1, A3] | [A2, A4]

VI. RESULTS

An experiment has been carried out to test the working of proposed Vertical Partitioning algorithm SBPA. It has been carried out on a system with core i3 processor, 3GB RAM, Matlab toolbox and MS Access database. A relation with name Project has been used for partitioning. The Project relation has been stored in MS Access database as following:

Table 5: Input Project

The Project relation has tested against set of four queries *Q1, Q2, Q3* and *Q4* generated from any of the three sites named *S1, S2* and *S3*.

Q1: Find the Budget from the Project where given its identification number.

(SELECT BUDGET, FROM PROJECT, WHERE PNO=Value)

Q2: Find the Name and Budget of all Projects.

(SELECT PNAME, BUDGET FROM PROJECT, WHERE LOCATION=Value)

Q3: Find the Name of projects located at given city.

(SELECT PNAME, FROM PROJECT, WHERE LOCATION=Value)

Q4: Find the PNo and total project Budget for each city.

(SELECT PNo, SUM (BUDGET), FROM PROJECT, WHERE LOCATION=Value)

The Attribute Usage Matrix of the above queries set is as following.

Table 7: Attribute Usage Matrix Project

Query	Attribute			
	PNo	PNAME	BUDGET	LOCATION
Q1	**1**	**0**	**1**	**0**
Q2	**0**	**1**	**1**	**1**
Q3	**0**	**1**	**0**	**1**
Q4	**0**	**0**	**1**	**1**

The frequency of queries Q1, Q2, Q3and Q4 at three sites has considered as following:

Table 8: Frequency Matrix Project

Query	Site		
	S1	S2	S3
Q1	20	25	10
Q2	5	2	0
Q3	16	18	30
Q4	3	2	1

Using the Bond Energy Algorithm proposed by Hoffer and severance in [2], Clustered Affinity Matrix is calculated from Attribute Usage Matrix Project in Table 7 and Frequency Matrix Project in Table 8.

After calculating the Clustered Affinity Matrix, relation project in Table 6 has been partitioned into two fragments using the SBPA. One fragment named as Project1 has attributes PName and Location while other fragment named as Project2 has attributes PNo and Budget as followed. So the relation project having four attributes *PNo*, *PName*, *Budget*, *Location* is partitioned into two fragments for the above given Attribute Usage Matrix in Table 7 and Frequency Matrix in Table 8 respectively.

Table 10: Project 1

PName	Location
Instrumentation	Montreal
Database Develop	New York
CAM/CAD	New York
Maintenance	Paris

Table 11: Project 2

PNo	Budget
P1	150000
P2	135000
P3	250000
P4	310000

VII. CONCLUSION

Distributed databases have many aspects and every organization has certain preferences. For the telecom sector, the response time is prioritized.

Our experiment showed that the how to calculate the fragments for different criteria used. In the distributed database, data is fragmented. These fragments are short compared to the full database (centralized database contains maximum columns). However, when we need data from multiple sites for a query (report queries), the response time is increased. Accessing data from multiple remote sites and then joining those takes long time. But in the centralized database since data is at one place so, it is easy and fast to search it.

Multiple Vertical Partitioning algorithms SBPA has presented and successfully implemented for improving the Query-Response-Time in Distributed Database System. The proposed algorithm SBPA has used CAM. In the first phase, CAM is calculated from AUM and FM. In the second phase, the multiple fragments of the relation are created by CAM Using SBPA.

REFERENCES

[1] Tamer O., ValduriezP (1999), Principles of Distributed Database Systems. Prentice Hall Englewood Cliffs, Second Edition, New Jersey 07362.

[2] Hoffer, J.A. and Severance, D.J. 1975.The use of cluster analysis in physical database design. In Proceedings of the 1st International Conference on Very Large Data Bases, New York, USA.

[3] Horowitz, E. and Sahni, S. 1978. Fundamentals of Computer Algorithms. Computer Science Press Rockville, Maryland.

[4] McCormick, W. T. Schweitzer P.J., and White T.W., "Problem Decomposition and Data Reorganization by A Clustering Technique," Operation Research, Vol. 20 No. 5, September 1972, 993-1009.

[5] Navathe, S., Ceri, S., Wierhold, G. and Dou, J., "Vertical Partitioning Algorithms for Database Design," ACM Transactions on Database Systems, Vol. 9 No. 4, December 1984, 680-710.

[6] Navathe, S. and Ra M., "Vertical Partitioning for Database Design: A Graph Algorithm," ACM Special Interest Group on Mamagement of Data (SIGMOD) International Conference on Management of Data, Vol. 18 No. 2, June 1989, 440-450.

[7] Chu, W. W. and Ieong, I. "A Transaction-Based Approach to Vertical Partitioning for Relational Database Systems," IEEE Transactions on Software Engineering, Vol. 19 No. 8, August 1993, 408-412.

[8] Bradley, P. S., Fayyad, U. M. and Reina, C., "Scaling Clustering Algorithms to Large Databases", in proceedings of the 4th International Conference on Knowledge Discovery & Data Mining , June 1998, 9-15.

[9] Guha, S., Rastogi, R. and Shim, K., "CURE: an efficient clustering algorithm for large databases", in proceedings of the 1998 ACM SIGMOD international conference on Management of data, Vol. 27, Issue 2, June 1998, 73-84.

[10] Ng, R. T. and. Han, J., "Efficient and Effective Clustering Methods for Spatial Data Mining", Proceedings of the 20th International Conference on Very Large Data Bases, September 1994, 144-155.

[11] Jain, A. and Dubes, R., "Algorithms for Clustering Data", Prentice Hall, New Jersey, 1998.

[12] Kaufman, L., Rousseuw, P., "Finding Groups in Data- An Introduction to Cluster Analysis", Wiley Series in Probability and Math. Sciences, 1990.

[13] Zhang, T., Ramakrishnan, R. and Livny, M., "An Efficient Data Clustering Method for Very Large Databases", in proceedings of the SIGMOD international conference on Management of data, June 1996, 103-114.

ABOUT THE AUTHOR

Sunil Kumar Verma Asst. Professor (Sr. Scale) Department Of Computer Science & Engineering, Feroze Gandhi Institute of Engineering & Technology Raebareli U.P. Sunil Kumar Verma is a Ph.D Scholar Department of Computer Science & Engineering at Himalayan University Itanagar Arunachal Pradesh(India. He has received Masters degree M.Tech (Software System) , Department Of Computer Science & Engineering from Birala Institute of Technology & Science (BITS) PILANI Rajasthan India and he has received his Bachelor of Technology degree from Bundelkhand Institute of Technology (BIET), Jhansi in Computer Science & Engineering Pradesh(India) ,

Dr. Neelendra Badal is an Associate Professor in the Department of Computer Science & Engineering at Kamla Nehru Institute of Technology (KNIT), Sultanpur (U.P), India. He received B.E. (1997) from Bundelkhand Institute of

Technology (BIET), Jhansi in Computer Science & Engineering, M.E. (2001) in Communication, Control and Networking from Madhav Institute of Technology and Science (MITS), Gwalior and PhD (2009) in Computer Science & Engineering from Motilal Nehru National Institute of Technology (MNNIT), Allahabad. He is Chartered Engineer (CE) from Institution of Engineers (IE), India. He is a Life Member of IE, IETE, ISTE and CSI-India. He has published about 60 papers in International/National Journals, conferences and seminars. His research interests are Distributed System, Parallel Processing, GIS, Data Warehouse & Data mining, Software engineering and Networking.