.

A Clustering Based Vertical Fragmentation and Allocation of a Distributed Database

Abhishesh Dahal
Department of Graduate Studies
Nepal College of Information Technology
(Pokhara University)
Lalitpur, Nepal
d.abhishesh@gmail.com

Shashidhar Ram Joshi
Department of Electronics and Computer Engineering
Institute of Engineering
(Tribhuvan University)
Lalitpur, Nepal
srjoshi@ioe.edu.np

***Abstract*—Distributed Database Management System (DDBMS) is often surrounded with an issue of identifying the best design strategy. Efficiency of this design strategy to a great extent rely upon fragmentation of a relation. It additionally relies upon allocation of the fragment pieces to some sites within the network. Fragmentation and allocation are considered independent, even though they utilize similar information to accomplish a common target. Among the existing fragmentation strategies, vertical fragmentation is regularly viewed as an entangled fragmentation strategies than other on the grounds that the immense number of choices makes it about difficult to get an ideal answer for the vertical fragmentation issue. Allocation of a fragment is a NP-hard problem as it is complex to search an optimal site for holding the fragment. In this manner, we can just hope to discover a heuristic arrangement of solution for all these activities. The main purpose of this research is to present a clustering based fragmentation technique in which table attributes with similarity reside in same fragment cluster. As per the frequency of user query, affinity matrix is generated. This affinity matrix is further utilized to generate affinity cluster. Based on this affinity cluster, attributes having least Euclidian distance between them are considered as similar attributes and vertical fragments are obtained. After the fragments are generated those fragments are allocated to certain desired sites based on allocation table. This approach aims to solve the issue and complexity that occurs in previous vertical fragmentation approach.**

Keywords—Distributed Database, Fragmentation, Allocation, User Query, K-Means Clustering

I. Introduction

Distributed Database is a type of distributed architecture of a database for collection of data instances which are physically scattered over various locations but are logically connected with each other via a computer network [1]. These data instances are handled by their respective sites which coordinates and communicates for efficient data placement and retrieval. These sites are further managed by a single distributed database management system (DDBMS). Every site connected in the network has autonomous processing capacity.

The main issue of DDBMS is its efficient design strategy. The fundamental complexity confronting DDBMS design are: identifying the fragmentation strategy to dissect the relation of a database and also identifying the site to allocate the pieces of fragments so that they are efficiently accessed without any huge cost overhead [2].

Fragmentation is a design method for separating a solitary relation of a database into at least multiple segments to an extent that the mix of those segments generates initial relation without any elimination of the data [1]. This diminishes the measure of insignificant information gotten to by the utilization of the database, in this manner lessening the quantity of storage access. Altogether there are three types of fragmentation strategy: Vertical Fragmentation (VF), Horizontal Fragmentation (HF) and Mixed Fragmentation (MF). VF is dissection on the basis of attributes, HF is dissection on the basis of records and MF is combination of VF and HF.

Fragmentation targets to enhance performance, reliability, balanced storage capacity and communication costs. To make decision on division of a relation into multiple fragment, quantitative and qualitative information are required. Quantitative information contains parameters like selectivity of the queries, access site for query execution and availability of various user query, etc. Qualitative information contains parameters like access data type, read/write information, etc.

Validation of the obtained fragment is governed by correctness rules of fragmentation, i.e. completeness rule, reconstruction rule and disjointness rule. Completeness rule state that the relation must be decomposed in such a way that there must not be any data loss. Reconstruction rule state that the initial relation must be regenerated from the decomposed relation without any alteration on the initial data. Disjointness rule state that any data after decomposition must be found only on a single fragment.

Fragment allocation is the way toward relegating the parts of a database to destination sites. At that point of data allocation, there may be presence of sole copy or a duplicate copy. Allocation generally boost the presence of sole copy. The fragment allocation improves the performance of application execution. While allocating fragment, attributes are grouped and each fragment group are put away at site with optimal distribution. Proper allocation ensures better operational handling and reduced communication cost. A concept similar to allocation is replication where a duplicate copy of the fragment are created and placed to desired destination site.

The main contribution of this research work includes the utilization of clustering based fragmentation technique where the table attributes with similarity reside in a same fragment cluster. This proposed method utilize the information regarding the frequency of user query to generate an attribute affinity matrix. Based on this matrix, affinity cluster are generated using K-Means clustering algorithm. As per the affinity cluster, attributes having least Euclidian distance between them are considered as similar attributes and vertical fragments are obtained. After the fragments are generated those fragments are allocated to certain desired sites. Finding of the desired site is also on the basis of the frequency of user query. The more the user query triggered form a site, more the chances of allocating the fragment on to that site.

The rest of the paper is organized as follows: Section II present the task done in past which are similar to this research task. Section III and its sub-section covers the research methodology used in this research. Findings of the experiments are discussed in section IV. Section V presents conclusion of the research.

II. RELATED WORK

Efficiency of database design strategy to a great extent rely upon fragmentation and allocation of a relation. Identifying better fragments and optimal site for its placement is major issue here. For this, many authors proposed variety of methods in context of distributed database design.

The authors of [3] propose a plan to achieve vertical fragmentation which is an extension of idea as compared to the plan presented at [4] for horizontal fragmentation. This plan permits vertical fragmentation, allocation and replication choices to be taken at the primary phase of planning an appropriated database. The proposed plan utilize Prim's algorithm for generating Minimum Spanning Tree (MST) to segments the distributed database relations vertically. Also, the specific fragment are allocated to the site with maximum manipulation value for that piece of fragment than different destination sites.

The author of [5] propose Bond Energy Algorithm (BEA) where the attributes of a table are grouped with each other on the basis of attribute affinity matrix however this method possess two major downsides. The main downside of this algorithm was consideration of non-overlapping n-ary partitioning of the relation where the complexity of such a calculation would be O (2^n). This drawback occurred in BEA due to n-ary partitioning can easily be handled by iteratively applying binary partitioning techniques. In any case, the real shortcoming of binary partitioning iteratively give rise to more of the complex issues at every stage and therefore, its multifaceted nature will be expanded. The secondary downside occurred in the same algorithm is the requirement to move tuples and attributes of attribute affinity matrix in every steps. Hence, the computational complexity of this methodology is O (n^2log (n)), where n indicates quantity of attributes.

The next method proposed by authors of [6] is based on graphical technique for generating vertical fragment. First phase start from generation of attribute affinity matrix which is later converted into a complete graph termed as affinity graph. With the help of this affinity graph, a linear connected tree is formed. At last, it splits the relation by thinking about a cycle of the tree as a fragment.

The study of [7] suggest another way to deal with vertical fragmentation of a distributed database relation. This is also a graphical way for generating vertical fragments similar to that as mentioned in [6]. This way also takes the affinity graph as input information. With this input, it develops a lot of cluster groups on the affinity graph. At long last, groups with most "affinity index" value are iteratively picked as fragments.

The creators of [8] likewise begin from the attribute affinity matrix. They create starting gatherings dependent on the attribute affinity matrix and union the underlying created gatherings to deliver last gatherings that present the fragments.

All the referenced strategies in [5-8] utilize the trait attribute affinity matrix as a contribution to the fragment creation algorithm. But, the main thing that plays role in generation of this attribute affinity matrix is the presence of the amount of user query which is not present at the primary phase of the database design.

Authors in [9] states a plan for vertically splitting the relation of a database at the structure design stage and the suggested plan decides to hit the proportion of produced splits. On the off chance that the hit proportion falls beneath a predetermined threshold, at that point the produced fragments are modified. It solves the issue of attribute affinity matrix but in addition it requires numerous improvements to the outcomes to get the past aftereffects of [5].

The creators of [10] propose a vertical fragmentation procedure which depends on Apriori algorithmic procedure. The proposed calculation utilizes the attribute usage matrix rather than the use of attribute affinity matrix. The creators of [11] propose an artificial intelligence based genetic algorithm employing several crossovers and mutations in order to create vertical fragments of a distributed database tables considering access path selection.

III. PROPOSED METHOD

The proposed method targets to obtain a vertical fragment out of the relation of a database. It also aims to allocate those fragments to certain desired sites as per user query. This method utilize the list of user query as its primary input. Based on this input, attribute affinity matrix is constructed. This attribute affinity matrix is utilized to create affinity cluster matrix which is considered as an input to the clustering algorithm which further utilize K-Means clustering to split the relation into multiple fragments clusters. Further each obtained fragment are allocated to the particular site based on the maximum access carried out by that site. This maximum access information along with fragment attribute is recorded by allocation table which is evaluated on the basis of same user query used and list of fragments. Here, the alternative name used for attribute affinity matrix is Site-Attribute Usage Matrix, name used for affinity cluster matrix is Attribute-Attribute Usage Matrix and the name of table used for holding allocation information is Allocation Table.

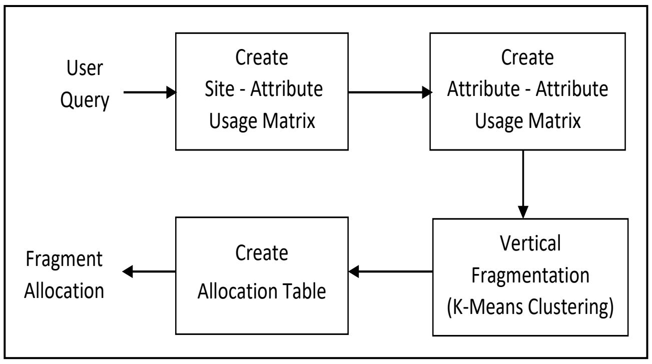
User Query
Create Site - Attribute Usage Matrix
Create Attribute - Attribute Usage Matrix
Vertical Fragmentation (K-Means Clustering)
Create Allocation Table
Fragment Allocation

FIGURE 1. FRAGMENTATION AND ALLOCATION

A. User Query

It is the list of query performed by different user placed at different sites. This list includes information regarding the attribute accessed by each query and also defines in which site the query was invoked. It also indicates the type of query operation performed to particular attribute.

TABLE I. USER QUERY LIST

Site	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10
S9	d r	d	d	d	d	d	d	d	d	d
S7			r			r		r		r
S3		r				r				
S10	c	c								
S5				r r	r		r	r	r	
S6				r	r		r			
S2	r		r							r
S4	c						c	c		
S1	c							c		
S8	d	d	d	d	d	d	d	d	d r	d

The value c, r, d in Table I indicates create, read and delete operations respectively.

B. Site-Attribute Usage Matrix

This matrix is n*m matrix where 'n' represent the number of sites and 'm' represent the number of attributes. The site - attribute usage matrix is constructed from the user query list using formula:

$$SAUM(S_i, A_j) = \text{number of user query performed on site } S_j \text{ to use attribute } A_j \tag{1}$$

The value in Table II is obtained from Table I using the formula mentioned just above.

TABLE II. SITE-ATTRIBUTE USAGE MATRIX

Attr. / Site	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10
S1	1	0	0	0	0	0	0	1	0	0
S2	1	0	1	0	0	0	0	0	0	1
S3	0	1	0	0	0	1	0	0	0	0
S4	1	0	0	0	0	0	1	1	0	0
S5	0	0	0	1	1	0	1	1	1	0
S6	0	0	0	1	1	0	1	0	0	0
S7	0	0	1	0	0	1	0	1	0	1
S8	1	1	1	1	1	1	1	1	1	1
S9	1	1	1	1	1	1	1	1	1	1
S10	1	1	0	0	0	0	0	0	0	0

The value in Table II indicates the number of query accessing the specified attributes from the site indicated. For example, the value of attribute 1 are A1(1,1,0,1,0,0,0,1,1,1) which indicates that A1 is accessed from Site 1, Site 2, Site 4, Site 8, Site 9 and Site 10 by one query each and there is no any access to attribute 1 from Site 3, Site 5, Site 6 and Site 7.

C. Attribute-Attribute Usage Matrix

The attribute-attribute usage matrix indicates the affinity relationship between the attributes. This is n*n square matrix where the value n indicates number of attributes. It is constructed from the site attribute usage matrix using formula:

$$AAUM(A_i, A_j) = \sum_{k=1}^{\text{Number Of Sites}} SAUM(k, A_i) * SAUM(k, A_j) \tag{2}$$

The value in Table III is obtained from Table II using the formula mentioned just above.

TABLE III. ATTRIBUTE-ATTRIBUTE USAGE MATRIX

Attr. / Attr.	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10
A1	6	3	3	2	2	2	3	4	2	3
A2	3	4	2	2	2	3	2	2	2	2
A3	3	2	4	2	2	3	2	3	2	4
A4	2	2	2	4	4	2	4	3	3	2
A5	2	2	2	4	4	2	4	3	3	2
A6	2	3	3	2	2	4	2	3	2	3
A7	3	2	2	4	4	2	5	4	3	2
A8	4	2	3	3	3	3	4	6	3	3
A9	2	2	2	3	3	2	3	3	3	2
A10	3	2	4	2	2	3	2	3	2	4

The value in Table III indicates the affinity value between the attributes of relation. For example, the value in second column shows the affinity relationship of attribute A1 with all other attributes. The value AAUM(A1,A2) is obtained by multiplying SAUM(S1,A1) and SAUM(S1,A2) for every site like S1 and adding all value from each site i.e. the value in the cell AAUM(A1,A2) = { 1*0 + 1*0 + 0*1 + 1*0 + 0*0 + 0*0 + 0*0 + 1*1 + 1*1 + 1*1 = 3 }.

D. Vertical Fragmentation

Vertical fragmentation is on the basis of K-Means Clustering algorithm. Total number of cluster for grouping attributes can be predefined during the design time or can be retrieved as per user requirement. The clustering algorithm is proceeded on the basis of Euclidean Distance similarity between the attributes based on the value of attribute - attribute similarity matrix. Euclidean Distance formula is given as:

$$d(p,q) = d(q,p) = \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2} \tag{3}$$

where,
p and q represent n dimensional item points in the cluster

Initially the clustering algorithm starts with all attributes of Table III as the data points. After that specified number of points randomly are assumed as cluster centers. Then Euclidean distance between each data point and cluster centers are computed. Among all the cluster, the data point is assigned to that cluster for which the distance between data point and cluster center is minimum. Again the newly formed cluster is initialized with new center point by taking average value of those newly assigned data points to the cluster. This process is repeated until the cluster hold same data points for consecutive rounds.

E. Fragment Allocation

Allocation of fragment refers to placement of fragment to some specific site so that access to the particular attribute of that fragment spends least cost as possible. Allocation of the fragment is totally access based i.e. site with more access value to the attributes of a fragment gets chance to hold the entire fragment in it.

A table named as allocation table is constructed which holds the information of the fragment with its attributes and the amount of access performed by each site to that fragment. The site with maximum access value in that table is allocated with that fragment.

IV. THE EXPERIMENTAL RESULTS

Data in query list is obtained as per the randomized behavior of user. A custom table of relational database with 10 attributes (A1, A2, A3, A4, A5, A6, A7, A8, A9 and A10) is taken in account to achieve our task. This table is fragmented into two fragment cluster (F1 and F2) based on K-Means clustering algorithm.

Initially, user query is obtained which is represented by Table I. Based on this user query Site Attribute Usage Matrix is constructed. This matrix is represented by Table II. With the help of Site Attribute Usage Matrix, Table III is constructed which represents Attribute-Attribute Usage Matrix. Again on the basis of attribute-attribute usage matrix, clustering algorithm is applied and vertical fragment of the table is generated. The fragment clusters are represented in Figure 2.

In Figure 2, there exist two fragment clusters namely F1 and F2. F1 indicates one fragment with 6 attributes i.e. F1(A1, A4, A5, A7, A8, A9) and F2 indicates another fragment with 4 attributes i.e. F2(A2, A3, A6, A10).

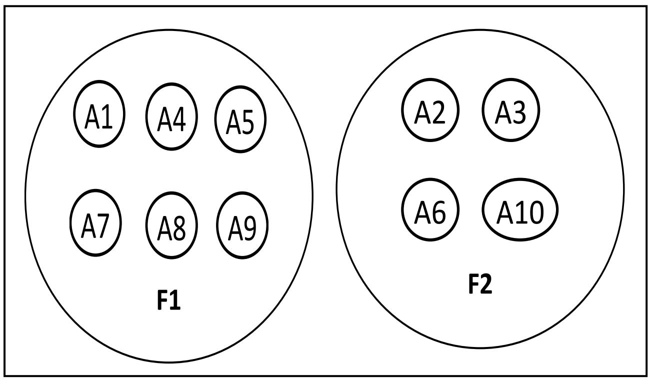

FIGURE 2. FRAGMENT CLUSTER

Again, each fragment is to be placed on some desired site so based on same user query and fragment information, allocation table is constructed. Table IV represent the fragment allocation table. The values in column represent the access value of the fragment by specified sites in particular row. The row with highest value means the site is accessing that fragment attributes frequently and the fragment need to be placed at that site.

TABLE IV. ALLOCATION TABLE

Site	Fragment 1	Fragment 2
	(A1, A4, A5, A7, A8, A9)	(A2, A3, A6, A10)
S1	2	0
S2	1	2
S3	0	2
S4	3	0
S5	5	0
S6	3	0
S7	1	3
S8	6	4
S9	6	4
S10	1	1

The value in Table IV indicates the information regarding the allocation of fragments. The value in column of Fragment 1 is obtained by evaluating the access of attributes in fragment 1 by the respective site. Similar methods holds for fragment 2 as well. For example, relating Table I and Table IV, it is observed that site S1 access only two attributes A1 and A8 among six attributes of fragment 1. Also only one query from Site 1 access those two attributes. Hence, the total access by Site 1 to all attributes of fragment 1 is 2 i.e. (1 attribute * 1 query + 1 attribute * 1 query). Also, site S1 doesn't access any of the four attributes of fragment 2. Therefore, the total access made by Site 1 to fragment 1 is 0.

In Table IV, maximum value for fragment 1 is contributed by Site 8 and Site 9. So all the attributes of fragment 1 are placed either on Site 8 or Site 9. Also, maximum value for fragment 2 is also contributed by Site 8 and Site 9. So all the attributes of fragment 2 are also placed on Site 8 or Site 9. Here, both Site 8 and Site 9 have same amount of access to the attribute therefore four possibilities of fragment placement are available i.e. F1 at Site 8 and F2 at Site 8; F1 at Site 8 and F2 at Site 9; F1 at Site 9 and F2 at Site 8; F1 at Site 9 and F2 at Site 9. Fragment are placed as per one of the possibilities mentioned above.

This section of the paper only shows the procedure to vertically fragment a relation of database onto two fragment clusters only. Fragmenting for more amount of cluster like 3, 4, and so on, the same clustering approach can be utilized. The main difference is only that at the initial phase of using K-Means algorithm, the required amount of cluster value is to be chosen.

V. CONCLUSION

It is likely that the use of clustering based fragmentation approach contributes some benefits in partitioning the relation of a distributed database. The method proposed in this research paper benefits the fragmentation process in two ways. These two ways are the enhancement to complexities that occurred in past methods implemented by authors in [3] and [5]. In [3], if the size of the number of sites grows then the architecture of the complete graph to generate minimum spanning tree would be huge giving rise to complexity. In [5], the more the use of attribute the more the complexity of n-ary iterative partitioning. Hence the first problem is solved by generating clusters in clustering approach without construction of complex complete graph. The next problem is solved by using just only iterative partitioning instead of n-ary iterative partitioning.

REFERENCES

[1] M. T. Ozsu and P. Valduriez, Principles of Distributed Database Systems. New Jersey: Alan Apt, 1999.

[2] A. Suganya and R. Kalaiselvi, "Efficient fragmentation and allocation in distributed databases," International Journal on Engineering, vol. 2, no. 1, pp. 1–7, January 2013.

[3] A. E. A. Raouf, N. L. Badr, and M. F. Tolba, "An optimized scheme for vertical fragmentation, allocation and replication of a distributed database," 7th International Conference on Intelligent Computing and Information Systems, IEEE, pp. 506–513, 2015.

[4] S. Khan and A. Hoque, "A new technique for database fragmentation in distributed systems," International Journal on Computer Application, vol. 5, no. 9, pp. 20–24, August 2010.

[5] S. B. Navathe, S. Ceri, G. Wiederhold, and J. Dour, "Vertical partitioning algorithms for database design," ACM TODS, vol. 9, pp. 680–710, December 1984.

[6] S. B. Navathe and M. Ra, "Vertical partitioning for database design: a graphical algorithm," SIGMOD Record, vol. 14, pp. 440–450, 1989.

[7] X. Lin, M. Orlowska, and Y. Zhang, "A graph based cluster approach for vertical partitioning in database design," Data Knowledge Engineering, vol. 11, Issue 2, pp. 151–169, 1993.

[8] F. Marir, Y. Najjar, M. AlFaress, and H. Abdalla, "An enhanced grouping algorithm for vertical partitioning problem in ddbs," 22nd International Symposium on Computer and Information Sciences, ISCIS 2007 - Proceedings, IEEE, pp. 39–44, November 2007.

[9] E. Abuelyaman, "An optimized scheme for vertical partitioning of a distributed database," IJCSNS International Journal on Computer Science, vol. 8, no. 1, pp. 310–316, January 2008.

[10] N. Singh, A. Jain, R. S. Raw, and R. Raman, "An apriori-based vertical fragmentation technique for heterogeneous distributed database transactions," Intelligent Computing, Networking, and Informatics, vol. 3, pp. 101–109, 2014.

[11] S. Song and N. Gorla, "A genetic algorithm for vertical fragmentation and access path selection," Computer Journal, vol. 43, no. 1, pp. 81–93, 2000.