A New Technique for Database Fragmentation in Distributed Systems

Shahidul Islam Khan
Department of Computer Science & Engineering
Bangladesh University of Engineering & Technology

Dr. A. S. M. Latiful Hoque
Department of Computer Science & Engineering
Bangladesh University of Engineering & Technology

ABSTRACT
Improving the performance of a database system is one of the key research issues now a day. Distributed processing is an effective way to improve reliability and performance of a database system. Distribution of data is a collection of fragmentation, allocation and replication processes. Previous research works provided fragmentation solution based on empirical data about the type and frequency of the queries submitted to a centralized system. These solutions are not suitable at the initial stage of a database design for a distributed system. In this paper we have presented a fragmentation technique that can be applied at the initial stage as well as in later stages of a distributed database system for partitioning the relations. Allocation of fragments is done simultaneously in our algorithm. Result shows that proposed technique can solve initial fragmentation problem of relational databases for distributed systems properly.

General Terms
Database**,** Distributed database, Fragmentation

Keywords
Initial Fragmentation, Allocation, Attribute locality precedence.

1. INTRODUCTION
A distributed database is a collection of data that logically belongs to the same system but is spread over the sites of a computer network. A distributed database management system (DDBMS) is defined as the software system that provides the management of the distributed database system and makes the distribution transparent to the users [1 - 2]. It is not necessary that database system have to be geographically distributed. The sites of the distributed database can have the same network address and may be in the same room but the communication between them is done over a network instead of shared memory. The communication network is the only shared resource for DDBMS [1]. As communication technology, hardware, software protocols advances rapidly and prices of network equipments falls every day, developing distributed database systems become more and more feasible. Design of efficient distributed database is one of the major research problems in database & information technology areas.

Distributed processing on database management systems (DBMS) is an efficient way of improving performance of applications that manipulate large volumes of data. This may be accomplished by removing irrelevant data accessed during the execution of queries and by reducing the data exchange among sites, which are the two main goals of the design of distributed databases [2]. Primary concern of distributed database system design is to making fragmentation of the relations in case of relational database or classes in case of object oriented databases, allocation and replication of the fragments in different sites of the distributed system, and local optimization in each site [1-3].

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 [1]. This reduces the amount of irrelevant data accessed by the applications of the database, thus reducing the number of disk accesses. Fragmentation can be horizontal, vertical or mixed/hybrid. Horizontal fragmentation (HF) allows a relation or class to be partitioned into disjoint tuples or instances. Vertical fragmentation (VF) allows a relation or class to be partitioned into disjoint sets of columns or attributes except the primary key. Combination of horizontal and vertical fragmentations to mixed or hybrid fragmentations (MF) are also proposed [3]. Allocation is the process of assigning the fragments of a database on the sites of a distributed network. When data are allocated, it may either be replicated or maintained as a single copy. The replication of fragments improves reliability and efficiency of read-only queries but increase update cost [1].

The main reasons of fragmentation of the relations are to: increase locality of reference of the queries submitted to database, improve reliability and availability of data and performance of the system, balance storage capacities and minimize communication costs among sites [1- 4].

Previous techniques of HF, VF or MF have the following problems in common:

- They use frequency of queries, minterm predicates' affinity or attribute affinity matrix (AAM) as a basis of fragmentation. These require sufficient empirical data that are not available in most cases at the initial stage.
- Most of them concentrate only fragmentation problem and overlooked allocation problem to reduce complexity.

In this paper we have presented a new technique for horizontal fragmentation of the relations of a distributed database. This technique is capable of taking proper fragmentation decision at the initial stage by using the knowledge gathered during requirement analysis phase without the help of empirical data about query execution. It can also allocate the fragments properly among the sites of DDBMS.

The rest of this paper is organized as follows. In section II we have presented literature reviews of HF, VF and MF techniques. Section III describes the system model that we have proposed.

Our results and discussion are presented in Section IV. Finally Section V concludes the paper with further research directions.

2. LITERATURE REVIEW

HF using min-term predicate is first proposed by Ceri et al. (1982) [5]. Navathe et al. (1984) used attribute usage matrix (AUM) and Bond energy algorithm to produce vertical fragments [6]. Navathe and Ra (1989) improved the previous work on VF by proposing an algorithm using a graphical technique [7]. Shin and Irani (1991) proposed knowledge based approach in which user reference clusters are derived from the user queries to the database and the knowledge about the data [8]. Ra (1993) presented a graph based algorithm for HF in which predicates are clustered based on the predicate affinities [9]. Chakravarthy et al. (1994) presented a partition evaluator to measure the goodness of a VF [10]. Navathe et al. (1995) proposed a MF technique. The input of the procedure comprises a predicate affinity table and an attribute affinity table [3]. Ozsu and Valduriez (1999) proposed an iterative algorithm COMMIN to generate a complete and minimal set of predicates from a given set of simple predicates [1]. Cheng et al. (2002) presented a genetic algorithm based fragmentation approach that treats horizontal fragmentation as a traveling salesman problem [11]. Baĭoo et al. (2004) inputted predicate affinity matrix to build a predicate affinity graph thus define horizontal class fragments [4]. Ma et al. (2006) used an attribute uses frequency matrix (AUFM) and a cost model for VF [12]. Alfares et al. (2007) used AAM to generate groups based on affinity values [13]. Marwa et al. (2008) uses the instance request matrix to horizontally fragment object oriented database [14]. Abuelyaman (2008) proposed a static algorithm StatPart for VF [15]. Mahboubi H. and Darmont J. (2009) used predicate affinity for HF in data warehouse [16].

To the best of our knowledge, only Abuelyaman [15] provided a solution for initial fragmentation of relations of a distribution database. A randomly generated reflexivity matrix, a symmetry matrix and a transitivity module has been used to produce vertical fragments of the relations and no algorithm for horizontal fragmentation. But he could not justify his hypothesis that why it will produce good fragments.

3. PROPOSED MODEL

To solve the problem of taking proper fragmentation decision at the initial stage of a distributed database, we have provided a new technique of fragmentation. That is to fragment a relation horizontally according to locality of precedence of its attributes. Attribute locality precedence (ALP) can be defined as the value of importance of an attribute with respect to sites of distributed database. ALP table will be constructed by database designer for each relation of a DDBMS at the time of designing the database with the help of modified CRUD (Create, Read, Update, and Delete) matrix and cost functions. A block diagram of our system is depicted in Figure 1.

Figure 1. Block diagram of the system

A relation in a database contains different types of attributes those describe properties of the relation. But the important thing is that the attributes of a relation do not have same importance with respect to data distribution in different sites. According to above importance we can calculate locality precedence of each attribute for each relation and construct ALP table for the relations.

A data-to-location CRUD matrix is a table of which rows indicate attributes of the entities of a relation and columns indicate different locations of the applications [18]. It is used by the system analysts and designers in the requirement analysis phase of system development life cycle for making decision of data mapping to different locations [17], [18]. We have modified the existing CRUD matrix according to our requirement of HF and name it Modified Create, Read, Update, and Delete (MCRUD) matrix. It is a table constructed by placing predicates of attributes of a relation as the rows and applications of the sites of a DDBMS as the columns. We have used MCRUD to generate ALP table for each relation.

We treated cost as the effort of access and modification of a particular attribute of a relation by an application from a particular site. For calculating precedence of an attribute of a relation we take the MCRUD matrix of the relation as an input and use the following cost functions:

$$C_{i,j,k,r} = f_C C + f_R R + f_U U + f_D D \quad (1)$$

$$S_{i,j,k} = \sum_{r=1}^{A_{ijk}} C_{i,j,k,r} \quad (2)$$

$$S_{i,j,m} = \text{Max}\,(S_{i,j,k}) \quad (3)$$

$$ALP_{ij} = S_{i,j,m} - \sum_{k \neq m}^{A_{ijk}} S_{i,j,k} \quad (4)$$

$$ALP_i = \sum_{j=1}^{l} ALP_{i,j} \quad (5)$$

Here f_C = frequency of create operation
f_R = frequency of read operation
f_U = frequency of update operation
f_D = frequency of delete operation
C = weight of create operation
R = weight of read operation
U = weight of update operation
D = weight of delete operation
$C_{i,j,k,r}$ = cost of predicate j of attribute i accessed by application r at site k
$S_{i,j,k}$ = sum of all applications' cost of predicate j of attribute i at site k
$S_{i,j,m}$ = maximum cost among the sites for predicate j of attribute i
ALP_{ij} = actual cost for predicate j of attribute i
ALP_i = total cost of attribute i (locality precedence)

For simplicity we have assumed that f_C, f_R, f_U and f_D=1 and C=2, R=1, U=3 and D=2. The justification of the assumption is that at the design time of a distributed database, the designer will not know the actual frequencies of read, delete, create and update of a particular attribute from different applications of a site and generally update incurs more cost than create and delete, and reading from database always incurs least cost.

After construction of ALP table for a relation, predicate set P will be generated for the attribute with highest precedence value in the ALP table. Finally each relation will be fragmented horizontally using the predicates of P as selection predicate. The procedures can be clearly understood from the following algorithm and pseudo code of Figure 2 and 3.

Input: Total number of sites: S = {S_1, S_2,… ,S_n}
Relation to be fragmented: R
Modified CRUD matrix: MCRUD[R]
Output: Fragments F = {F_1, F_2, F_3,…, Fn}
Step 1: Construct ALP[R] from MCRUD[R] based on Cost functions
Step 2: For the highest valued attribute of ALP table
a. Generate predicate set P={ P_1, P_2, … ,P_m }
b. Rearrange P so that #P = #S
c. Fragment R using P as selection predicate

$$\forall_p \mid \sigma_p(R)$$

d. Allocate F to S

Figure 2. FragmentationAllocation algorithm

4. RESULT AND DISCUSSION

To justify our technique we have implemented a distributed banking database system. One of the relations of the database is Accounts shown in Table 1. Initially number of sites of the distributed system is three as shown in Figure 4.

Table 1. Accounts relation

AccountNo	Type	CustId	OpenDate	Balance	BrName
01	Ind	001	20/1/09	12500	Dhk
02	Cor	002	23/1/09	35000	Dhk
03	Cor	003	28/2/09	5200	Ctg
04	Ind	004	25/3/09	15000	Khl
05	Cor	005	17/4/09	50000	Dhk

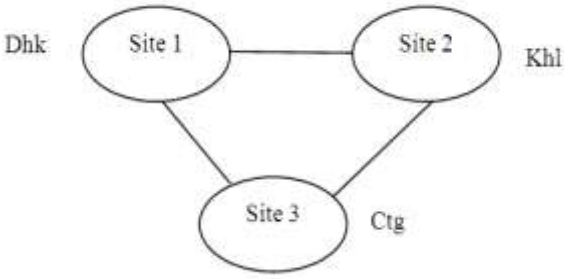

Figure 4. Distributed banking database system

Input: MCRUD of a relation that to be fragmented
Output: ALP table for that relation

```
for ( i =1; i <= TotalAttributes; i++)
{
for ( j =1; j <= TotalPredicates[i]; j++)
{
MAX[i][j] = 0;
for ( k =1; k <= TotalSites; k++)
{
for ( r =1; r <= TotalApplications[k]; r++) /* Calculating sum of
all applications' cost of predicate j of attribute i at site k */
{
C[i][j][k][r] = fc*C + fr*R + fu*U + fd*D
S[i][j][k] + = C[i][j][k][r]
}
If S[i][j][k] > MAX[i][j]      /*Find out at which site cost of
predicate j is maximum*/
{
MAX[i][j] = S[i][j][k]
POS[i][j] = k
}
SumOther = 0
for ( r =1; r <= A[i][j][k][k]; r++)
{
If (r!=k)
SumOther + = S[i][j][r]
}
}
ALPsingle[i][j] = S[i][j][POS[i][j]] – SumOther      /* actual
cost for predicate j of attribute i */
}
ALP[i] = 0
for ( j =1; j <= TotalPredicates[i]; j++)      /*calculating total
cost for attribute i (locality precedence)*/
{
ALP[i] + = ALPsingle[i][j]
}
}
```

Figure 3. ALP-table-construction Pseudo-code

4.1 Construction of MCRUD Matrix

We have constructed the MCRUD matrix for the Accounts relation in the requirement analysis phase. Part of MCRUD matrix is shown in Figure 5.

Site.Application / Entity.Attribute.Predicates	Site1			Site2			Site3		
	Ap1	Ap2	Ap3	Ap1	Ap2	Ap3	Ap1	Ap2	Ap3
Accounts .AccountNo<10000	C		RU						R
Accounts .AccountNo>=10000		R							
Accounts.Type=Ind	CRD	RU	RUD		R				
Accounts.Type=Cor		RU	R				CRUD	RU	R
. . .									
Accounts.Balance<50000	R		R			CRUD			R
Accounts.Balance>=50000		CR							
Accounts.BrName=Dhk	CRUD	RU	CRUD			R	R		
Accounts.BrName=Ctg		R		CRUD	CRUD	R		R	
Accounts.BrName=Khl							CRUD	RD	CRU

Figure 5. MCRUD matrix of Accounts

4.2. Calculation of ALP

We have calculated locality precedence of each attribute from the MCRUD matrix of Accounts relation according to the cost functions of equation (1)-(5). Calculating the locality precedence of the attribute BrName is shown in Figure 6-8.

Site.Application / Entity.Attribute.Predicates	Site1			Site2			Site3		
	Ap1	Ap2	Ap3	Ap1	Ap2	Ap3	Ap1	Ap2	Ap3
Accounts .AccountNo<10000	C		RU						R
Accounts .AccountNo>=10000		R							
Accounts.Type=Ind	CRD	RU	RUD		R				
Accounts.Type=Cor		RU	R				CRUD	RU	R
. . .									
Accounts.Balance<50000	R		R			CRUD			R
Accounts.Balance>=50000		CR							
Accounts.BrName=Dhk	CRUD	RU	CRUD			R	R		
Accounts.BrName=Ctg		R		CRUD	CRUD	R		R	
Accounts.BrName=Khl							CRUD	RD	CRU

Figure 6. ALP cost for BrName=Dhk

According to the cost functions, value of the predicate BrName=Dhk is (8+4+8) - (1+1) = 18, BrName=Ctg is (8+8+1) – (1+1) = 15 and BrName=Khl is (8+3+6) – 0 = 17. So ALP of BrName = 18+15+17 = 50.

Site.Application / Entity.Attribute.Predicates	Site1			Site2			Site3		
	Ap1	Ap2	Ap3	Ap1	Ap2	Ap3	Ap1	Ap2	Ap3
Accounts .AccountNo<10000	C		RU						R
Accounts .AccountNo>=10000		R							
Accounts.Type=Ind	CRD	RU	RUD		R				
Accounts.Type=Cor		RU	R				CRUD	RU	R
. . .									
Accounts.Balance<50000	R		R			CRUD			R
Accounts.Balance>=50000		CR							
Accounts.BrName=Dhk	CRUD	RU	CRUD			R	R		
Accounts.BrName=Ctg		R		CRUD	CRUD	R		R	
Accounts.BrName=Khl							CRUD	RD	CRU

Figure 7. ALP cost for BrName=Ctg

Site.Application / Entity.Attribute.Predicates	Site1			Site2			Site3		
	Ap1	Ap2	Ap3	Ap1	Ap2	Ap3	Ap1	Ap2	Ap3
Accounts .AccountNo<10000	C		RU						R
Accounts .AccountNo>=10000		R							
Accounts.Type=Ind	CRD	RU	RUD		R				
Accounts.Type=Cor		RU	R				CRUD	RU	R
. . .									
Accounts.Balance<50000	R		R			CRUD			R
Accounts.Balance>=50000		CR							
Accounts.BrName=Dhk	CRUD	RU	CRUD			R	R		
Accounts.BrName=Ctg		R		CRUD	CRUD	R		R	
Accounts.BrName=Khl							CRUD	RD	CRU

Figure 8. ALP cost for BrName=Khl

4.3 Construction of ALP Table

ALP values of all the attributes of the Accounts relation was computed from its MCRUD matrix. The attribute with highest precedence value will be treated as most important attribute for fragmentation. Table II shows the ALP table for Accounts relation.

Table 2. ALP table of Accounts relation

Attribute Name	Precedence
AccountNo	6
Type	22
CustId	6
OpenDate	7
Balance	10
BrName	50

4.4 Generation of Predicate Set

Predicate set was generated for BrName, the attribute with highest locality precedence of Accounts relation.

P= {p1: BrName=Dhk, p2: BrName=Ctg, p3: BrName= Khl}

4.5 Fragmentation of Relation

According to the predicate set P, Account relation was fragmented and allocated to 3 sites shown in table III- V.

Table 3. Part of Accounts relation allocated to site 1

AccountNo	Type	CustId	OpenDate	Balance	BrName
01	Ind	001	20/1/09	12500	Dhk
02	Cor	002	23/1/09	35000	Dhk
05	Cor	005	17/4/09	50000	Dhk

Table 4. Part of Accounts relation allocated to site 2

AccountNo	Type	CustId	OpenDate	Balance	BrName
04	Ind	004	25/3/09	15000	Khl

Table 5. Part of Accounts relation allocated to site 3

AccountNo	Type	CustId	OpenDate	Balance	BrName
03	Cor	003	28/2/09	5200	Ctg

4.6 Addition of a New Site to DDBMS

We have added another site in Dhk to the current DDBMS (see Figure 9). In this case the fragment in site 1 re-fragmented horizontally based on next higher precedence attribute of ALP table.

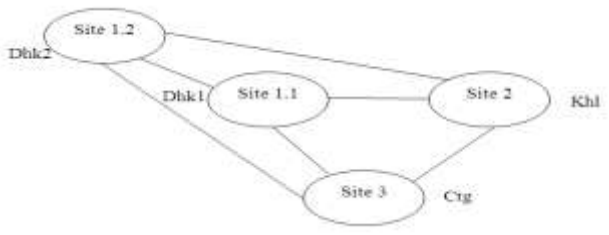

Figure 9. DBDS after new site added in Dhk

Here the attribute is Type. Predicates of Type are p_i: Type= Ind, p_{i+1}: Type= Cor. These two predicates produced minterm with the former predicate of site 1, p_1: branch=Dkh. Now $P=\{p_{11},p_{12},p_2,p_3\}$ where p_{11}: branch=Dkh Λ Type= Ind , p_{12}: branch=Dhk Λ Type= Cor. Account relation was then fragmented according to P and allocated to 4 sites as shown in Table VI- IX.

Table 6. Part of Accounts relation allocated to site 1.1

AccountNo	Type	CustId	OpenDate	Balance	BrName
01	Ind	001	20/1/09	12500	Dhk

Table 7. Part of accounts relation allocated to site 1.2

AccountNo	Type	CustId	OpenDate	Balance	BrName
02	Cor	002	23/1/09	35000	Dhk
05	Cor	005	17/4/09	50000	Dhk

Table 8. Part of accounts relation allocated to site 2

AccountNo	Type	CustId	OpenDate	Balance	BrName
04	Ind	004	25/3/09	15000	Khl

Table 9. Part of accounts relation allocated to site 3

AccountNo	Type	CustId	OpenDate	Balance	BrName
03	Cor	003	28/2/09	5200	Ctg

From the above result we can see that our technique has successfully fragmented the Accounts relation and allocated the fragments among the sites of the distributed system. As we have only taken highest valued attribute from ALP table, no unwanted fragments were created. Other relations of the distributed banking database can be fragmented in the same way like Accounts. For simplicity we have considered only four sites of the system for allocation. It is worth mentioning that our fragmentation technique will work in the same way for large number of sites of any distributed system.

5. CONCLUSIONS

Making proper fragmentation of the relations and allocation of the fragments is a major research area in distributed databases. Many techniques have been proposed by the researchers using empirical knowledge of data access and query frequencies. But proper fragmentation and allocation at the initial stage of a distributed database has not yet been addressed. In this paper we have presented a fragmentation technique to partition relations of a distributed database properly at the initial stage when no data access statistics and query execution frequencies are available. Using our technique no additional complexity is added for allocating the fragments to the sites of a distributed database as fragmentation is synchronized with allocation. So performance of a DDBMS can be improved significantly by avoiding frequent remote access and high data transfer among the sites. This research can be extended to support fragmentation in distributed object oriented databases as well.

6. REFERENCES

[1] M. T. Ozsu and P. Valduriez, Principles of Distributed Database Systems, 2nd ed., New Jersey: Prentice-Hall, 1999.

[2] S. Ceri and G. Pelagatti, Distributed Databases Principles and System, 1st ed., New York: McGraw-Hill, 1984.

[3] S. Navathe, K. Karlapalem, and M. Ra, "A mixed fragmentation methodology for initial distributed database design," Journal of Computer and Software Engineering Vol. 3, No. 4 pp 395–426, 1995.

[4] F. Bai˜ao, M. Mattoso, and G. Zaverucha, "A distribution design methodology for object DBMS," Distributed and Parallel Databases, Springer, Vol. 16, No. 1, pp. 45–90, 2004.

[5] S. Ceri, M. Negri, and G. Pelagatti, "Horizontal data partitioning in database design," in Proc. ACM SIGMOD, 1982, pp. 128–136.

[6] S. B. Navathe, S. Ceri, G. Wiederhold, and J. Dour, "Vertical partitioning algorithms for database design," ACM Transactions on Database Systems (TODS), Vol. 9, No. 4, pp. 680–710, 1984.

[7] S. B. Navathe, and M. Ra, "Vertical partitioning for database design: A graphical Algorithm," ACM SIGMOD Record, Vol. 14, No. 4, pp. 440-450, 1989.

[8] D. G. Shin, and K. B. Irani, "Fragmenting relations horizontally using a knowledge based approach," IEEE Transactions on Software Engineering (TSE), Vol. 17, No. 9, pp. 872–883, 1991.

[9] M. Ra, "Horizontal partitioning for distributed database design," In Advances in Database Research, World Scientific Publishing, pp. 101–120, 1993.

[10] S. Chakravarthy, J. Muthuraj, R. Varadarajan, and S. B. Navathe, "An objective function for vertically partitioning relations in distributed databases and its analysis," Distributed and Parallel Databases, Springer, Vol. 2, No. 2, pp. 183–207, 1994.

[11] C. H. Cheng, W. K. Lee, and K. F. Wong, "A genetic algorithm-based clustering approach for database partitioning," IEEE Transactions on Systems, Man, and Cybernetics, Vol. 32, No. 3, pp. 215–230, 2002.

[12] H. Ma, K. D. Schewe, and M. Kirchberg, "A heuristic approach to vertical fragmentation incorporating query information," in Proc. 7th International Baltic Conference on Databases and Information Systems (DB&IS), 2006, pp. 69–76.

[13] M. AlFares et al, "Vertical Partitioning for Database Design: A Grouping Algorithm", in Proc. International Conference on Software Engineering and Data Engineering (SEDE), 2007, pp. 218-223.

[14] F. F. Marwa, I. E. Ali, A. A. Hesham, "A heuristic approach for horizontal fragmentation and alllocation in DOODB," in Proc. INFOS2008, 2008, pp. 9-16.

[15] E. S. Abuelyaman, "An optimized scheme for vertical partitioning of a distributed database," Int. Journal of Computer Science & Network Security, Vol. 8, No. 1, 2008.

[16] H. Mahboubi and J. Darmont, "Enhancing XML Data Warehouse Query Performance by Fragmentation," in Proc. ACM SAC09, 2009, pp.1555-1562.

[17] J. Whitten, L. Bentley, and K. Dittman, Systems Analysis and Design Methods, 6th Ed, McGraw-Hill, 2004.

[18] P. Surmsuk, "The integrated strategic information system planning Methodology," IEEE Computer Society Press, pp. 467-475, 2007.