# Efficient Partitioning of Large Databases without Query Statistics

Shahidul Islam KHAN
Department of Computer Science and Engineering (CSE)
Bangladesh University of Engineering and Technology (BUET), Dhaka, Bangladesh
nayeemkh@gmail.com, shahid@grad.cse.buet.ac.bd

*An efficient way of improving the performance of a database management system is distributed processing. Distribution of data involves fragmentation or partitioning, replication, and allocation process. Previous research works provided partitioning based on empirical data about the type and frequency of the queries. These solutions are not suitable at the initial stage of a distributed database as query statistics are not available then. In this paper, I have presented a fragmentation technique, Matrix based Fragmentation (MMF), which can be applied at the initial stage as well as at later stages of distributed databases. Instead of using empirical data, I have developed a matrix, Modified Create, Read, Update and Delete (MCRUD), to partition a large database properly. Allocation of fragments is done simultaneously in my proposed technique. So using MMF, no additional complexity is added for allocating the fragments to the sites of a distributed database as fragmentation is synchronized with allocation. The performance of a DDBMS can be improved significantly by avoiding frequent remote access and high data transfer among the sites. Results show that proposed technique can solve the initial partitioning problem of large distributed databases.*

***Keywords****: Distributed Database, Partitioning, Fragmentation, Allocation, MCRUD matrix*

## 1 Introduction

A distributed database (DDB) is a collection of data that logically belongs to the same system but spreads over the sites of a computer network. It is not necessary that database system has 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. DDB is an efficient way of improving the performance of applications that manipulate large volumes of data. The design of efficient distributed databases is one of the major research problems in database and information technology areas. Primary concerns of distributed database design are partitioning the relations or tables, allocating them in different sites of a distributed system, and local optimization in each site [1], [2].

Database partitioning or 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. 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) partitioned a relation or class into disjoint sets of columns or attributes except the repetition of primary key column. The combination of horizontal and vertical fragmentations to form mixed or hybrid fragmentations (MF/ HF) is also proposed [3]. Allocation is the process of assigning the fragments of a database on the sites of a distributed network. The replication of fragments improves reliability and efficiency of read-only queries but increase update cost. The main reasons of fragmentation of the relations are to increase locality of reference of the queries, improve reliability and availability of data and performance of the system, balance storage capacities, and minimize

communication costs [1]-[4].

### 1.1 Problem Definition

In distributed database design, the basis of fragmentation (horizontal, vertical or mixed) of relations is one of the follows:

- Frequency of different queries executed in a system at runtime,
- Affinity matrix of minterm predicates constructed from combination of predicates
- Attribute affinity matrix constructed based on the relationship between different attributes of a table and run time transactions that access the attributes

To know actual query frequencies or to construct above matrices sufficient experiential data are required that are not available in most cases at the initial stage of a distributed database. Moreover, almost all the previous techniques concentrated only fragmentation problem and overlooked allocation problem to reduce the complexity of the problem. But the overall performance of a distributed system fragmented by a very good fragmentation technique can be very low if proper allocation of the fragments to the sites of the distributed system cannot be ensured.

Available techniques developed by the researchers so far to support fragmentation cannot provide a solution at the initial level of a distributed system. They use frequency of queries executed in a system at runtime, affinity matrix of minterm predicates constructed from combination of predicates or attribute affinity matrix constructed based on the relationship between different attributes of a table and run time transactions that access the attributes as a basis of fragmentation of the relations. To construct these matrices sufficient experiential data are required that are not available in most cases at the initial stage of a distributed system. So using currently available techniques for fragmentation, the database administrator has to put the whole database in a single site of the system and perform fragmentation and allocation after a long period when sufficient empirical data will be available to him. During this period facilities of distributed database cannot be enjoyed. After the period the database can be fragmented correctly to some extent and allocated to the sites with a high communication cost of transferring a huge amount of data from central node to all other nodes of the system. Due to the deficiencies of fragmentation and allocation techniques existing in the literature, my research focused fragmentation and allocation in an integrated manner. Based on locality of data, partitioning the database and allocating the partitions are performed with the objective of minimizing data transmission costs and maximizing locality of data

In this paper, I have presented a fragmentation technique namely Matrix based Fragmentation (MMF) that is capable of partitioning relations of a distributed database properly at the initial stage when data access statistics and query execution frequencies are not available. Instead of using empirical data, I have developed a matrix namely Modified Create, Read, Update and Delete (MCRUD) to make fragmentation decisions. 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 the performance of a DDBMS can be improved significantly by avoiding frequent remote access and high data transfer among the sites. This will improve the bandwidth of the system as well.

### 1.2 Contributions of the Paper

- The main contribution is to develop a fragmentation technique that can partition relations without empirical data.
- Relations are fragmented initial with the help of MCRUD matrix. This overcomes initial fragmentation problem of distributed database that is not properly addressed in other fragmentation techniques.

- A very good hit rate (Approximately 90%) is achieved using my proposed technique for various kinds of insertion, selection, join, deletion and other queries.
- In our technique large amount of costly data transfer using communicational network can be avoided as fragments are correctly allocated to different sites at the initial stage of the system.

The rest of the paper is organized as follows. In Section 2, a brief review of the research in horizontal, vertical and mixed fragmentation technique of distributed database is presented and limitations of the available fragmentation techniques are also discussed. Section 3 describes the details of Matrix based Fragmentation (MMF) technique In Section 4, I have presented the results of the experiments to show the performance of my proposed technique. Finally, Section 5 concludes the paper and provides suggestions for future research.

## 2 Literature Review

### 2.1Complexity of the problem

The combined problem of fragmentation and allocation is proven NP-hard [6]. In the case of Horizontal fragmentation, if n simple predicates are considered $2^n$ is the number of horizontal fragments using minterm predicates. If there are k nodes, the complexity of allocating horizontal fragments is O ( ).

For example, using 6 simple predicates to perform horizontal fragmentation results in $2^6 = 64$ fragments. To find the optimal allocation of the fragments in 4 sites one needs to compare all the $4^{64} \approx 10^{39}$ possible allocations.

For vertical fragmentation, if a relation has m non-primary key attributes, the number of possible fragments: Bell number B (m) ≈ $m^m$. The fragment allocation is of complexity

O ( ). Due to the complexity of both fragmentation and allocation, allocation of the fragments are often treated independently than fragmentation of the database.

### 2.2 Horizontal Fragmentation

There are two types of horizontal fragmentation, primary and derived. Primary horizontal fragmentation of a relation or a class is performed using predicates of queries accessing this relation or class, while derived horizontal fragmentation of a relation or a class is performed based on horizontal fragmentation of another relation or class.

In the context of the relational data model, existing approaches for horizontal fragmentation mainly fall into following three categories [7], [1]:

- minterm-predicate-based approaches: which perform primary horizontal fragmentation using a set of minterm predicates, e.g., [1], [2], [8].
- affinity-based approaches: which, at first, group predicates according to predicate affinities and then perform primary horizontal fragmentation using conjunctions of the grouped predicates, e.g., [9] - [12].
- other approaches: approaches other than minterm predicate or predicate affinity-based approach, e.g., [13] – [16].

### 2.3 Vertical Fragmentation

Vertical fragmentation has been studied since the 1970s. There are two main approaches [7]:

- The pure affinity-based approach takes attribute affinities as the measure of togetherness of attributes to fragment attributes of a relation schema. Research work includes [17]-[24].
- The cost-driven approach uses a cost model while partitioning attributes of a relation schema. Research work includes [25] - [30].

*Initial Vertical Fragmentation*

Abuelyaman [30] provided a solution of initial fragmentation of database using vertical fragmentation technique namely

StatPart. To fragment a relation, it starts with a randomly generated matrix of attribute vs. queries called the reflexivity matrix. It then constructs symmetry matrix from the reflexivity matrix using two equations. Symmetry matrix is inputted to transitivity module which uses an algorithm to produce two set of attributes those will be used to break the relation into two binary vertical fragments.

Main two drawbacks of StatPart [30] are:

- It can suggest only two binary vertical fragments independent of the number of sites of the distributed system. So this technique is not suitable for a distributed system with more than two allocation sites.
- As it starts with a randomly generated matrix that represents the relationship among attributes and queries, optimum fragmentation decision cannot be provided using this algorithm. So it continuously shifts attributes from one fragment to another fragment trial and error basis to improve hit ratio.

Recent research on Horizontal or Vertical partitioning includes fragmentation of very large databases, cloud-based systems, multimedia databases etc. [31]-[35].

### 2.4 Mixed Fragmentation

Navathe et al. [3] proposed a mixed fragmentation methodology that simultaneously applies horizontal and vertical fragmentation on a relation. The input of the procedure comprises a predicate affinity table and an attribute affinity table. A set of grid cells is created first which may overlap each other. Then some grid cells are merged such that total disk accesses for all transactions can be reduced. Finally, the overlap between each pair of fragments is removed using two algorithms for the cases of contained and overlapping fragments.

Adopting some developed heuristics and algorithms in [3] to fragmentation in object oriented databases, Baĩao and Mattoso [36] proposed a design procedure which includes a sequence of steps: analysis phase, vertical and horizontal fragmentation. In the first step, a set of classes that are needed for horizontal fragmentation, vertical fragmentation, or non-fragmentation, are identified. In the second and third steps, vertical and horizontal fragmentations are performed on the classes identified in the first step, using algorithms extended from the one in [3]. All fragmentation algorithms are affinity based. The evaluations of the resulting fragmentation are not based on any cost model. Baĩao et al. [4] considered mixed fragmentation as a process of performing vertical fragmentation on classes first and then performing horizontal fragmentation on the set of vertical fragments.

### 2.5 Allocation

In the literature, allocation problems are first addressed for file allocation. Chu [37] presented a simple model for a non-redundant allocation of files. Casey [56] proposed a model which allows the allocation of multiple copies. Queries and updates are distinguished in the model. Mahmoud and Riodon [38] proposed a model for studying file allocation and the capacity of communication capacities to obtain an optimized solution which minimizes storage and communication cost. Since the early 1980s, data allocation has been studied in the context of relational databases. Due to the complexity of the problem of data allocation, different researchers make different assumptions to reduce the size of the problem. Some works do not consider replication while making a decision of allocation [39, 40] while some others do not consider storage capabilities of network nodes [6, 41].

### 2.6 Summary

Most of the literature about database distribution considers fragmentation and allocation as two different steps even though they are strongly related problems. Both fragmentation and allocation take the same input information to achieve the same

objectives of improving system performance, reliability, and availability. Existing approaches for primary horizontal fragmentation can be characterized into three streams, one using minterm predicates, one using predicate affinity, and a cost-driven approach using a cost model. Even though each of the approaches claims to be able to improve system performance, there is no evaluation to prove that resulting fragmentation schemata can indeed improve the system performance. Horizontal fragmentation with minterm predicates often results in a large number of fragments which will later be allocated to a limited number of network nodes. Affinity-based horizontal fragmentation approaches cannot guarantee to achieve optimal system performance because the information of data local requirement is lost while computing predicate affinities. Cost-driven approaches use cost models to measure the number of disk accesses without considering transportation cost.

For vertical fragmentation, there are two main approaches existing in the literature: affinity based and cost-based. The affinity-based vertical fragmentation approach originated for centralized databases with hierarchical memory levels, for which the number of disk accesses is the main factor that affects the system performance. Later, this approach was adapted to distributed databases for which transportation cost is the main cost that affects the system performance. Attribute affinities only reflect the togetherness of attributes accessed by applications. Vertical fragmentation based on affinities may reduce the number of disk accesses. However, there is no clear proof that affinity-based vertical fragmentation can indeed improve data local availability and thus improve system performance. The cost-driven approach performs vertical fragmentation based on a cost model that measures the number of disk accesses. The optimal solution chosen by this approach is the vertical fragmentation schema that has the fewest number of disk accesses. However, there is no fragmentation approach, for both horizontal and vertical fragmentation, taking data locality into consideration.

Due to the complexity of the allocation problem, it is infeasible to find optimal solutions. Researchers provided heuristic solutions with many assumptions to reduce the complexity of the problem. The assumption that fragmentation is completed properly is not reasonable. Because it is not possible to solve the fragmentation problem independently from the allocation problem as the optimal fragmentation can only be achieved with respect to the optimal allocation of fragments.

## 3 Matrix based Fragmentation Technique (MMF)

To solve the problem of taking proper fragmentation decision at the initial stage of a distributed database, I have developed a new partitioning technique based on locality precedence of the attributes. Instead of using empirical data, I have developed Modified Create, Read, Update and Delete (MCRUD) matrix to obtain fragmentation decisions. The details of the technique are discussed in the following sections.

### 3.1 CRUD Matrix

A data-to-location CRUD matrix is a table in which rows indicate attributes of the entities of a relation and columns indicate locations of the applications [42]. It is used by the system analysts and designers in the requirement analysis phase of system development life cycle for making a decision of data mapping to different locations [42], [43]. An example of a traditional CRUD Matrix is shown in the Fig. 1.

| Entity / Use Case | Order | Chemicals | Requestor | Vendor Catalog |
|---|---|---|---|---|
| Place Order | C | R | R | R |
| Change Order | U, D | | R | R |
| Manage Chemical Inventory | | C, U, D | | |
| Report on Orders | R | R | R | |
| Edit Requesters | | | C, U | |

**Fig. 1** Example of a CRUD Matrix adopted from [44]

### 3.2 MCRUD Matrix

I have modified the existing CRUD matrix according to our requirement of horizontal fragmentation and name it Modified Create, Read, Update, and Delete (MCRUD) matrix. It is a table constructed by placing predicates of attributes of a relation on the row side and applications of the sites of a DDBMS on the column side. I have used MCRUD matrix to generate attribute locality precedence (ALP) table for each relation. An example of an MCRUD Matrix is shown in Fig. 2. In this example, the distributed system has three sites and one application is running on each site. Entity set, attribute, and predicate are denoted by *e*, *a* and *p* respectively. If an application of a site has chances to perform create or read or update or delete operation to an attribute's certain predicate then C or R or U or D will be written in the intersecting cell of the matrix.

| Site.Application / Entity.Attribute.Predicates | Site1 | Site2 | Site3 |
|---|---|---|---|
| | Ap1 | Ap1 | Ap1 |
| $e.a_1.p_1$ | CRUD | R | R |
| $e.a_1.p_2$ | RU | CRUD | CRU |
| $e.a_2p_1$ | R | R | CRUD |
| $e.a_2.p_2$ | R | RU | R |
| $e.a_3.p_1$ | CRUD | | R |
| $e.a_3.p_2$ | R | R | CRUD |

**Fig. 2** Example of a MCRUD Matrix

### 3.3 Attribute Locality Precedence (ALP)

In my developed technique, a relation is fragmented according to the locality of precedence of its attributes. Attribute Locality Precedence (ALP) as the value of importance of an attribute with respect to the sites of a distributed database [45], [46]. 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. For example in Fig. 2, there are three attributes $a_1$, $a_2$ and $a_3$. Among them, one may be more significant than others to increase data locality and to reduce remote access in the case of fragmentation. According to the above importance, we can calculate locality precedence of each attribute for each relation and construct ALP table for the relations.

### 3.4 ALP Table

ALP values of different attributes of a relation will be placed in a table called ALP table. 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 MCRUD matrix and cost functions. The algorithm that will be used to calculate ALP and to construct ALP table is given in Algorithm I. An example of ALP table for the MCRUD matrix of Fig. 2 is shown in Table 1.

Algorithm I: ALP calculation

```
Input: MCRUD of a relation
Output: ALP table of the
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
                  Count =0
for ( r =1; r <= A[i][j][k];
r++){
If (r!=k)
SumOther + = S[i][j][r]
If S[i][j][r]>MAX[i][j]/2
/* selecting the sites where
      Replicate[Count]=r
replication of a fragment
                     Count++
                  will be
performed */
           ALPsingle[i][j] =
MAX[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]
```

Table 1 Example of an ALP table

| Entity. Attribute Name | Precedence |
|---|---|
| $e.a_1$ | 4 |
| $e.a_2$ | 8 |
| $e.a_3$ | 13 |

### 3.5 ALP Cost Functions

I treated cost as the effort of access and modification of a particular attribute of a relation to an application from a particular site. For calculating precedence of an attribute of a relation, I take the MCRUD matrix of the relation as an input and use the cost functions of [45], [46].

Using the form of Table 2, more accurate estimation of the frequency of *create*, *read*, *update* and *delete* operation by an application can be possible. This form will be used at the requirement analysis phase of a DDBMS design.

Table 2 Information Need Analysis Form

| Access Statistics / Users | Site k | | | |
|---|---|---|---|---|
| | Application r | | | |
| | $attribute_i$. $predicate_j$ | | | |
| | Create | Read | Update | Delete |
| $U_1$ | | x | | |
| $U_2$ | | x | x | |
| $U_3$ | x | x | x | x |
| $U_4$ | | x | | |
| . . . | | | | |
| $U_n$ | x | x | x | |

### 3.6 Fragmentation based on MCRUD Matrix

Here, I am describing MMF technique in details. The main functionalities of the technique are shown in Fig. 3 adopted from [46]. There are *n* numbers of relations in the database named $R_1$, $R_2$,…, $R_n$. First n number of MCRUD matrices will be constructed by the system designer at design time. These *n* matrices will be the input of our technique. Then using the cost functions, *n* number of ALP tables ALP ($R_1$), ALP ($R_2$), …, ALP ($R_n$) will be constructed. Then in the next step, *n* numbers of predicate sets named $P_1$, $P_2$, …, $P_n$ will be generated for attributes with highest ALP value for each ALP table. Each predicate set $P_i$ will contain *m* numbers of predicates. According to the predicate sets, each of the *n* relations $R_i$ will be fragmented into m fragments and allocate to the *m* sites.

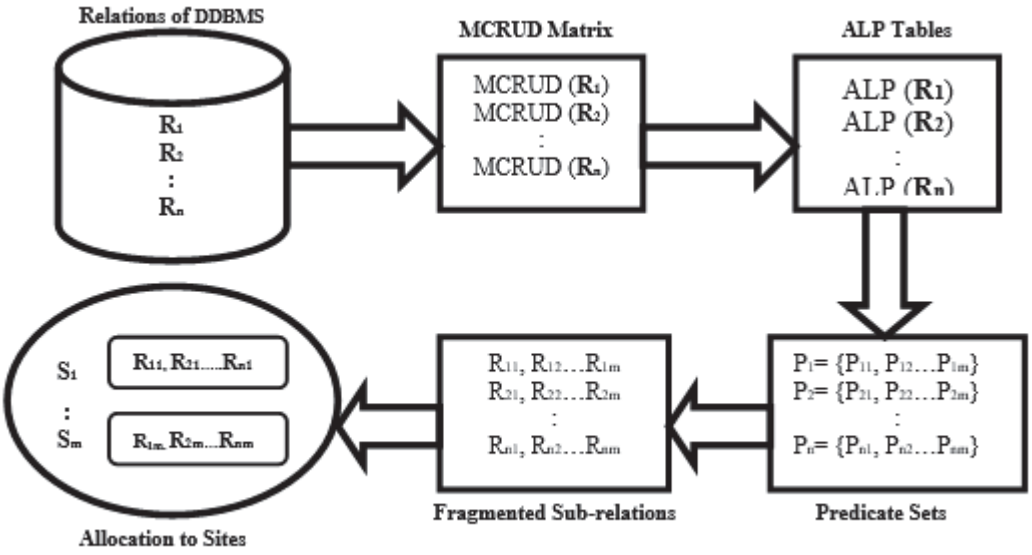


**Fig. 3** Block diagram of MMF

Following algorithm, Algorithm II has been used to implement MMF technique.

Algorithm II: FragmentationAllocation

```
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 significant highest valued attribute of ALP table*
*a. Generate predicate set P={ $P_1$, $P_2$, … ,$P_m$ }*
*b. Fragment R using P as selection predicate*

$$\forall_P \mid \sigma_P(R)$$

*c. ALLOCATE F to S*
*Step 3: For non-significant-highest-value (Max-Highest<1.5*2nd-Highest) in ALP[R]*

*a. REPLICATE R to* $\sum_{j=1}^{n} S_j$ *if R is an entity set*
*b. Derive Horizontally Fragment R using its owner relation if R is a relationship set*

---

Algorithm II takes a relation to be fragmented, MCRUD matrix of the relation and number of allocation sites as input. It finally produces fragments and allocates them in the sites of DDBMS.

### 3.7 Implementation of other Fragmentation Types

In this paper, I have performed the fragmentation of the relations of distributed database using horizontal fragmentation technique. This is because of improving performance significantly of a distributed database, we have to maximize locality of data or hit rate of the queries. That is query generating in one site access data of that site only. This will reduce remote access cost and cost of data transfer among the sites. The locality of data can be achieved more using horizontal fragmentation than vertical fragmentation.

MMF technique is not limited to horizontal fragmentation only. If we slightly modify the MCRUD matrix that is if we place attributes of a relation on the row side and applications of the sites of a DDBMS on the column side and modifying the cost functions we can produce vertical fragmentation using MMF technique. Modification of MCRUD matrix for vertical fragmentation is shown in Fig. 4:

| Site.Application / Entity.Attribute | Site1 | | | Site2 | | | Site3 | | |
|---|---|---|---|---|---|---|---|---|---|
| | Ap1 | Ap2 | Ap3 | Ap1 | Ap2 | Ap3 | Ap1 | Ap2 | Ap3 |
| Accounts .AccountNo | C | | RU | | | | | | R |
| Accounts.Type | CRD | RU | RUD | | R | | | | |
| . . | | | | | | | | | |
| Accounts.Balance | R | | R | | | CRUD | | | R |
| Accounts.BrName | CRUD | RU | CRUD | | | R | R | | |

**Fig. 4** MCRUD Matrix for Vertical Fragmentation

Like other Hybrid or Mixed fragmentation techniques, MF can be performed in our MMF technique by applying vertical fragmentation followed by horizontal fragmentation or vice versa. If it worth mentioning that MF is only applied in distributed databases if the relations have too many attributes and a huge number of records in the relations.

## 4 Results and Discussion

The objective of my experimental works is to verify the applicability and feasibility of MMF, the proposed fragmentation technique based on MCRUD matrix. The experimental evaluation has been performed with

synthetic data and a reasonable number of queries.

### 4.1 Experimental Environment

To validate proposed technique, I have implemented a distributed banking database system in the post-graduate lab of BUET using DELL workstations. I have used Windows XP operating system and Oracle 10g for database creation. Schema of the implemented database is shown in Fig. 5.

Customer-Schema = (Cid, Cname, Caddr, Cphn, BrNo)

Loans-Schema = (LnNo, LnType, Amount)

Accounts-Schema = (AccNo, AccType, AccBalance)

Branch-Schema = (BrNo, BrName, BrAddress)

LnCust-Schema = (LnNo, Cid)

AccCust-Schema = (AccNo, Cid)

AccofBranch-Schema = (AccNo, Opendate, Status, BrNo)

LnofBranch-Schema = (LnNo, Issuedate, Status, BrNo)

**Fig. 5** Relation schema

Initially number of sites of the distributed system is three as shown in Fig. 6. In each site, three applications were executed.

- Application 1 deals with Customer related information.
- Application 2 deals with Account related information.
- Application 3 deals with Loan related information.

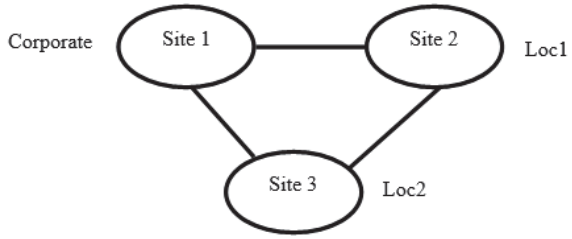


**Fig. 6** Distributed banking database system

### 4.2 Construction of MCRUD Matrix

I have constructed the MCRUD matrix for each of the eight relations of Fig. 5 in the requirement analysis phase. An MCRUD matrix is constructed for each relation by placing predicates of attributes in the row side and applications of the sites of a DDB on the column side of a table in the requirement analysis phase of system development. Two of the matrices constructed are shown in Table 3 - 4.

Table 3 MCRUD matrix of Branch relation

| Site.Application | Site1 | | | Site2 | | | Site3 | | |
|---|---|---|---|---|---|---|---|---|---|
| Entity.Attribute.Predicates | Ap1 | Ap2 | Ap3 | Ap1 | Ap2 | Ap3 | Ap1 | Ap2 | Ap3 |
| Branch.BrNo=B01 | R | R | R | | | | | R | |
| Branch.BrNo=B02 | | | R | R | | R | | | |
| Branch.BrNo=B03 | | | | | | | R | | R |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Branch.BrName=Corporate | R | R | | | | | | | |
| Branch.BrName=Loc1 | | | | R | R | | | R | |
| Branch.BrName=Loc2 | R | | | R | | | R | R | |
| Branch.BrAddress=? | | | R | | | | | | |

Table 4 MCRUD matrix of Loan relation

| Site.Application<br><br>Entity.Attribute.Predicates | Site1 | | | Site2 | | | Site3 | | |
|---|---|---|---|---|---|---|---|---|---|
| | Ap1 | Ap2 | Ap3 | Ap1 | Ap2 | Ap3 | Ap1 | Ap2 | Ap3 |
| Loan .LnNo<10000 | RU | R | CRUD | RU | R | CRUD | R | R | CRUD |
| Loan .LnNo>=10000 | R | R | CRUD | R | RU | CRUD | R | RU | CRUD |
| Loan.LnType=SME | R | | RU | RU | R | CRUD | R | | RU |
| Loan.LnType=HOME | RU | RU | CRUD | R | | RU | R | | RU |
| Loan.LnType=CAR | R | | RU | R | | RU | RU | | CRUD |
| Loan.Amount<50000 | R | | CRUD | R | | CRUD | R | | CRUD |
| Loan.Amount=50000:100000 | R | R | CRUD | R | | CRUD | R | | CRUD |
| Loan.Amount>100000 | R | | CRUD | R | | CRUD | R | | CRUD |

### 4.3 Calculation of ALP Values and Construction of ALP Tables

I have calculated locality precedence of each attribute from the MCRUD matrix of each relation using attribute locality precedence (ALP) calculation algorithm. Using the ALP values I have constructed ALP table for each relation. ALP table is a 2D array where attributes of a relation and its locality precedence is stored. For each attribute, Create, Read, Update, and Delete operation over its predicates from different applications of different sites is calculated and sum up to have locality precedence of that attribute. An attribute with the highest precedence implies that taking predicates of this attribute as selection predicate for horizontal fragmentation will maximize the hit ratio. It is depicted in Table 5.

Table 5 Precedence calculation for LnType attribute of Loan relation

| Attribute Name | Predicates | Precedence in Site 1 | Precedence in Site 2 | Precedence in Site 3 | Precedence of Predicate | ALP | Decision |
|---|---|---|---|---|---|---|---|
| LnType | LnType = SME | 5 | 13 | 5 | 13-5-5=3 | 3+6+2=11 | Fragment in Site 2 |
| | LnType = HOME | 16 | 5 | 5 | 16-5-5=6 | | Fragment in Site 1 |
| | LnType = CAR | 5 | 5 | 12 | 12-5-5=2 | | Fragment in Site 3 |

Table 6 ALP table of Loan relation

| Attribute Name | Precedence |
|---|---|
| LnNo | -20 |
| LnType | 11 |
| LnAmount | -26 |

### 4.4 Generation of Predicate Set and Fragmentation of the Relations

Predicate set was generated for the attributes with highest locality precedence of the relations respectively. These predicate sets were used to fragment the relations.

$P_{Loan}$ ={LnType=SME, LnType=HOME , LnType=CAR }
$P_{Customer}$ ={BrNo=B01, BrNo=B02, BrNo=B03}
$P_{Accounts}$ ={AccType=Ind, AccType=Cor}
$P_{AccofBranch}$ ={BrNo=B01, BrNo=B02, BrNo=B03}
$P_{LnofBranch}$ ={BrNo=B01, BrNo=B02, BrNo=B03}

As for AccCust and LnCust relations, no attribute has significant higher precedence than other attributes, so predicate set was not generated for the relations. Instead these relations are to be fragmented derived horizontally with the help of their mother relation.

For Horizontal fragmentation of Customer relation, following queries are used:

$Q_{Customer1}$ =Select * from Customer where BrNo=B01;
$Q_{Customer2}$ =Select * from Customer where BrNo=B02;
$Q_{Customer3}$ =Select * from Customer where BrNo=B03;

For Horizontal fragmentation of Loan relation, following queries are used:

$Q_{Loan1}$ =Select * from Loan where LnType=SME;
$Q_{Loan2}$ =Select * from Loan where LnType= HOME;
$Q_{Loan3}$ =Select * from Loan where LnType= CAR;

For Horizontal fragmentation of Accounts relation, following queries are used:

$Q_{Accounts1}$ =Select * from Accounts where AccType=Ind;
$Q_{Accounts2}$ =Select * from Accounts where AccType=Cor;

For Horizontal fragmentation of AccofBranch relation, following queries are used:

$Q_{AccofBranch1}$ =Select * from AccofBranch where BrNo=B01;
$Q_{AccofBranch2}$ =Select * from AccofBranch where BrNo=B02;
$Q_{AccofBranch3}$ =Select * from AccofBranch where BrNo=B03;

For Horizontal fragmentation of LnofBranch relation following queries are used:

$Q_{LnofBranch1}$ =Select * from LnofBranch where BrNo=B01;
$Q_{LnofBranch2}$ =Select * from LnofBranch where BrNo=B02;
$Q_{LnofBranch3}$ =Select * from LnofBranch where BrNo=B03;

For Horizontal fragmentation of AccCust relation, following queries are used:

$Q_{AccCust1}$ =Select AccNo, Cid from AccCust, Customer where AccCust.Cid = Customer.Cid and Customer.BrNo=B01;
$Q_{AccCust2}$ =Select AccNo, Cid from AccCust, Customer where AccCust.Cid = Customer.Cid and Customer.BrNo=B02;
$Q_{AccCust3}$ =Select AccNo, Cid from AccCust, Customer where AccCust.Cid = Customer.Cid and Customer.BrNo=B03;

For Horizontal fragmentation of LnCust relation, following queries are used:

$Q_{LnCust1}$ =Select LnNo, Cid from LnCust, Customer where LnCust.Cid = Customer.Cid and Customer.BrNo=B01;
$Q_{LnCust2}$ = Select LnNo, Cid from LnCust, Customer where LnCust.Cid = Customer.Cid and Customer.BrNo=B02;
$Q_{LnCust3}$ = Select LnNo, Cid from LnCust, Customer where LnCust.Cid = Customer.Cid and Customer.BrNo=B03;

Branch relation was not fragmented as it is a very small relation and most access to its records is by *read* operation. Instead, Branch relation was replicated to all the sites of the DBDS.

In this way all the relation schemas of the distributed banking system of Fig. 5 were fragmented using the above queries and allocated to the three computers (sites).

### 4.5 Queries for Performance analysis of Matrix based Fragmentation (MMF)

I have executed twenty queries in each site with a total of sixty selected queries in the distributed system according to Pareto Principle often referred as 80/20 rule [47] – [49] to see the performance of MMF. The queries were selected from the following *query domain* to accomplish enough variation of real database system:

- Insertion e.g. Insert into RRR values (xxx, yyy, zzz);
- Selection (Point) e.g. Select $A_1$, $A_{2...}$

$A_n$ from RRR where xxx= P

- Selection (Range) e.g. Select $A_1$, $A_2$... $A_n$ from RRR where xxx< BBB
- Selection (Join) e.g. Select $A_1$, $A_2$ ... $A_n$ from $R_1$, $R_2$ where $R_1.A_i=R_2.A_j$ AND $R_1.A_k$=CCC
- Selection (Aggregation) e.g. Select Sum (AA) from RRR where P
- Update e.g. Update RRR set $A_i$ = xxx where $A_j$ = yyy
- Deletion e.g. Delete * from RRR where P

I have defined hit as a result of a query of any type accessed records of a local fragment of the site where the query was initiated and miss as a result of a query of any type accessed records of one or more remote fragments of other sites. Partial results of my experiments are shown in Table 7 – 8 and Fig. 7 – 8:

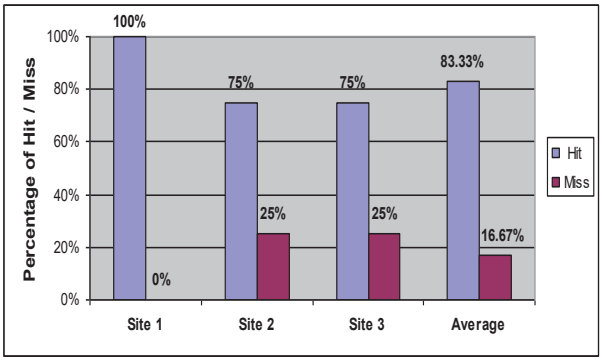


Fig. 7 Hit Miss ratio for Loan relation

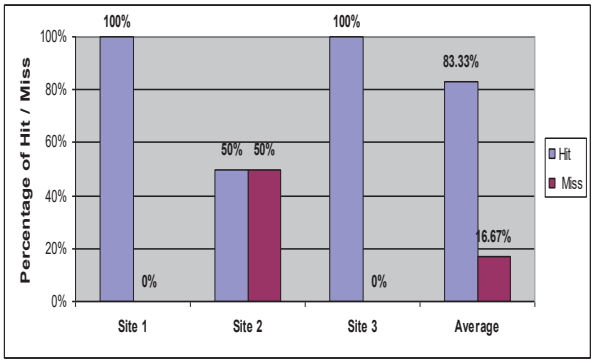


Fig. 8 Hit Miss ratio for Accounts relation

From Table 7 we can see that all the queries of Site 1 accessed records from local fragment of Loan relation. So hit ratio in Site 1 is 100%. We also see that 75% queries executed at Site 2 and Site 3 accessed records of local fragment and 25% queries accessed records of fragment stored on other (remote) site rather than query generation site. Average hit ratio for Loan relation is 83.33%.

Table 7 Hit Miss ratio for Loan

| Site | Percentage of Hit | Percentage of Miss |
|---|---|---|
| Site 1 | 100% | 0% |
| Site 2 | 75% | 25% |
| Site 3 | 75% | 25% |
| Average | 83.33% | 16.67% |

From Table 8 we can see that all the queries of Site 1 and Site 3 accessed local fragment of Accounts relation.
So hit ratio in Site 1 and 3 is 100%. 50% of queries executed at Site 2 accessed records of the local fragment and 50% queries accessed records of fragment stored on other (remote) site rather than Site 2. Average hit ratio for Accounts relation is 83.33%.

Table 8 Hit Miss ratio for Accounts

| Site | Percentage of Hit | Percentage of Miss |
|---|---|---|
| Site 1 | 100% | 0% |
| Site 2 | 50% | 50% |
| Site 3 | 100% | 0% |
| Average | 83.33% | 16.67% |

Table 9 shows the overall performance of the distributed system after fragmenting the relations using MMF technique. We can see that after fragmentation and allocation using MMF technique, 85.71% of the queries generated in any site accessed records of only that site and remote access reduced to 14.29%. This is definitely a significant achievement.

Table 9 Overall System Performances of MMF

| Site Name | Queries executed | Accessed fragment stored in local site | Accessed fragment stored in remote site | Percentage of Hit | Percentage of Miss |
|---|---|---|---|---|---|
| Site 1 | 20 | 19 | 1 | 92.86 % | 7.14 % |
| Site 2 | 20 | 16 | 3 | 78.57% | 21.43% |
| Site 3 | 20 | 17 | 2 | 85.71% | 14.29% |
| DDBMS | 60 | 52 | 6 | 86.6 % | 13.4% |

**4.6 Comparison with other Techniques**

I have named the techniques deals with fragmentation problem of distributed database without addressing the initial stage problem as Techniques Without Initial Fragmentation (TWIF) as in [1] – [30]. TWIF first store the relations of a distributed database in a single site of the distributed system as a centralized database. The other sites where the database is not stored, access the database with various type of queries using remote network connection of the system. Information about attribute, predicate access pattern and frequencies of access by different queries from different sites are gathered in tables called Attribute Usage Matrix (AUM) or Predicate Usage Matrix (PUM) or similar tables. After a certain period when sufficient statistical data are gathered for calculating the relationship (known as affinity) of an attribute or predicate with the transaction of sites, Attribute Affinity Matrix (AAM) or Predicate Affinity Matrix (PAM) are generated using Bond Energy algorithm or similar algorithm. From AAM and PAM, vertical and horizontal fragmentation decision is made respectively. Then produced fragments are to be stored in the sites of the distributed database though almost all TWIF ignore allocation of the fragments to reduce complexity.
I have implemented the above model in the lab and execute the same forty-two queries those were used to test our technique with the assumption that at the initial stage the centralized database is stored at Site 1. Table 10 shows the overall system performance of

TWIF before DDBMS is fragmented and allocated to sites. We can see that during a long period before reasonable amount of statistical record access frequencies by transactions are available for constructing attribute affinity matrix or predicate affinity matrix and to fragment and allocate the database among the three sites, percentage of hit of the overall system is only 33.33% which is much less in comparison with our achieved 85.71% hit rate.

Table 10 Overall System Performance of TWIF

| Site Name | Queries executed | Access fragment stored in local site | Access fragment stored in remote site | Percentage of Hit | Percentage of Miss |
|---|---|---|---|---|---|
| Site 1 | 20 | 20 | 0 | 100% | 0% |
| Site 2 | 20 | 0 | 14 | 0% | 100% |
| Site 3 | 20 | 0 | 14 | 0% | 100% |
| DDBMS | 60 | 20 | 28 | 33.33% | 66.66% |

### 4.7 Impact of Site Number Increase

Now we want to experiment the generalization of MMF so that we can verify if our technique is applicable to any number of sites of distributed system. I have increased a total number of sites to four at design time by adding a local branch of DBDB named Loc3 at Site 4. This situation is depicted in Fig. 9

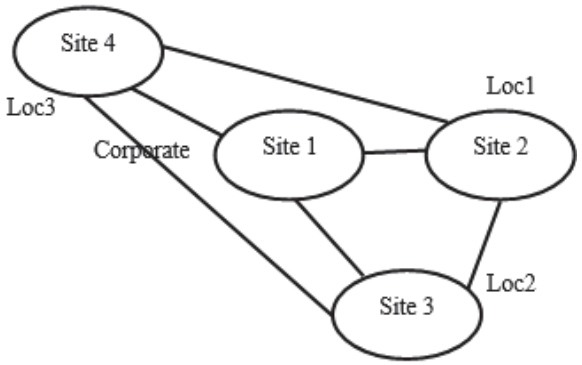


Fig. 9 DBDB with four sites

I have constructed the MCRUD matrix of Loan relation for four sites with three applications running on each site. It is shown in Table 11 below:

Table 11 MCRUD matrix of Loan relation for four sites

| Site.Application | Site1 | | | Site2 | | | Site3 | | | Site4 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Entity.Attribute.Predicates | Ap1 | Ap2 | Ap3 | Ap1 | Ap2 | Ap3 | Ap1 | Ap2 | Ap3 | Ap1 | Ap2 | Ap3 |
| Loan .LnNo<10000 | RU | R | CRUD | RU | R | CRUD | R | R | CRUD | RU | R | CRUD |
| Loan .LnNo>=10000 | R | R | CRUD | R | RU | CRUD | R | RU | CRUD | R | RU | CRUD |
| Loan.LnType=SME | R | | RU | RU | R | CRUD | R | | RU | RU | | CRUD |
| Loan.LnType=HOME | RU | RU | CRUD | R | | RU | R | | RU | R | | RU |
| Loan.LnType=CAR | R | | RU | R | | RU | RU | | CRUD | RU | R | CRUD |

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Loan.Amount<50000 | R | | CRUD | R | | CRUD | R | | CRUD | RU | R | CRUD |
| Loan.Amount=50000:100000 | R | R | CRUD | R | | CRUD | R | | CRUD | R | R | CRUD |
| Loan.Amount>100000 | RU | R | CRUD | R | | CRUD | R | | CRUD | R | | RU |

From Table 11, I have calculated ALP table for Loan relation shown in Table 12. The process of how fragmentation and replication decision is made in four sites can be understood from Table 13.

Table 12 ALP table of Loan relation with four sites

| Attribute Name | Precedence |
|---|---|
| LnNo | -46 |
| LnType | -17 |
| LnAmount | -42 |

Table 13 Precedence calculation and fragmentation decision for Loan relation

| Attribute Name | Predicates | Precedence in Site 1 | Precedence in Site 2 | Precedence in Site 3 | Precedence in Site 4 | Decision |
|---|---|---|---|---|---|---|
| LnType | LnType = SME | 5 | 13 | 5 | 12 | Fragment in Site 2 Replica in site 4 |
| | LnType = HOME | 16 | 5 | 5 | 5 | Fragment in Site 1 |
| | LnType = CAR | 5 | 5 | 12 | 13 | Fragment in Site 4 Replica in site 3 |

Predicate set is generated for the attribute LnType of Loan relation.

$P_{Loan}$ ={LnType=SME, LnType=HOME , LnType=CAR }

For Horizontal fragmentation of Loan relation, following queries were used:

$Q_{Loan1}$ =Select * from Loan where LnType=HOME;

$Q_{Loan2}$ =Select * from Loan where LnType= SME;

$Q_{Loan3}$ =Select * from Loan where LnType= CAR;

$Q_{Loan4.1}$=Select * from Loan where LnType=SME;

$Q_{Loan4.2}$ =Select * from Loan where LnType= CAR;

I have executed same queries as previous in four sites of DBDS to check the impact of site addition on the hit-miss ratio. The result is shown in Table 14 and Fig 10.

We can see that average hit ratio is 81.25% that is very close to our previous result 83.33% achieved for three sites.

Table 14 Performance table of MMF for Loan relation distributed in four sites

| Site | Percentage of Hit | Percentage of Miss |
|---|---|---|
| Site 1 | 100% | 0% |
| Site 2 | 75% | 25% |
| Site 3 | 75% | 25% |
| Site 4 | 75% | 25% |
| Average | 81.25% | 19.75% |

Fig. 10 shows the performance of MMF and TWIF with the increase of a number of sites in the distributed system. We can see that MMF shows better and quite steady performance as sites increases from three to five. In the same time, performance of TWIF falls gradually as new sites are added to the system.

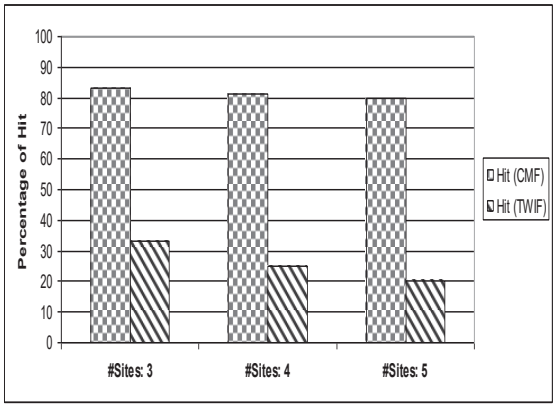


Fig. 10 Comparison of Hit ratio between MMF and TWIF with increasing number of sites

### 4.8 Summary

From the above result, we can see that our technique has clearly outperformed the technique stated in [30]. Our fragmentation technique achieved a very good hit rate which is approximately 84%. As other techniques described in [1] – [29] could not provide solutions for initial state of the distributed system, using TWIF initial performance (hit ratio) of the system is only 33.33%. After a long period when sufficient data for fragmenting the centralize database were available, hit rate of TWIF increased significantly as much as 91.66% but in the price of high transfer cost incurred for transferring data among the sites of the distributed system using communication network. Another thing is to mention that MMF achieves a steady hit rate over 80% and TWIF's performance falls gradually from 33.33% to 20% with the increase of a number of sites of DBDS from three to five. I have increased the number of sites in the system up to ten and found similar results.

## 7 Conclusions

Making proper fragmentation of the relations and allocation of the fragments is a major research area in distributed systems. Many techniques have been proposed by the researchers using empirical knowledge of data access by different queries and frequencies of queries executed in different sites of a distributed system. But proper fragmentation and allocation at the initial stage of a distributed database have not yet been addressed. In this paper, I 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. Instead of using empirical data, I have developed a matrix namely Modified Create, Read, Update and

Delete (MCRUD) to make fragmentation decisions. 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 the performance of a DDBMS can be improved significantly by avoiding frequent remote access and high data transfer among the sites.

**Acknowledgment**
The Author is grateful to Dr. Abu Sayed Md. Latiful Hoque, Professor, Dept. of Computer Science and Engineering, Bangladesh University of Engineering and Technology (BUET) for his guidance.

**References**

[1] Ozsu, M. T., and Valduriez, P. Principles of Distributed Database Systems. Prentice-Hall, New Jersey, 1999.
[2] Ceri, S., and Pelagatti, G. Distributed Databases Principles and System. McGraw- Hill, New York, 1984.
[3] Navathe, S., Karlapalem, K., and Ra, M. A mixed fragmentation methodology for initial distributed database design. Journal of Computer and Software Engineering 3, 4 (1995), 395–426.
[4] Bai˜ ao, F., Mattoso, M., and Zaverucha, G. A distribution design methodology for object dbms. Distributed and Parallel Databases 16, 1 (2004), 45–90.
[5] Tamhankar, A. M., and Ram, S. Database fragmentation and allocation: An integrated methodology and case study. IEEE Transactions on Systems Management 28, 3 (1998), 194–207.
[6] Blankinship, R., Hevner, A. R., and Yao, S. B. An iterative method for distributed database design. In Proceedings of the 17th International Conference on Very Large Data Bases (1991), G. M. Lohman, A. Sernadas, and R. Camps, Eds., Morgan Kaufmann, pp. 389–400.
[7] Ma, H. "Distribution design for complex value databases" Doctoral thesis, Department of Information Systems, Massey University, 2007.
[8] Ceri, S., Negri, M., and Pelagatti, G. Horizontal data partitioning in database design. In Proceedings of the 1982 ACM SIGMOD international conference on Management of data (1982), ACM Press, pp. 128–136.
[9] Zhang, Y. On horizontal fragmentation of distributed database design. In Advances in Database Research (1993), M. Orlowska and M. Papazoglou, Eds., World Scientific Publishing, pp. 121–130.
[10] Ra, M. Horizontal partitioning for distributed database design. In Advances in Database Research (1993), M. Orlowska and M. Papazoglou, Eds., World Scientific Publishing, pp. 101–120.
[11] Cheng, C.-H., Lee, W.-K., and Wong, K.-F. A genetic algorithm-based clustering approach for database partitioning. IEEE Transactions on Systems, Man, and Cybernetics, Part C 32, 3 (2002), 215–230.
[12] Mahboubi H. and Darmont J. "Enhancing XML Data Warehouse Query Performance by Fragmentation" ACM SAC, 2009, pp.1555-1562.
[13] Chang, S.-K., and Cheng, W.-H. A methodology for structured database decomposition. IEEE Transactions on Software Engineering (TSE) 6, 2 (1980), 205–218.
[14] Shin, D. G., and Irani, K. B. Partitioning a relational database horizontally using a knowledge-based approach. SIGMOD Record 14, 4 (1985), 95–105.
[15] Shin, D.-G., and Irani, K. B. Fragmenting relations horizontally using a knowledgebased approach. IEEE Transactions on Software Engineering (TSE) 17, 9 (1991), 872–883.
[16] Khalil, N., Eid, D., and Khair, M. Availability and reliability issues in distributed databases using optimal horizontal fragmentation. In Database and Expert Systems Applications (DEXA) (1999), T. J. M. Bench-Capon, G. Soda, and A. M. Tjoa, Eds.,vol. 1677

of Lecture Notes in Computer Science, Springer, pp. 771–780.

[17] Hoffer, J. A., and Severance, D. G. The use of cluster analysis in physical database design. In Proceedings of the First International Conference on Very Large Data Bases (VLDB) (1975), pp. 69–86.

[18] Navathe, S. B., Ceri, S., Wiederhold, G., and Dour, J. Vertical partitioning algorithms for database design. ACM Transactions on Database Systems (TODS) 9, 4 (1984), 680–710.

[19] Navathe, S. B., and Ra, M. Vertical partitioning for database design: A graphical algorithm. SIGMOD Record 14, 4 (1989), 440–450.

[20] Lin, X., and Zhang, Y. A new graphical method of vertical partitioning in database design. In Proceedings of the 4th Australian Database Conference (ADC) (1993), M. P. P. Maria E. Orlowska, Ed., World Scientific, pp. 131–144.

[21] Ma, H., Schewe, K.-D., and Kirchberg, M. A heuristic approach to vertical fragmentation incorporating query information. In Proceedings of the 7th International Baltic Conference on Databases and Information Systems (DB&IS), IEEE Computer Society Press, 2006, pp. 69–76.

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

[23] Ngo T.H., "New Objective Function for Vertical Partitioning in Database System" In Proceedings of the SYRCODIS, 2008.

[24] Runceanu A. "Fragmentation in Distributed Databases", Innovations and Advanced Techniques in Systems, Computing Sciences and Software Engineering, Springer, 2008, pp. 57–62.

[25] Cornell, D., and Yu, P. A vertical partitioning algorithm for relational databases. In International Conference on Data Engineering (1987), pp. 30–35.

[26] Cornell, D. W., and Yu, P. S. An effective approach to vertical partitioning for physical design of relational databases. IEEE Transactions on Software Engineering 16, 2 (1990), 248–258.

[27] Chu, P.-C. A transaction oriented approach to attribute partitioning. Information Systems 17, 4 (1992), 329–342.

[28] Chakravarthy, S., Muthuraj, J., Varadarajan, R., and Navathe, S. B. An objective function for vertically partitioning relations in distributed databases and its analysis. Distributed and Parallel Databases 2, 2 (1994), 183–207.

[29] Son, J. H., and Kim, M. H. An adaptable vertical partitioning method in distributed systems. Journal of Systems and Software 73, 3 (2004), 551–561.

[30] Abuelyaman E. S. "An Optimized Scheme for Vertical Partitioning of a Distributed Database" International Journal of Computer Science and Network Security, VOL.8 No.1, January 2008, pp 310-316.

[31] Raouf, A. E. A., Badr, N. L., & Tolba, M. F. (2014). Dynamic distributed database over cloud environment. In International Conference on Advanced Machine Learning Technologies and Applications (pp. 67-76). Springer International Publishing.

[32] Rodríguez-Mazahua, L., Alor-Hernández, G., Abud-Figueroa, M. A., & Peláez-Camarena, S. G. (2014). Horizontal Partitioning of Multimedia Databases Using Hierarchical Agglomerative Clustering. In Mexican International Conference on Artificial Intelligence (pp. 296-309). Springer International Publishing.

[33] Harikumar, S., & Ramachandran, R. (2015). Hybridized fragmentation of very large databases using clustering. In Signal Processing, Informatics, Communication and Energy Systems (SPICES), 2015 IEEE International Conference on (pp. 1-5). IEEE.

[34] Hababeh, I., Khalil, I., & Khreishah, A. (2015). Designing high performance web-based computing services to promote telemedicine database management system. IEEE Transactions on Services Computing, 8(1), 47-64.
[35] Hess, H. (2016). Evaluating Domain-Driven Design for Refactoring Existing Information Systems (Doctoral dissertation, Ulm University).
[36] Bai˜ ao, F., and Mattoso, M. A mixed fragmentation algorithm for distributed object oriented databases. In Proceedings of the International Conference Computing and Information (ICCI) (1998), pp. 141–148.
[37] Chu, W. W. Optimal file allocation in a multiple computer system. IEEE Transactions on Computers 18, 10 (1969), 885–889.
[38] Casey, R. G. Allocation of copies of files in an information network. In Proceedings of AFIPS SJCC (1972), vol. 40, AFIPS Press, pp. 617–625.
[39] Mahmoud, S., and Riordon, J. S. Optimal allocation of resources in distributed information networks. ACM Transactions on Database Systems (TODS) 1, 1 (1976), 66–78.
[40] Ahmad, I., Karlapalem, K., Kwok, Y.-K., and So, S.-K. Evolutionary algorithms for allocating data in distributed database systems. Distributed Parallel Databases 11, 1 (2002), 5–32.
[41] Menon, M.-S. Allocating fragments in distributed databases. IEEE Transactions on Parallel and Distributed Systems 16, 7 (2005), 577–585.
[42] Surmsuk P. "The integrated strategic information system planning methodology", IEEE Computer Society Press, 2007, pp. 467-475.
[43] Whitten J. et al. "Systems Analysis and Design Methods", 6th Ed. McGraw-Hill, 2004.
[44] Wiegers K. E. Software Requirements, 2nd Edition, Microsoft Publication, 2003.
[45] Khan, S. I., and Hoque, A. S. M. L. (2010). A new technique for database fragmentation in distributed systems. International Journal of Computer Applications, 5(9), 20-24.
[46] Khan, S. I., and Hoque, A. S.M. L. (2012). Scalability and performance analysis of CRUD matrix based fragmentation technique for distributed database. In Computer and Information Technology (ICCIT), 2012 15th International Conference on (pp. 567-562). IEEE.
[47] Pareto principle, accessed from http://www. en.wikipedia.org/wiki/Pareto_principle.
[48] Craig S. Mullins "Defining Database Performance", accessed from http://www.craigsmullins.com/cnr_db.htm.
[49] Fritchey G. and Dam S. "SQL Server 2008 Query Performance Tuning Distilled", 1st Ed. Apress, 2009.

Shahidul Islam Khan obtained his B.Sc. and M.Sc. Engineering Degree in Computer Science and Engineering (CSE) from Ahsanullah University of Science and Technology (AUST) and Bangladesh University of Engineering and Technology (BUET), Dhaka, Bangladesh in 2003 and 2011. He is now a Ph.D. candidate in the Department of CSE, BUET, which is the highest ranked technical university of Bangladesh. His current fields of research are database systems, data mining, and health informatics. He has twenty published papers in referred journals and conferences. He is also an Associate Professor (Currently in study leave) in the Dept. of CSE, International Islamic University Chittagong (IIUC), Bangladesh.