A Novel Technique for Initial Fragmentation of Distributed Database

Shahidul Islam Khan, Bangladesh University of Engineering and Technology
Abu Sayed Md. Latiful Hoque, Bangladesh University of Engineering and Technology

Available at: https://works.bepress.com/shahidulislam-khan/11/
A Novel Technique for Initial Fragmentation of Distributed Database

Shahidul Islam Khan¹, A. S. M. Latiful Hoque²

¹Department of CSE, International Islamic University Chittagong (IIUC), Chittagong, Bangladesh
²Department of CSE, Bangladesh University of Engineering & Technology (BUET), Dhaka, Bangladesh
E-mail: nayeemkh@yahoo.com, asmlatifulhoque@cse.buet.ac.bd

ABSTRACT
An efficient way of improving performance of a database management system is distributed processing. Distribution of data involves fragmentation, replication and allocation process. Previous research works provided fragmentation solution based on empirical data which are not applicable at initial stage of distributed database. 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. Experimental results show that our proposed technique can solve the initial fragmentation problem properly.

KEYWORDS: Distributed system, Initial fragmentation, Allocation, MCRUD matrix, Attribute locality precedence

I. INTRODUCTION
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. 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. Design of efficient distributed databases is one of the major research problems. Primary concern of distributed database design is to make the fragmentation of relations, allocation of the fragments in different sites of the system, and local optimization in sites [1], [2].

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]. Fragmentation can be horizontal, vertical or mixed/hybrid. Horizontal fragmentation (HF) allows a relation or class to be partitioned into disjoint tuples. Vertical fragmentation (VF) partitioned a relation or class into disjoint sets of columns or attributes except the primary key [1], [2]. Combination of horizontal and vertical fragmentations to form 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.

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 a system [1]-[4]. Combined problem of fragmentation and allocation is NP-hard [5]. HF using minterm predicate is first proposed by Ceri et al. (1982) [6]. Navathe et al. (1984) used attribute usage matrix (AUM) and Bond energy algorithm to produce vertical fragments [7]. Navathe and Ra (1989) improved the previous work on VF by proposing an algorithm using a graphical technique [8]. Shin and Irani (1991) proposed knowledge based approach in which user reference clusters are derived from the user queries [9]. Ra (1993) presented a graph based algorithm for HF in which predicates are clustered based on the predicate affinities [10]. Navathe et al. (1995) first proposed MF technique[3]. Ozsu and Valduriez (1999) proposed an iterative algorithm COMMIN to generate a complete and minimal set of predicates from simple predicates [1].


In this paper we have presented a novel technique for horizontal fragmentation of the relations of a distributed relational database that is capable of taking proper fragmentation decision at initial stage. The work of this paper is to extend our work in [16].

The rest of this paper is organized as follows. In section II we have described our developed technique. Our results and discussion are presented in Section III. Finally Section IV concludes the paper with further research directions.

II. MCRUD MATRIX BASED FRAGMENTATION
To solve the problem of taking proper fragmentation decision at the initial stage of a distributed database, we have developed a new fragmentation technique namely MCRUD Matrix based Fragmentation technique (MMF). Instead of using empirical data, we have developed Modified Create, Read, Update and Delete (MCRUD) matrix to obtain fragmentation decisions. That is to fragment a relation horizontally...
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], [19]. We have modified the existing CRUD matrix according to our requirement of HF and name it 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 [16]. We treated cost as the effort of access and modification of a particular attribute of a relation by an application from a site. To calculate precedence of attributes we took MCRUD matrix as input and use following cost functions:

\[ C_{i,j,k,r} = f_c C + f_R R + f_U U + f_D D \]

\[ S_{i,j,k} = \sum_{r=1}^{A} C_{i,j,k,r} \]  

\[ S_{i,j,m} = \max (S_{i,j,k}) \]  

\[ ALP_{i,j,k} = \sum_{k=1}^{m} S_{i,j,k} \]  

\[ ALP_i = \sum_{j=1}^{A} ALP_{i,j} \]  

Here:  
- \( f_c \) = frequency of create operation  
- \( f_R \) = frequency of read operation  
- \( f_U \) = frequency of update 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 \( k \)

\( S_{i,j,1,k} \) = 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 \( i \)

\( ALP_{i,j,k} \) = actual cost for predicate \( j \) of attribute \( i \)

\( ALP_i \) = total cost of attribute \( i \) (locality precedence)

After construction of ALP table, predicate set \( P \) is generated for the attribute with highest precedence value. Finally each relation is fragmented horizontally using the predicates of \( P \) as selection predicate. The procedures can be understood from the following algorithm of Fig. 2. The space and time complexity of MMF have been discussed in following subsections.

**Input:** Total number of sites: \( S = \{S_1, S_2, ..., S_f\} \)  
Relation to be fragmented: \( R \)  
Modified CRUD matrix: \( \text{MCRUD}[R] \)  

**Output:** Fragments \( F = \{ F_1, F_2, ..., F_n \} \)

**Step 1:** Construct \( ALP[R] \) from \( \text{MCRUD}[R] \) based on Cost functions

**Step 2:** For significant highest valued attribute of \( ALP \) table
  a. Generate predicate set \( P = \{ P_1, P_2, ..., P_n \} \)
  b. Fragment \( R \) using \( P \) as selection predicate

\[ \forall \ P \mid \sigma (P) (R) \]

**Step 3:** For non-significant-highest-value (Max-Highest<1.5*2 \( n \)) in \( ALP[R] \)
  a. REPLICATE \( R \) to \( S \) if \( R \) is an entity set
  b. Derive Horizontally Fragment \( R \) using its owner relation if \( R \) is a relationship set

**Fig. 2. Fragmentation Allocation algorithm**

**A. Memory Cost Analysis**

To store the MCRUD matrices of different relations in the system, we have used four dimensional arrays. For calculating ALP values we have to store four things: site number, application number, attribute number and predicate number. These values will be stored in 4D arrays and ALP tables are constructed using the arrays. Cost of each cell of MCRUD matrix is computed by the cost function of equation 1.

This is represented in pseudo-code of Fig.3 in [17] as:

\[ C[i][j][k][r] = f_c * C + f_R * R + f_U * U + f_D * D. \]

Where \( i \) = attribute number, \( j \) = predicate number, \( k \) = site number and \( r \) = application number. In this thesis number of sites is denoted by \( S \), number of applications is denoted by \( N \), number of Predicates is denoted by \( P \) and number of attributes is denoted \( A \). So space requirement to store an MCRUD matrix will be \( O(S*N*A*P) \) or \( O(n^4) \) if \( S = N = A = P = n \).

**B. Computational Cost Analysis**

Creation of MCRUD matrix for every relation and calculation of ALP from each matrix adds some additional cost in our system. Its maximum
computational cost is dominated by the computation within four nested loops. The code is as follows:

```java
for ( i =1; i <= TotalAttributes; i++)
    for ( j =1; j <= TotalPredicates[i]; j++)
        for ( k =1; k <= TotalSites; k++)
            for ( r =1; r <= TotalApplications[k]; r++)
                C[i][j][k][r] = f_c * C + f_r * R + f_u * U + f_d * D
                S[i][j][k] += C[i][j][k][r]
```

So computational order is $O(i*j*k*r)$. As $i_{max} = A$, $j_{max} = P$, $k_{max} = S$ and $r_{max} = N$, so we can rewrite the order as $O(n^4)$ if we treat $S \approx N \approx A \approx P \approx n$. Actual problem of horizontal fragmentation and allocation is $O(n^k)$ where there are $n$ simple predicates and $k$ sites. This is impossible in practical large database systems [8]. We have reduced it to $O(n^4)$ by providing solution based on heuristic that use MCRUD matrix. Our cost can be ignored because ALP calculation from MCRUD matrix will be performed offline during analysis phase.

III. Result and Discussion

We have implemented a distributed banking database system in the post-graduate lab of BUET using three DELL computers each with Core-two Duo 2.80 processors and 2GB RAM. We have used Oracle 10g for database creation. Initially number of sites of the distributed system is three as shown in Fig.3.

![Distributed banking database system](image)

**Figure 3. Distributed banking database system**

A. Construction of MCRUD Matrix

We have constructed the MCRUD matrix for each relations in the requirement analysis phase.

B. Calculation of ALP values

We have calculated locality precedence of each attribute from the MCRUD matrix. Using the ALP values we have constructed ALP table for each relation. Detail calculation is explained in [16], [17].

C. Generation of Predicate Set & Fragmentation

Predicate set was generated for the attributes with highest locality precedence of all relations respectively. According to predicate set, fragmentations of all relations were performed.

We have executed twenty queries in each site with a total of sixty selected queries in the distributed system according to Pareto Principle [20] to see the performance of MMF.

The queries were selected from following query domain:

- **Insertion** e.g. Insert into RRR values (xxx, yyy, zzz)
- **Selection (Point)** e.g. Select A1, A2, A3 from RRR where xxx = P
- **Selection (Range)** e.g. Select A1, A2... An from RRR where xxx < BBB
- **Selection (Join)** e.g. Select A1, A2... An from R1, R2 where R1.A1=R2.A1 AND R1.A2 = CCC
- **Selection (Aggregation)** e.g. Select Sum (AA) from RRR
- **Update** e.g. Update RRR set A1 = xxx where A2 = yyy
- **Deletion** e.g. Delete * from RRR where P

We define hit as a result of a query of any type accessed records of 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. Table I shows the overall performance of the distributed system after fragmenting the relations using MMF technique.

![Table I: Overall System Performance of MMF](image)

We 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] – [12], [14] - [15]. TWIF uses the model in general as shown in Fig. 4.

![Figure 4 Model of Non-initial Fragmentation Techniques (TWIF)](image)
comparison with our achieved 85% hit rate. The reason of poor performance of all other fragmentation techniques (TWIF) is that, all sites other than central site have no data. So all queries that are generating in those sites requires remote data access thus scores miss. Comparison of MMF and TWIF by their hit ratio shown in Fig. 5.

Figure 5 Comparison of Hit between MMF & TWIF for 3 Sites

Fig. 6 shows the hit ratio of MMF and TWIF with the increase of number of sites in the distributed system. We can see that MMF shows much better and quite steady performance as sites increases from three to ten. In the same time performance of TWIF falls gradually as new sites are added to the system.

Existing technique that provided a solution of initial fragmentation is StatPart described in [13]. To fragment a relation, it starts with a randomly generated matrix called reflexivity matrix. It always suggests two binary vertical fragments independent of number of sites of the DDBS. So this technique is not suitable for a distributed system with more than two allocation sites. Also as it starts with a random matrix steady fragmentation decision cannot be provided using StatPart algorithm.

IV. 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 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. 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.

REFERENCES