Indian Journal of Science and Technology, *Vol 9(3), DOI: 10.17485/ijst/2016/v9i3/59963, January 2016*

ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645

A New Approach to Re-Fragment in a Distributed Database Environment

Ankita Bihani[1]*, Mudireddy Shruthi Reddy[1] and Aswin Chandrasekharan[2]

[1]School of Computing Science and Engineering, VIT University, Vellore - 632014, Tamil Nadu, India; ankita.bihani@gmail.com, shruthimudireddy@gmail.com
[2]School of Information Technology and Engineering, VIT University, Vellore - 632014, Tamil Nadu, India; kcsaswin@gmail.com

Abstract

The key idea of this paper is to present a novel method for Re-fragmentation in a Distributed Database System that ensures that the Database System is adaptive to the changes in access frequency patterns. Conventionally, fragmentation is done to reduce network transfer cost and communication costs. But with time, changing access patterns at the sites defies the very purpose of fragmentation. In this paper, we have presented an algorithm to maximize and maintain the efficiency of fragmentation in a Distributed Database System. This algorithm takes into account the change in the access patterns and queries; evaluates and quantifies them and performs re-fragmentation periodically according to the updated access frequency patterns. This helps in sustaining the efficiency of fragmentation irrespective of the changing access patterns. The technique used for studying the efficiency of the new algorithm was to employ the new algorithm on a sample Distributed Database system and compare the same with the existing conventional algorithm. The performance of a Distributed Database system is inversely proportional to its communication cost and execution time of queries[1,2]. On analysis of the presented algorithm, we see that the time complexity of this algorithm is linear. Also the total Network Transfer Cost is considerably lesser than the conventional fragmentation algorithm for Distributed Database. The algorithm presented in the paper is applicable to all Distributed Database systems, which have changing access frequency patterns.

Keyword: Access Frequency, LAF Matrix, NLAF Matrix, Network Transfer Cost, Optimization Re-Fragmentation

1. Introduction

In the present scenario of multinational corporations and global business environment, handling large amount of data at one centralized location becomes tedious and inefficient. This calls for an imminent need for a Distributed Database System by fragmenting the database[3]. Fragmentation in its essence is the partitioning of database into disjoint, non-overlapping fragments[4,5]. The process of assigning these fragments to the sites of a distributed network is called Allocation[3,4]. The basic advantage of fragmentation is

Optimization[7,8] i.e., reduced storage costs, access costs and network transfer cost[5,9]. In addition to this, fragmentation increases the degree of concurrency allowing transactions to execute in parallel. Besides, enhanced database security is assured, as data that is not used at a site is not stored at that site allowing access privileges for sensitive data to be restricted to certain users only. The initial fragmentation in a Distributed Database System is based on the access frequencies of queries entered at the user sites. But as time progresses and user requirement changes, the present set of access frequencies can get outdated. When queries frequently need to access fragments that are not local to a site, it results in increased network transfer cost, thereby degrading the performance of the Distributed Database System. This necessitates the need for Re-Fragmentation[2,10]. This paper proposes

**Author for correspondence*

an efficient algorithm to perform Re-fragmentation that takes into account the changing patterns of access frequencies at the various sites in a Distributed Database System.

2. Proposed Algorithm

An algorithm to re-fragment a Distributed Database System is explained in this paper. The database is initially fragmented and stored at several sites based on the initial access frequencies[11,12]. Each fragment has a unique fragment identifier or ID associated with it. A data directory stored at a central site stores this fragment identifier, the set of conditions on the attributes used for fragmentation and the site at which each fragment is present.

For every site, we have two m*3 matrices where 'm' stands for the number of fragments. The two matrices are:

- Local Access Frequency (LAF) matrix M1 that stores the local fragments' ID, the corresponding access frequency and a pseudo-bit and
- Non-Local Access Frequency (NLAF) matrix M2 that stores the ID of fragments that are not present at its local site, the corresponding access frequency and a pseudo bit.

The idea of Re-fragmentation is to periodically check whether the fragmentation is still efficient and that it does not increase the network transfer cost.

Every time a local fragment is accessed, the LAF matrix is updated by incrementing the access frequency while every time a site accesses a fragment that is not present at that site; the NLAF matrix at the site is updated. These values will be reset after every time period T1 (where time period TI can be initialized by the database administrator).

The system waits for 40% of the queries to be executed or for time period T2 (where time period T2 can be initialized by the database administrator), whichever occurs earlier and then the two matrices M1 and M2 are checked. The following four possibilities arise:

- In the LAF matrix, if the access frequency value of one or more fragments is less than a predefined threshold value (which may vary from scenario to scenario and is initialized by the database administrator), then the corresponding pseudo-bit is decremented by 1. After that, for every time period T2, if the access frequency for some or all of those fragments is again found to be less than the threshold value, then the corresponding pseudo bits for those fragments that are less than or equal to 0 are decremented by 1.
- In the LAF matrix, if the access frequency of a fragment is more than the predefined threshold value, the corresponding pseudo-bit remains unchanged.
- In the NLAF matrix, if the access frequency of one or more fragments is greater than a predefined threshold value, then the corresponding pseudo- bit is incremented by 1. After that, for every time period T2, if the access frequency of some or all of those fragments is again found to be greater than the predefined threshold value, then the corresponding pseudo bits that are greater than or equal to 0 are incremented by 1.
- In the NLAF matrix, if the access frequency of the fragments is less than the threshold value, the corresponding pseudo- bits remain unchanged.

In addition to the above periodic system check, if the system gets overloaded at any point of time, i.e. the total size of the fragments stored at a site becomes more than half the storage space available at that site; then the local access frequency matrix at that site is checked. If there are fragments with pseudo-bits less than 0, then it is checked whether any of the sites has pseudo-bits greater than 0 for the same fragment in its NLAF matrix. Three possibilities arise here:

- If a single match is found then the fragment is transferred from its current site to the site with pseudo-bit greater than 0 in the NLAF matrix.
- If multiple matches are found, then the site with the highest pseudo-bit in the NLAF matrix is chosen as the receiving site.
- If there are two sites with the same value of pseudo-bit in the NLAF matrix, then the fragment can be transferred to either of them.

The above procedure is repeated at all the sites. The flowchart in Figure 1 explains the algorithm.

3. Illustrative Example

Let us consider a simple database with the following relations:

Emp [Eno, Ename, Deptno] as shown in Table 1

Dept [Dno, Dname, Dloc] as shown in Table 2

The fragmentation is based on the following clauses:

Table 1. EMP Table

ENO	ENAME	DEPTNO
1	Alice	1
2	Joe	1
3	Bob	1
4	Jimmy	2
5	Elena	2
6	George	2
7	Alex	3
8	Patrick	3
9	Albert	3
10	Denny	4
11	Veronica	4
12	Daisy	4
13	Harriet	5
14	Tom	5
15	Sophie	5

Table 2. Dept Table

DNO	DNAME	DLOC
1	Research	London
2	Finance	Japan
3	Testing	Australia
4	Purchase	Malaysia
5	Sales	Hyderabad

Fragment 1 stores the Emp and Dept tuples corresponding to the query Q1:

SELECT * from Emp, Dept WHERE Eno <=3 AND Dloc="London"

Fragment 2 stores the Emp and Dept tuples corresponding to the query Q2:

SELECT * from Emp, Dept WHERE Eno BETWEEN 4 AND 6 AND Dloc="Japan"

Fragment 3 stores the Emp and Dept tuples corresponding to the query Q3:

SELECT * from Emp, Dept WHERE Eno BETWEEN 7 AND 9 AND Dloc="Australia"

Fragment 4 stores the Emp and Dept tuples corresponding to the query Q4:

SELECT * from Emp, Dept WHERE Eno BETWEEN 10 AND 12 AND Dloc="Malaysia"

Fragment 5 stores the Emp and Dept tuples corresponding to the query Q5:

SELECT * from Emp, Dept WHERE Eno >=13 AND Dloc="Hyderabad"

It is assumed that there are 3 sites S1, S2 and S3.

At S1, tuples corresponding to the fragment 1 and fragment 2 are stored as shown in Table 3 and Table 4 respectively.

Table 3. Fragment 1

ENO	ENAME	DEPTNO	DNAME	LOC
1	Alice	1	Research	London
2	Joe	1	Research	London
3	Bob	1	Research	London

Table 4. Fragment 2

ENO	ENAME	DEPTNO	DNAME	LOC
4	Jimmy	2	Finance	Japan
5	Elena	2	Finance	Japan
6	George	2	Finance	Japan

At S2, tuples corresponding to the fragment 3 and fragment 4 are stored as shown in Table 5 and Table 6 respectively.

Table 5. Fragment 3

ENO	ENAME	DEPTNO	DNAME	LOC
7	Alex	3	Testing	Australia
8	Patrick	3	Testing	Australia
9	Albert	3	Testing	Australia

Table 6. Fragment 4

ENO	ENAME	DEPTNO	DNAME	LOC
10	Denny	4	Purchase	Malaysia
11	Veronica	4	Purchase	Malaysia
12	Daisy	4	Purchase	Malaysia

At S3, tuples corresponding to the fragment 5 are stored as shown in Table 7.

Table 7. Fragment 5

ENO	ENAME	DEPTNO	DNAME	LOC
13	Harriet	5	Sales	Hyderabad
14	Tom	5	Sales	Hyderabad
15	Sophie	5	Sales	Hyderabad

Initially, all the local and non-local access frequency matrices are initialized with the fragment ID, their corresponding access frequencies and the pseudo bits

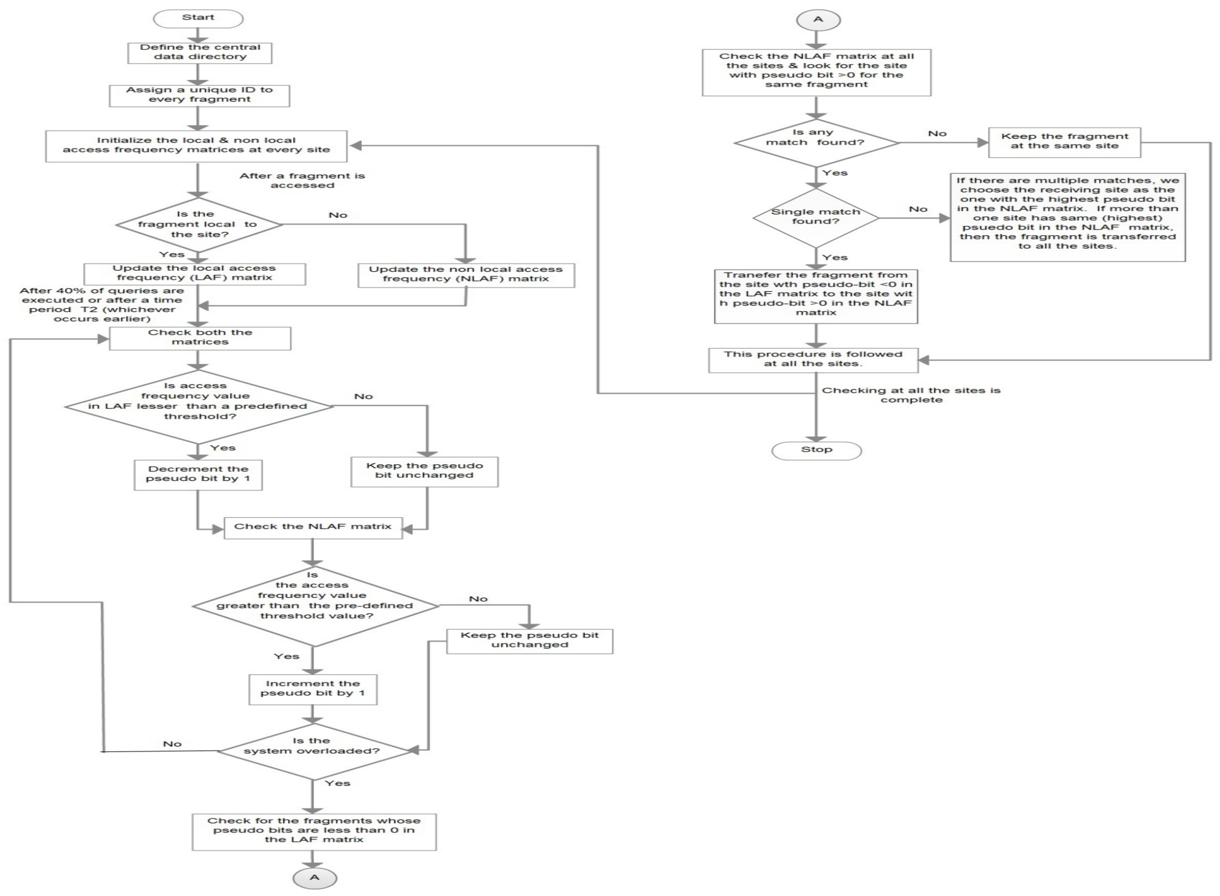
Start
Define the central data directory
Assign a unique ID to every fragment
Initialize the local & non local access frequency matrices at every site
After a fragment is accessed
Is the fragment local to the site?
No
Yes
Update the local access frequency (LAF) matrix
Update the non local access frequency (NLAF) matrix
After 40% of queries are executed or after a time period T2 (whichever occurs earlier)
Check both the matrices
Is access frequency value in LAF lesser than a predefined threshold?
No
Yes
Decrement the pseudo bit by 1
Keep the pseudo bit unchanged
Check the NLAF matrix
Is the access frequency value greater than the pre-defined threshold value?
No
Keep the pseudo bit unchanged
Yes
Increment the pseudo bit by 1
Is the system overloaded?
No
Yes
Check for the fragments whose pseudo bits are less than 0 in the LAF matrix
A
A
Check the NLAF matrix at all the sites & look for the site with pseudo bit >0 for the same fragment
Is any match found?
No
Keep the fragment at the same site
Yes
Single match found?
No
If there are multiple matches, we choose the receiving site as the one with the highest pseudo bit in the NLAF matrix. If more than one site has same (highest) psuedo bit in the NLAF matrix, then the fragment is transferred to all the sites.
Yes
Tranefer the fragment from the site wth pseudo-bit <0 in the LAF matrix to the site wit h pseudo-bit >0 in the NLAF matrix
This procedure is followed at all the sites.
Checking at all the sites is complete
Stop

Figure 1. Flow chart of the proposed algorithm.

(initialized to zero). Each fragment is allocated to one site only. However, a site may have more than one fragment.

Let us consider the predefined threshold to be equal to 10. As mentioned in the algorithm, the central directory stores the fragment details, which include the fragment id, the set of predicates, which define the fragment and the site which stores the fragment.

The central directory initially stores the data in Table 8.

Table 8. Initial central data directory

Fragment id	Site no	Predicate set
1	2	Eno <=3 and Dloc="London"
2	1	Eno >3 and Eno<=6 and Dloc="Japan"
3	2	Eno >6and Eno<=9 and Dloc="Australia"
4	3	Eno >9 and Eno<=12 and Dloc="Malaysia"
5	1	Eno >13 and Dloc="Hyderabad"

After time period T2 or when 40 % of the queries are executed, let us assume that the matrices have the following values:

At S1, the local and non-local access frequency matrices are as follows:

LAF:

$$\begin{bmatrix} 1 & 8 & -4 \\ 2 & 12 & 0 \end{bmatrix}$$

NLAF:

$$\begin{bmatrix} 3 & 7 & 0 \\ 4 & 6 & 6 \\ 5 & 15 & 7 \end{bmatrix}$$

At S2, the local and non-local access frequency matrices are as follows:

LAF:

$$\begin{bmatrix} 3 & 11 & 0 \\ 4 & 6 & -3 \end{bmatrix}$$

NLAF:

$$\begin{bmatrix} 1 & 14 & 3 \\ 2 & 6 & 0 \\ 5 & 13 & 2 \end{bmatrix}$$

At S3, the local and non-local access frequency matrices are as follows:
LAF:

$$\begin{bmatrix} 5 & 4 & -2 \end{bmatrix}$$

NLAF:

$$\begin{bmatrix} 1 & 4 & 0 \\ 2 & 16 & 4 \\ 3 & 7 & 0 \\ 4 & 15 & 6 \end{bmatrix}$$

The check on site S1is started.

The pseudo bit of fragment 1 in the LAF matrix at site 1 is -4 [<0]. So, all the sites are checked one by one to find the site where the pseudo bit of fragment 1 in NLAF matrix is >0. Such a match is found at site 2.Thus fragment 1 is transferred from site 1 to site 2.

The pseudo bit of fragment 2 in the LAF matrix at site 1 is 0. Hence, no further checking is done.

Now, proceeding to site 2:

The pseudo bit of fragment 3 in the LAF matrix at site 2 is 0. Hence, no further checking is done.

The pseudo bit of fragment 4 in the LAF matrix at site 2 is -3 [<0]. So, all the sites are checked to find where the pseudo bit of fragment 4 in NLAF matrix is >0.Such a match is found at sites 3 and 1.The pseudo bit value at both sites is the same. Thus fragment 4 is transferred from site 2 to either site 3 or 1. Let us assume that it is transferred to site 3.

Now, proceeding to site 3:

The pseudo bit of fragment 5 in the LAF matrix at site 3 is -2 [<0]. So, all the sites are checked to find out where the pseudo bit of fragment 5 in NLAF matrix is >0.Such a match is found at sites 1 and 2. Now, the pseudo bits in the NLAF matrices at sites 1 and 2 are compared. Site 1 has a greater value of the pseudo-bit. Thus fragment 5 is transferred from site 3 to site 1.

Thus the final distribution after re-fragmentation is as under:

Site 1 has the fragments 2 and 5

Site 2 has the fragments 1 and 3.

Site 3 has the fragment 4.

The same procedure repeats periodically after a set time interval T.

The central data directory is updated as given in Table 9.

Table 9. Updated central data directory

Fragment id	Site no	Predicate set
1	1	Eno <=3 and Dloc=”London”
2	1	Eno >3 and Eno<=6 and Dloc=”Japan”
3	2	Eno >6and Eno<=9 and Dloc=”Australia”
4	2	Eno >9 and Eno<=12 and Dloc=”Malaysia”
5	3	Eno >=13 and Dloc=”Hyderabad”

4. Analysis of the Algorithm

The complexity analysis of the proposed algorithm is given as under:

Let us assume we have K sites, and N fragments in a Distributed Database System.

4.1 Time Complexity of Construction and Initialization of LAF, NLAF at each site and the Central Data Directory

The LAF and NLAF matrices are implemented using Hash Tables at every site. The Fragment-id is taken as the key and the pair of access frequency and Pseudo-bit as the value.

At each site, inserting the details of all the *N* fragments into the Hash Tables (Local Access Frequency matrix and Non-local access frequency matrix) will require O(N) time where ‘*N*’ is the total number of fragments.

The central Data Directory is also implemented using a Hash Table. Hence, the time complexity of construction of this Hash Table will also be O(N) where ‘*N*’ is the total number of fragments.

4.2 Time Complexity of Periodic Update of Pseudo-Bits in the LAF and NLAF Matrices at each site

At every site, each fragment’s access frequency is compared with the predefined threshold frequency. Based on this comparison, the pseudo bit is changed in accordance with the algorithm.

Hence, complexity of periodically updating the pseudo-bits will be O(N) where '*N*' is the total number of fragments at each site.

4.3 Time Complexity of Transferring Fragments from one Site to Another

Assuming there are *Y* fragments present at a site, the number of fragments in the LAF matrix will be Y. The pseudo-bits of all the *Y* fragments will be checked, which takes O(Y) time.

If the pseudo-bit of one or more fragments is found to be less than 0, then the NLAF matrix at each site will be checked. This takes O(K *(N-Y)) time, if (N-Y) fragments are assumed to be present at each site in the NLAF Hash Table.

The time complexity of the algorithm is shown in the Figure 2. The x-axis represents the time (in minutes) while the y-axis represents the number of fragments (in unit numbers). A linear relationship is seen between the two; thereby the algorithm has linear time complexity.

Once, the match is found for transfer (according to the algorithm), the particular fragment is transferred. Let the Network Transfer Cost per transfer be C.

After the transfer is complete, the Central Data Directory needs to be updated. This will take O(P) time, assuming P is the number of transfers. Therefore, the following two equations are obtained:

$$T(N) = O(Y) + O(K*(N-Y)) + O(P) \tag{1}$$

$$C_{total} = C*P \tag{2}$$

where T(N) is the Total time Complexity of transferring fragments from one site to another,

C_{total} is the Total Network transfer cost,

Y is the number of fragments at a site,

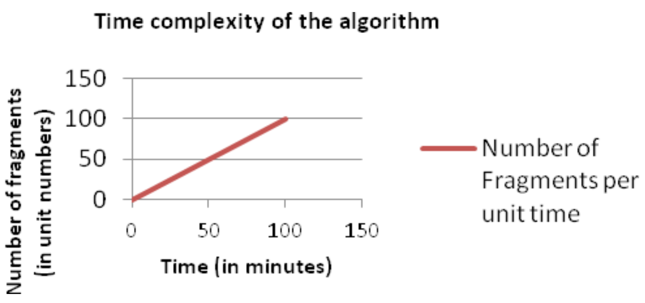

Figure 2. Time complexity of the algorithm.

K is the number of sites,

N is the total number of fragments in the Distributed Database System,

P is the number of transfers and

C is the Network Transfer cost per transfer

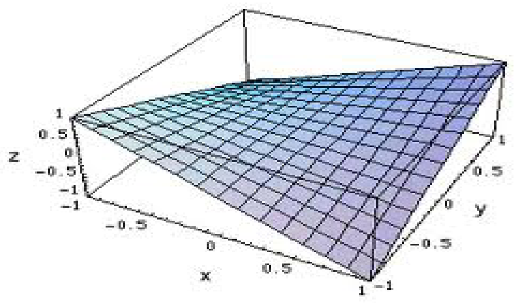

Figure 3. Total network transfer cost as a function of unit network transfer cost and number of fragments.

In Figure 3, x axis represents the number of fragments (in unit numbers), y axis represents the Unit Network Transfer Cost and z axis represents the Total Network Transfer Cost. Hence, we see that the time complexity of this algorithm is linear. We see that the total Network Transfer Cost in this case is considerably lesser than the scenario where the fragmentation becomes unusable with time and thus increases the network cost tremendously.

5. Conclusion

In this paper, an idea to maximize and maintain the efficiency of fragmentation has been presented. The basic purpose of fragmentation is to reduce the transfer and communication costs. However, after a certain time period, the access patterns and queries by the sites might not remain the same. The old patterns thus become ineffectual which in turn increases the communication cost that defies the whole purpose of fragmentation itself. Thus, by taking into account the updated access patterns and queries, the database is re-fragmented periodically according to the latest users' site requirements. This sustains the efficiency of fragmentation irrespective of the changing access patterns.

Table 10 shows the comparison of the existing Distributed Database systems with the proposed solution:

6. Future Scope

In the algorithm that has been proposed in this paper, sthe unused fragments whose pseudo bits are negative

Table 10. Comparative study of the existing system and proposed solution

Basis of comparison	Existing method	Proposed method
Usability of the system	With time, the access patterns may change which makes the fragmentation unusable.	Re-fragmentation is done periodically based on the current access frequencies to keep the system updated and usable.
Network transfer costs	As the access patterns change, the access of the fragments that are not local to a site may increase. These in turn cause an increase in the network transfer costs, thereby increasing the total cost.	After Re-fragmentation is done based on the current access frequencies, the network transfer costs will greatly reduce.
Complexity	Fragmentation is done based on initial parameters after which there is no further checking and/or reallocation. Hence the system is simple to implement	Checking is done periodically at all the sites and a central directory needs to be updated regularly. However, the complexity for the same is linear hence it is feasible to implement this solution
Performance	The system is not adaptive to changes in access patterns. Hence performance of the system degrades with time.	The system is adaptive to changes in access patterns. Hence the performance of the system is maintained.
Scalability	New user sites need to be allocated fragments explicitly and do not have learning capability	New user sites need not be allocated any fragments in the beginning and will acquire them through the use of this algorithm

in the local access frequency matrix and do not have a demand elsewhere either (which means none of the sites have positive pseudo bits for the same fragment in their non-local access frequency matrix), continue to remain in the same fragment. This increases the load on the site (with a lot of unused fragments). A further enhancement to the proposed algorithm can be made by gathering all the fragments with negative pseudo bits into a dedicated central archive that stores all the unused fragments and is also monitored by the central data directory. If newer access patterns then demand for the fragments from this archive, it could be reallocated using the re-fragmentation technique proposed above.

7. Acknowledgments

The authors would like to thank Prof. Sharmila Banu K for providing constructive criticism throughout the research work. She has been a constant source of support, guidance and motivation throughout the work. In addition, the authors would also like to thank Supriya Moond, Shravya Thandra, Prince Hemrajani and Pranusha Byreddy for their invaluable inputs during the research work.

8. References

1. Henver AR, Yao SB. Query Processing in a Distributed Database System. IEEE Transactions on Software Engineering. 1979 May; SE-5(3).
2. Rababaah H. Distributed Databases fundamentals and research. Advanced Database–B561. Spring. 2005.
3. Sleit A, AlMobaideen W, Al-Areqi S, Yahya A. A dynamic object fragmentation and replication algorithm in distributed database systems. American Journal of Applied Sciences. 2007; 4(8):613–8.
4. Tamer eOzsu M, Valduriez P. Principles of Distributed Database System, 3rd edition.
5. Bhardwaj A, Singh A, Kaur P, Singh B. Role of Fragmentation in Distributed database system. International Journal of Networking and Parallel Computing. 2012 Sep; 1(1).
6. Huang Y-F, Chen J-H. Fragment Allocation in Distributed Database Design. Journal of Information Science and Engineering.2001; 17:491–506
7. Bhuyar PR, Gawande AD. Distributed Database: Fragmentation and Allocation. Journal of Data Mining and Knowledge Discovery. 2012; 3(1):58–64. ISSN: 2229–6662, ISSN: 2229–6670.
8. Ciobanu NM. Algorithm for Dynamic Partitioning and Reallocation of Fragments in a Distributed Database. International Journal of Computer Science. 3.
9. Brown KP, Mehta M, Carey MJ, Livny M. Towards automated performance tuning for complex workloads. In VLDB. 1994; 94:72–84.
10. Kumar M, Batra N, Aggarwal H. Cache Based Query Optimization Approach in Distributed Database. International Journal of Computer Science Issues (IJCSI). 2012; 9(6).

11. Khan SI, Latiful Hoque ASM. A New Technique for Database Fragmentation in Distribuated Systems. International Journal of Computer Applications. 2010 Aug; 5(9). (0975 – 8887).
12. Gupta S, Panda S. Vertical ragmentation, Allocation and Re-Fragmentation in Distributed Object Relational Database Systems-(Update Queries Included). International Journal of Engineering Research and Development. 2012 Nov; 4(7):45–52. e-ISSN: 2278-067X, p-ISSN: 2278-800X, Available from: www.ijerd.com