Efficient Fragmentation and Allocation in Distributed Databases

A. Suganya
II M.E, Computer Science and Engineering, Sri Shakthi Institute of Engineering and Technology, Coimbatore, India.

R. Kalaiselvi
Assistant Professor, Department of Computer Science and Engineering, Sri Shakthi Institute of Engineering and Technology, Coimbatore, India.

Abstract

An efficient functionality of any distributed database system is highly dependent on its proper design in terms of adopted fragmentation and allocation methods. Fragmentations of large, global databases are performed by dividing the database horizontally, vertically or combination of both. In order to enable the distributed database systems to work efficiently, the fragments have to be allocated across the available sites in such a way that reduces communication cost of data. A method of clustering sites is proposed where the sites which are nearer such that they have low cost among them for communication are grouped as one cluster and the fragments are allocated to the cluster. Also the static allocation of fragments provides only the limited response to the changes in workload. Hence dynamic methods are adopted for fragmentation of both structured and unstructured databases. This reduces the movement of data and also improves the overall system performance.

Keywords—Distributed database, fragmentation, allocation of fragments, cluster of sites.

1. Introduction

Distributed database systems comprise a single logical database that is partioned and distributed across various sites in a communication network. Database technology has become prevalent in most business organizations. Distributed Database System (DDS) are becoming more affordable and useful. A DDS typically consist of a number of distinct yet interrelated databases (fragments) located at different geographic sites which can communicate through a network. Typically, such a system is managed by a distributed database management system (DDBMS). Each site of the DDS has its own hardware and is capable of autonomous operation. A site participates in the execution of global transactions involving databases at two or more remote sites.

Designing distributed database systems is fairly complex task because it involves several interacting design decisions.
The primary decisions are as follows.

- Fragmentation: A single database needs to be divided into two or more pieces such that the combination of the pieces yields the original database without any loss of information. Each resulting piece is known as a database fragment.
- Allocation: Each fragment must be allocated to a location in the distributed environment such that the system functions effectively and efficiently.
- Replication: Copies of a fragment must be allocated to other locations in the distributed environment to enhance system performance. Too many copies of a fragment tend to slow down updates while enhancing the performance of read-only queries. Too few copies of a fragment will decrease the availability of data and the performance of read-only queries.
- Concurrency control: Appropriate techniques are required to synchronize the various copies of a fragment. These techniques must take into account of the requirements on the concurrency of data held in the copies and the existence of multiple users.
- Query processing: Since a query may access multiple fragments, and since each fragment may have multiple copies, query optimization becomes an important issue.

In addition, there are other design decisions such as the configuration of the network connecting the database sites, allocation of storage capacity and security. Many of the decisions outlined above are not independent of each other. For example, fragmentation and allocation are very closely related, with each decision affecting the other. Fragmentation and allocation typically use similar input parameters (e.g., a description of user queries, updates, data access frequencies, communication cost, and relationships among data objects). In distributed databases, the communication costs can be reduced by partitioning database tables horizontally into fragments, and allocating these fragments to the sites where they are most frequently accessed. The aim is to make most data accesses local, and avoid remote reads and writes. The read cost can be further reduced by the replication of fragments when beneficial. Obviously, important challenges in fragmentation and replication are how to fragment, when to replicate fragments, and how to allocate the (replicated) fragments.

Previous works on data allocation has focused on static fragmentation based on analyzing queries. These techniques are only useful in contexts where read queries dominate. However, in many application areas, workloads are very dynamic with frequent changes in access patterns at different sites. One common reason for this is that their data usage often consists of two separate phases: a first phase where writing of data dominates (for instance during simulation when results are written), and a subsequent second phase when a subset of the data, for example results, is mostly read. The dynamism of the overall access pattern is further increased by the different instances of the applications executing in different phases at different sites. Because of dynamic workloads, static/manual fragmentation and replication may not always be optimal. Instead, the fragment and replication management should be dynamic and automatic i.e., change in access patterns should result in refragmentation and reallocation of fragments when beneficial, as well as in the creation or removal of fragment replicas.

The primary concern of this paper describes the approach to perform fragmentation of structured data and the secondary concern is to fragment an unstructured data. Furthermore allocation of fragments to the cluster of sites is carried out in order to reduce the communication cost.

The rest of the paper is organized as follows: Section 2 describes the fragmentation concept on structured data. Section 3 describes the fragmentation concept on unstructured data. Section 4 describes the allocation of fragments to cluster of sites rather than allocating to individual sites. Section 5 describes some concluding remarks of the paper.

2. Fragmentation on structured data

A. Architecture

The Distributed Database System(DDBS) must be capable to support more complex and more sophisticated functionality. Networks have several types of topologies that define how nodes are physically and logically connected. One of the popular topology used in DDBS, the client-server architecture is described as follows: the principle idea of this architecture is to define specialized servers with specific functionalities. The servers are connected to a network of clients that can access the services of the servers. Stations (servers or clients) can have different design complexities starting from diskless client to combined server-client machine. The DBMS functions are divided between servers and clients using different approaches. The client refers to a data distribution dictionary to know how to decompose the global query into multiple local queries.

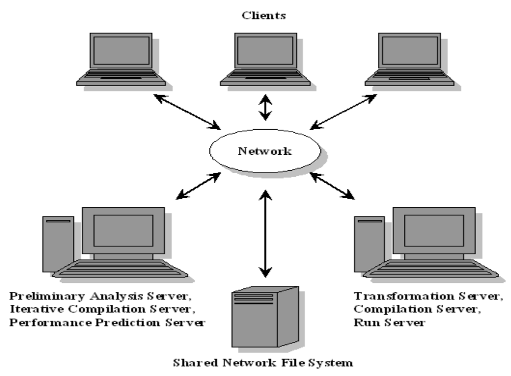

Clients
Network
Preliminary Analysis Server,
Iterative Compilation Server,
Performance Prediction Server
Transformation Server,
Compilation Server,
Run Server
Shared Network File System

Fig.2.1 Client Server Architecture

The interactions are as follows:

1. Client parses the user's query and decomposes it into independent site queries.
2. Client forwards each independent query to the corresponding server by consulting with the data distribution dictionary.
3. Each server processes the local query and sends back the resulting relation to the client.
4. Client combines (manually by the user, or automatically by client abstract) the received subqueries, and do more processing if needed to get to the final target result.

B. Fragmentation

The Primary concern of distributed database system design is to perform fragmentation of the relations in case of relational database or classes in case of object oriented databases and allocation of the fragments into different sites of the distributed system. Fragmentation is a design technique to divide a single relation or class of a database into two or more partitions such that on combining the partitions provides the original database without any loss of information. This reduces the amount of irrelevant data accesses by the applications, thus reducing the number of disk accesses. Fragmentation is classified into horizontal, vertical and mixed/hybrid.

i. Horizontal Fragmentation

Horizontal fragmentation (HF) allows a relation or class to be partitioned into disjoint tuples or instances. Intuition behind horizontal fragmentation is that every site should hold all information that is used to query at the site and the information at the site should be fragmented so the queries of the site run faster. Horizontal fragmentation is defined as selection operation of the relational algebra, σ (R).

Computing horizontal fragmentation

a. Compute the frequency of the individual queries $Q_1, . . Q_q$ of the site.
b. Rewrite the queries of the site in the conjunctive normal form (disjunction of conjunctions); the conjunctions are called minterms.
c. Compute the selectivity of the minterms.
d. Find the minimal and complete set of minterms (predicates).
e. The set of predicates is complete if and only if any two tuples in the same fragment are referenced with the same probability by any application.
f. The set of predicates is minimal if and only if there is at least one query that accesses the fragment.

ii. Vertical Fragmentation

Vertical fragmentation (VF) allows a relation or class to be partitioned into disjoint sets of columns or attributes except the primary key. Each partition must include the primary key attribute(s) of the table. This arrangement can make sense when different sites are responsible for processing different functions involving an entity.

Objective of vertical fragmentation is to partition a relation into a set of smaller relations so that many of the applications will run on only one fragment.

a. Vertical fragmentation of a relation R, produces the fragments R_1, R_2 etc. Each of which contains a subset of R’s attributes.

b. Vertical fragmentation is defined using the projection operation of the relational algebra: Π (R)

iii. Hybrid Fragmentation

Combination of horizontal and vertical fragmentations is mixed or hybrid fragmentations (MF). In this type of fragmentation scheme, the relation is divided into arbitrary blocks based on the transactions. Each fragment can be allocated on to a specific site. This type of fragmentation is the most complex one, which needs more management. In most cases simple horizontal or vertical fragmentation of a DB schema will not be sufficient to satisfy the requirements of the applications.

Mixed fragmentation (hybrid fragmentation) is carried out either by horizontal fragmentation followed by vertical fragmentation, or vertical fragmentation followed by horizontal fragmentation. Mixed Fragmentation is defined using the selection and projection operations of relational algebra:

Π_p(_A1,. .., An(R))

Π _A1,. .., An(_p(R))

A fragment of a relation is also a relation. Fragments can be further fragmented.

Overview of dynamic fragmentation

This section describes the proposed approach to fragment tables dynamically, and replicate those fragments on different sites in order to improve locality of table accesses and thus reduce communication costs. The proposed approach has two main components: 1) detecting replica access patterns, and 2) Decision on refragmentation and reallocation. Each site can take decisions to carry out fragmentation, replication and migration independently of other sites. This makes it possible to use our approach without communication overhead, changing the network protocol. In order to make informed decisions about fragmentation and replica changes, future accesses have to be predicted. As with most algorithms, predicting the future is based on knowledge of the past. In our approach, this means the detecting replica access patterns, i.e., which fragments are accessed by which sites. This is performed by recording replica accesses. Because of recording the access patterns continuously, old data may be discarded periodically such that statistics only include recent accesses. In this way, the system can adapt to the changes in access patterns. Statistics are stored using histograms design.

3. Fragmentation on unstructured data

The world-wide web (WWW) is often considered to be the world's largest database and the eXtensible Markup Language (XML) is then considered to provide its data model. There raises the question, how to obtain a suitable distribution design for XML documents. Here horizontal and vertical fragmentation techniques are generalised from the relational data model to XML. Furthermore, splitting is introduced as a third kind of fragmentation. Then it is shown how relational techniques for defining reasonable fragments can be applied to the case of XML.

In this section, XML is described as a data model. Extended DTDs(Document Type Definitions) are used to define schemata. Equivalently, XML-Schema is used, but extensions would be needed. Then it is considered to be the standard for XML documents as databases over such schemata. The queries are used with an extension of XML-QL. Equivalently, XQuery could be used, but again extensions would be needed in both cases.

A. Schemata and Document Type Definitions

A document type definition (DTD) may be considered as some kind of schema. Within such a DTD the regular expressions can be considered as some form of typing. We will make this view explicit and introduce a typed version of XML. The Types are to be considered are used as abstract syntax.

$t = b \mid \square \square t^0 \mid t^* \mid t^+ \mid t_{1 \ldots\ldots} t_n \mid t_1 \square_{\ldots\ldots} \square\ t_n$

Here, b represents as usual a collection of base types. Among these base types it is assumed to have a type ID, i.e., a type representing a not further specified set of identifiers. There may be other base types such as INT , STRING , URL for integer, character strings and URL-addresses respectively. □ is a type representing just an empty sequence or tuple. t* and t^+ represent arbitrary or non-empty sequences, respectively, with values of type t. t^0 represents the values of type t or the empty sequence. $t_{1 \ldots\ldots} t_n$ represents sequences or tuples. Finally, $t_1 \square_{\ldots\ldots} \square t_n$ represents a disjoint union.

B. Fragmentation

i. Split fragmentation

The splitting operation originates from work on object oriented databases. It simply takes a complex expression inside a class definition and replaces it by a reference to a new class. In the context of XML it is convenient to assume that the root-element has a defining expression of the form $n_1\ \square_{\ldots\ldots}\ \square\ n_k$, so that we could refer to each n_i as a class. Splitting would thus result in a new class n_{k+1}, and in some element we would now reference to this new class. It is also possible to place n_{k+1} into a completely new document with a newly created URL-address, in which the external references would be obtained.

The construction of a new XML document can be obtained by a query, though the following example.

Example query for split fragmentation:

```
<db>
CONSTRUCT
<wine w-id=$I producer=$P price=$Q>
<name_ref ref_to_name newID($N)/>
<rest>$R</>
</wine>
<name n-id=$N>$M</>
FROM
<db><wine w-id=$I producer=$P price=$Q>
<name>$M</>
<rest>$R</>
</></>IN"XYZ"
.
.
</db>
```

In the above example, splitting is carried out to separate names of wines from wine themselves where XYZ is an URL.

ii. Horizontal fragmentation

The two versions of generalising horizontal fragmentation from RDM to the object oriented case is considered. The first version which addresses horizontal fragmentation on the level of classes, whereas the second one addresses the problem on the level of bulk types inside the structure definition of the classes. However, at the end it turns out that the second version only leads to fragmentation, if it is followed by a splitting fragmentation. In this case, the same results can b

e obtained by applying the splitting fragmentation. The horizontal fragmentation can be achieved by the following selection query.

Example selection query for horizontal fragmentation:

```
<db>
CONSTRUCT
<cheap_wine w-id=$I producer=$p price=$Q>$W</>
FROM
<db><wine w-id=$I producer=$P
price=$Q>$W</></>IN "XYZ"
$Q<10
CONSTRUCT
<expensive_wine w-id=$I producer=$P
price=$Q>$W</>
FROM
<db><wine w-id=$I producer=$P
price=$Q>$W</></>IN "XYZ"
$Q>=10
.
.
</db>
```

In the above example, the dots indicate the parts dealing with carrying over vineyard and region-elements to the new document. Wines are fragmented into cheap wines with price less than 10 dollars and expensive wines with price greater than 10 dollars.

iii. Vertical Fragmentation

The Vertical fragmentation in the RDM corresponds to projecting to subsets of attributes. For an XML-element it is possible to define a vertical fragmentation, when

the defining expression is a sequence. The vertical fragmentation can be realized by projection queries. The queries are used on XML for fragmenting elements. The splitting cannot be expressed as a special form of vertical fragmentation.
Example projection query for vertical fragmentation:

```
<db>
CONSTRUCT
<wine w-id=$I price=$Q>
<name>$N</>
<year>$Y</></>
<wine-info for_wine=$I producer=$P>
<grapes>$G</></>
FROM
<db><wine w-id=$I producer=$P price=$Q>
<name>$N</>
<year>$Y</>
<grapes>$G</></></></>IN"XYZ"
.
.
</db>
```

In the above example, element wine is fragmented into new elements wine and wine-info.

4. Fragment allocation

The fragment allocation design is an essential issue that improves the performance of the applications processing in the Distributed Database systems. The database queries that access the applications on the distributed database sites should be performed effectively. Therefore the fragments that are accessed by queries are needed to be allocated to the distributed database sites so as to reduce the communication cost during the applications execution and handling their operational processing. A method for grouping the sites is proposed to optimize the cost of the fragment allocation functions and to reduce the queries processing time by allocating the fragments to the cluster of sites instead of allocating the fragments site by site.

Clustering sites

Clustering is the process of grouping sites according to a Communication Cost Range(CCR) to increase the system I/O performance and reduce storage overheads. Clustering helps in reducing the communication costs between the sites during the process of data allocation.
Two sites (S_i, S_j) are grouped in one cluster if the communication cost between them is less than or equal to a CCR; the number of communication units which is allowed for the maximum difference of the communication cost between the sites to be grouped in the same cluster, this number is determined by the network of the DDBs.

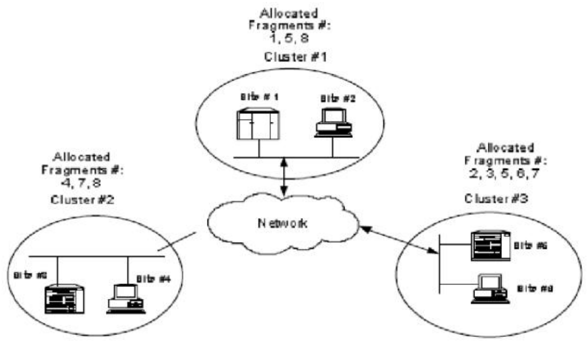

Allocated Fragments #: 1, 5, 8
Cluster #1
Site # 1
Site #2
Allocated Fragments #: 4, 7, 8
Cluster #2
Site #3
Site #4
Network
Allocated Fragments #: 2, 3, 5, 6, 7
Cluster #3
Site #5
Site #6

Fig. 4.1- shows the distribution of the fragments over the clusters.

Estimating the cost:

i. Cost of Allocating a Fragment to a Cluster

The cost for allocating the fragment F_i to the cluster C_j is computed as the sum of the following:

- The average cost of retrievals locally at cluster C_j times(CLRsum) the average number of frequency of retrieval(FREQLR) issued by the transaction T_k to the fragment F_i at cluster $C_{j.}$
  CLRsum(T_k, F_i, C_j) = CLR(T_k, F_i, C_j) * FREQLR(T_k, F_i, C_j)
- The average cost of local updates at cluster C_j times the average number of frequency of update(FREQLU) issued by the transaction T_k to the fragment F_i at the cluster C_j.
  CLUsum(T_k, F_i, C_j) = CLU(T_k, F_i, C_j)* FREQLU(T_k, F_i, C_j)
- The cost of space(CSP) occupied by the fragment F_i in the cluster C_j times the size of the fragment C_j times the size of the fragment F_i (in bytes).
  CSPsum(T_k, F_i, C_j)= CSP(T_k, F_i, C_j)* Fsize(Tk,Fi)
- Remote updates(CRU) sent from other clusters C_x; the average cost of local updates at cluster C_j times the average number of frequency of update(FREQRU) issued by the transaction T_k to the fragment F_i for each cluster other than the current one.
  CRUsum(T_k, F_i, C_j)= CLU(T_k, F_i, C_j)* FREQRU(T_k, F_i, C_j)
- Remote communications(CRC) from other clusters C_x; the update ratio(Uratio) (Unit Update/Unit Communication) times the average number of frequency of update issued by the transaction T_k to the fragment F_i at the cluster C_j times the average cost of communication between clusters other than the current one.
  CRCsum(T_k, F_i, C_j) = Uratio * FREQLU(T_k, F_i, C_j) *CRC(T_k, F_i, C_j)
- According to the previous formulas the Cost of Allocation CA(T k, Fi, Cj) is defined as the sum of

the following costs: local retrievals, local updates, space, remote update, and remote communication. $CA(T_k, F_i, C_j) = CLRsum(T_k, F_i, C_j) + CLUsum(T_k, F_i, C_j) + CSPsum(T_k, F_i, C_j) + CRUsum(T_k, F_i, C_j) +CRCsum(T_k, F_i, C_j)$

ii. Cost of Not Allocating a Fragment to a Cluster

The cost for not allocating the fragment F_i to the cluster C_j is computed as the sum of the following:

- The average cost of local retrievals at cluster C_j times the average number of frequency of retrieval issued by the transaction T_k to the fragment F_i at the cluster C_j. It is the same as defined in previous section.
- Retrievals from other clusters C_x of remote sites; the retrieval ratio(CRR) (Unit Retrieval/Unit Communication) times the average number of frequency of retrieval issued by the transaction T_k to the fragment F_i at the cluster C_j for each cluster other than the current one times the average cost of communication between clusters(CCC).
  $CRRsum(T_k, F_i, C_j) = Ratio * FREQRR(T_k,F_i,C_j) * CCC$
- According to the formulas specified previously, the Cost of Not Allocation CN (T_k, F_i, C_j) is defined as the sum of cost of local retrievals and sum of cost of remote retrievals.
  $CN(T_k, F_i, C_j) = CLRsum(T_k, F_i, C_j) +CRRsum(T_k, F_i, C_j)$

iii. The Decision Value for Allocating a Fragment to a Cluster

The decision values for allocating the fragment F_i to the cluster C_j is a logical value and computed as follows.
$D(T_k, F_i, C_j)= CN(T_k, F_i, C_j)>= CA(T_k, F_i, C_j)$

The following is an illustration for fragment allocation method, in which the fragments and their number of frequencies of retrieval and update requested from each cluster and its respective sites(Table 1), the costs of space, retrieval, and update is calculated based on the following number of bytes which required for the computation of the update and retrieval ratios according to their use in the DDBs: 2 bytes in each unit of retrieval, 3 bytes in each unit of update, and 5 bytes in each unit of communication(Table 2).

Table 1- Fragments and their frequencies of retrievals and updates in the clusters and their respective sites

Fragment	Cluster	Site	Retrieval frequency	Update frequency
F_1	C_1	S_1	80	10
		S_2	60	26
	C_2	S_3	60	16
		S_4	0	0
	C_3	S_5	35	5
		S_6	25	5
F_2	C_1	S_3	20	4
		S_4	20	6
	C_2	S_5	5	30
		S_6	105	20
F_3	C_2	S_3	30	0
		S_4	0	0
	C_3	S_5	40	30
		S_6	30	10

Table 2- Cost of space, retrieval and update

Cluster	Site	Cost Of Space	Cost Of Retrieval	Cost Of Update
C_1	S_1	0.004	0.15	0.25
	S_2	0.006	0.25	0.35
C_2	S_3	0.005	0.15	0.25
	S_4	0.007	0.17	0.27
C_3	S_5	0.003	0.13	0.23
	S_6	0.005	0.15	0.25

On applying the formulas described in section IV-i,ii,iii with the given data, the fragments that are allocated to the cluster and the fragments that are cancelled when the communication cost is more or less same for allocating to the sites individually are determined in the Table 3.

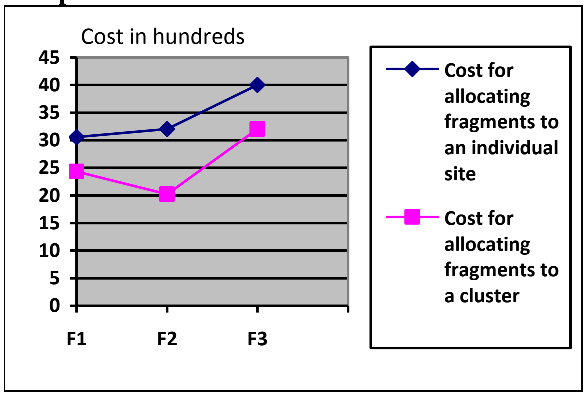
Table 3- Allocated and cancelled fragments to the clusters

Frag-ment	Cluster	CA	CN	DV	Allocation Status
F_1	C_1	59.45	177.24	1	Allocated
	C_2	74.83	74.76	0	Cancelled
	C_3	85.5	74.16	0	Cancelled
F_2	C_2	74.26	49.84	0	Cancelled
	C_3	30.01	135.96	1	Allocated

F_3	C_1	86.56	96.21	1	Allocated
	C_2	103.2	37.38	0	Cancelled
	C_3	54.72	86.52	1	Allocated

In the above table, CA indicates Cost of Allocation, CN indicates Cost of Non allocation and DV indicates Decision Value.

Comparison chart:



The above graph describes the comparison between the cost of allocating the fragments to an individual site and cluster of sites.

5. Conclusion

The importance of the distributed database systems has increased further with developments in networking technologies. Effective distribution of the database fragments plays a critical role in the functioning of the database in terms of performance and cost. In this paper, a new formulation for the problem of fragmenting and allocating those fragments at minimum cost is presented for both structured and unstructured data. Results from the application of these formulations can be utilized for a large number of data sets.

References

[1]Syam Menon, "Allocation of fragments in distributed system",IEEE Transactions on parallel and distributed system,vol 16, No.7,July 2005.
[2]Hassan I Ahdalla,"A New data reallocation model for distributed database systems", International journal of database theory and application.vol.5, No.2,June 2012.
[3]Yin- Fu.Huang and Jyh- Herchen,"Fragment Allocation in distributed database design", Journal of information science and engineering 17,491-506(2001).
[4]Leon Tambulea, Manuela," Redistributing fragments into distributed databases", Int. J. of computers,communication & control, ISSN 1841-9836 vol III(2008).
[5]Shahidul Islam Khan,Dr.A.S.M Latiful Hoque,"A new technique for database fragmentation in distributed systems",International journal of computer applications vol.5,No.9,August 2010.
[6]C.I Ezeife,Jlan Zheng, "Dynamic database object horizontal fragmenatation".
[7]Ismail.O.Habebeh,"A method for fragment allocation design in the distributed database systems",The sixth annual UAE university research conference.
[8]Valentina ciriani,Sabrina De Capitani di vimercati,Sara foresti,"Fragmentation Design for efficient query execution over sensitive distributed databases",IEEE transaction,2009.
[9]David Pinto,Guadalupe Torres,"On dynamic fragmentation of distributed database using partial replication".
[10]Azzam Sleit, Wesam Al Mobaideen,"A Dyanamic object fragmentation and replication algorithm in distributed database system", American journal of applied science,2007.
[11]Arjun Singh and K.S. Kahlon," Non replicated dynamic data allocation in distributed database systems", IJCSNS International journal of computer science and network security, vol.9 No.9,September 2009.
[12]Alessandro Mei, Luigi V.Mancini and Sushil Jajodia," Secure dynamic fragment and replica allocation in large-scale distributed file systems", IEEE transactions on parallel and distributed systems, Vol.14,No.9,September 2003.
[13]Jon Olav Hauglid, Norvald H.Ryeng and Kjetil, "Dynamic fragmentation and replica management in distributed database systems", Distributed and parallel database manuscript.
[14]Ajit M.Tamhankar and Sudha Ram, "Database fragmentation and allocation: An integrated methodology", IEEE Transaction on systems,Cybernetics, Vol.28,No.3,May 1998.
[15]Hui Ma, Klaus Dieter Schewe, "Fragmentation of XML document", Massey University, Information systems.
[16]Imran R. Mansuri, Sunita Sarawgi," Integrating unstructured data into relational databases".
[17]Ayaz Ahmed Shariff K, Mohammed Ali Hussain, Sambath kumar," Leveraging unstructured data into intelligent information- Analysis and evaluation", 2011 International Conference on Information and Network Technology, IPCSIT vol.4 (2011).