Available at: www.researchpublications.org

Distributed Database: A Survey

Himanshu Joshi[#1], G.R.Bamnote[*2]

[#]*PRMIT&R Anjangoan Bari Road,*
Badnera Amravati-444701

[1]himanshujoshi18@gmail.com
[3]grbamnote@rediffmail.com

***Abstract*—The purpose of this paper is to present an introduction to Distributed Database. It contains two main parts: first one is fundamental concept in Distributed Database and second one is Different technique use in Distributed Database. Database with a production of huge data sets and their processing in real-time applications, the needs for environmental data management have grown significantly. Management Systems (DBMSs) are a ubiquitous and critical component of modern computing. The architecture and motivation for the design have also been presented in this paper. The Proposed Method is Distributed Data Mining. It is also use for to reduce the complexity of database.**

***Keywords*—Distributed Database, DBMS, Computing.**

I. INTRODUCTION

This A distributed database is a database distributed between several sites. The reasons for the data distribution may include the inherent distributed nature of the data or performance reasons. In a distributed database the data at each site is not necessarily an independent entity, but can be rather related to the data stored on the other sites.[1]

A distributed database (DDB) is a collection of multiple, logically interrelated databases distributed over a computer network. A distributed database management system (DDBMS) is the software that manages the DDB, and provides an access mechanism that makes this distribution transparent to the user. Distributed database system (DDBS) is the integration of DDB and DDBMS. This integration is achieved through the merging the database and networking technologies together. [2]

A distributed database can reside on network servers on the Internet, on corporate intranets or extranets, or on other company networks. The replication and distribution of databases improves database performance at end-user worksites. To ensure that the distributive databases are up to date and current, there are two processes: replication and duplication. Replication involves using specialized software that looks for changes in the distributive database. Once the changes have been identified, the replication process makes all the databases look the same.[3]

A Distributed Database has three types

Homogenous distributed database system
Heterogeneous distributed database system
Client/server Database System

Architecture of Distributed Database

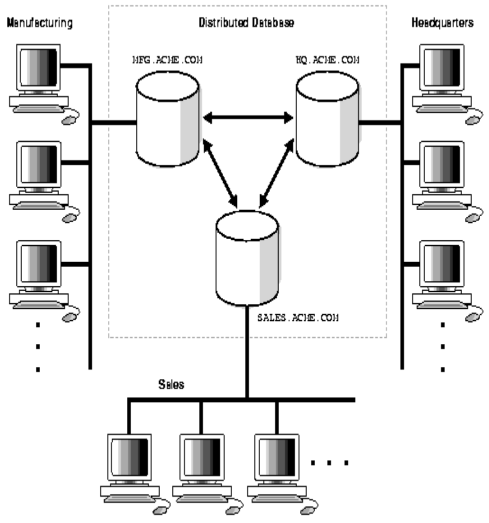

Fig. Architecture of Distributed Database

A) Survey on Distributed database

B.M. Monjurul Alom et al used Query Processing and Optimization in Distributed Database

Systems improve.In that they used FRS strategy, In this strategy, if no relation referenced by a query is fragmented, it is necessary to decide which relation is to be partitioned into fragments; which copy of the relation should be used; how the relation is to be partitioned.[4]

P. R. Bhuyar et. al. used Horizontal Fragmentation Technique for Distributed database. Horizontal fragmentation was used for improving the performance of application. Allocation of fragments is done simultaneously in the algorithm. 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.[5]

Ceri et al. used HF using min-term predicate to reduce the complexity of database. This technique used for the determination of relevant portions of data are defined and a methodology for determining the access parameters [6]

Ozsu and Valduriez proposed an iterative algorithm COMMIN to generate a complete and minimal set of predicates from a given set of simple predicates.[7]

Available at: www.researchpublications.org

Bai˜oo et al. inputted predicate affinity matrix to build a predicate affinity graph thus defines horizontal class fragments. The detailed methodology for the design of distributed object databases that includes: (i) an analysis phase, to indicate the most adequate fragmentation technique to be applied in each class of the database schema and semi; (ii) a horizontal class fragmentation algorithm, and (iii) a vertical class fragmentation algorithm. Basically, the analysis phase is responsible for driving the choice between the horizontal and the vertical partitioning techniques, or even the combination of both, in order to assist distribution designers in the fragmentation phase of object databases.[8]

Shin and Irani proposed knowledge based approach in which user reference clusters are derived from the user queries to the database and the knowledge about the data [9]

Cheng et al. presented a genetic algorithm based fragmentation approach that treats horizontal fragmentation as a travelling salesman problem [10]

Abuelyaman proposed a static algorithm Stat Part for Vertical Fragmentation. This algorithm used for improving the performance of database systems. The algorithm uses the number of occurrences of an attribute in a set of queries rather than the FOQ accessing these attributes. This enables the fragmentation of a database schema even before its tables are populated. [11]

Leon âmbulea and Manuela Horvat-Petrescu used heuristic algorithm for redistributing the fragments. The algorithm uses the statistical information relative to the requests send to a distributed database. This algorithm minimizes the size of the data transferred for solving a request. The algorithm generates a plan to transfer data fragments, plan that will be used to evaluate a request [12]

An electronic copy can be downloaded from the conference website. For questions on paper guidelines, please contact the conference publications committee as indicated on the conference website. Information about final paper submission is available from the conference website.

II. Components of Distributed database

A) Distributed catalog management:

It's related to the unique identification of each fragment that has been either partitioned or replicated. This can be done by using a global name server that can assign globally unique names.

This can be implemented by using the following two fields:-

1. Local name field – locally assigned name by the site where the relation is created. Two objects at different sites can have same local names.

2. Birth site field – indicates the site at which the relation is created and where information about its fragments and replicas is maintained [2][3].

B) Catalog Structure:

A centralized system catalog is used to maintain the information about all the transactions in the distributed database but is vulnerable to the failure of the site containing the catalog. This could be avoided by maintaining a copy of the global system catalog but it involves broadcast of every change done to a local catalog to all its replicas. Another alternative is to maintain a local catalog at every site which keeps track of all the replicas of the relation.

C) Distributed Data Independence:

fragments or replicas of a relation which has to be It means that the user should be able to query the database without needing to specify the location of the done by the DBMS Users can be enabled to access relations without considering how the relations are distributed as follows:

The local name of a relation in the system catalog is a combination of a user name and a user-defined relation name.

When a query is fired the DBMS adds the user name to the relation name to get a local name, and then adds the user's site-id as the (default) birth site to obtain a global relation name. By looking up the global relation name in the local catalog if it is cached there or in the catalog at the birth site the DBMS can locate replicas of the relation.

III. Design Issues

A) Distributed Database Design

In the partitioned scheme the database is divided into a number of disjoint partitions each of which is placed at a different site. Replicated designs can be either fully replicated (also called fully duplicated) where the entire database is stored at each site, or partially replicated (or partially duplicated) where each partition of the database is stored at more than one site, but not at all the sites. The two fundamental design issues are fragmentation, the separation of the database into partitions called fragments, and distribution, the optimum distribution of fragments [4][5].

B) Distributed Directory Management

A directory contains information (such as descriptions and locations) about data items in the database. Problems related to directory management are similar in nature to the database placement problem discussed in the preceding section. A directory may be global to the entire DDBS or local to each site; it can be centralized at one site or distributed over several sites; there can be a single copy or multiple copies [2].

C) Distributed Query Processing

Query processing deals with designing algorithms that analyse queries and convert them into a series of data manipulation operations. The problem is how to decide on a strategy for executing each query over the network in the most cost-effective way, however cost is defined. The factors to be considered are the distribution of data, communication costs, and lack of sufficient locally-available information. The objective is to optimize where the inherent parallelism is used to improve the performance of executing the transaction, subject to the above-mentioned constraints. The problem is NP-hard in nature, and the approaches are usually heuristic [3].

D) Distributed Concurrency Control

Concurrency control involves the synchronization of accesses to the distributed database, such that the integrity of the database is maintained. It is, without any doubt, one of the most extensively studied problems in the DDBS field. The concurrency control problem in a distributed context is somewhat different than in a centralized framework. One not only has to worry about the integrity of a single database [3] [4].

E) Distributed Deadlock Management

The deadlock problem in DDBSs is similar in nature to that encountered in operating systems. The competition among users for access to a set of resources (data, in this case) can result in a deadlock if the synchronization mechanism is based on locking [3].

IV. Detailed Example

A) Distributed query processing

In a distributed system several factors complicates the query processing. One of the factors is cost of transferring the data over network. This data includes the intermediate files that are transferred to other sites for further processing or the final result files that may have to be transferred to the site where the query result is needed.

Although these cost may not be very high if the sites are connected via a high local n/w but sometime they become quite significant in other types of network. Hence, DDBMS query optimization algorithms consider the goal of reducing the amount of data transfer as an optimization criterion in choosing a distributed query execution strategy [1].

Consider an EMPLOYEE relation.

The size of the employee relation is 100 * 10,000=10^6 bytes

The size of the department relation is 35 * 100=3500 bytes

EMPLOYEE
(Fname Lname SSN Bdate Add Gender Salary Dnum)
-10,000 records
-Each record is 100 bytes
-Fname field is 15 bytes long
-SSN field is 9 bytes long
-Lname field is 15 bytes long
-Dnum field is 4 byte long

DEPARTMENT
(Dname Dnumber MGRSSN MgrStartDate)
-100records
-Each record is 35 bytes long
-Dnumber field is 4 bytes long
-Dname field is 10 bytes long
-MGRSSN field is 9 bytes long
-Now consider the following query:

"For each employee, retrieve the employee name and the name of the department for which the employee works."Using relational algebra this query can be expressed as:-

FNAME, LNAME, DNAME
(EMPLOYEE * DNO=DNUMBER DEPARTMENT)

If we assume that every employee is related to a department then the result of this query will include 10,000 records. Now suppose that each record in the query result is 40 bytes long and the query is submitted at a distinct site which is the result site.

Then there are 3 strategies for executing this distributed query:

1. Transfer both the EMPLOYEE and the DEPARTMENT relations to the site 3 that is your result site and perform the join at that site. In this case a total of 1,000,000 + 3500 = 1,003,500 bytes must be transferred.
2. Transfer the EMPLOYEE relation to site 2 (site where u have Department relation) and send the result to site 3. The size of the query result is 40 * 10,000 = 400,000 bytes so 400,000 + 1,000,000 = 1,400,000 bytes must be transferred.
3. Transfer the DEPARTEMNT relation to site 1 (site where u have Employee relation) and send the result to site 3. in this case 400,000 + 3500 = 403,500 bytes must be transferred [5].

Conclusions

The Distributed database is basically used for to store large amount of data on different site or server. There are Various Partitions technique uses in distributed database, such as vertical partition and vertical partition. In this paper provides overview of various partition methods and algorithms for distributed database. The future works mainly concentrate on use of Distributed data mining on Distributed database.

References

[1] Arti Kadav et. al. Data Mining Standard

[2] Haraun Rababach,"Distributed Database Fundamental And Research"

[3] Abadi, D., Carney, D., Cetintemel, U., Cherniack, M., Convey, C., Lee, S., Stonebraker, M., Tatbul, N., and Zdonik, Aurora: A new model and architecture for data stream management. Summer-2003

[4] Akal, F., Bohm, K., and Schek, H.-J. Olap query evaluation in a database ¨cluster: A performance study on intra-query parallelism. August 2002.

[5] Maniural B.M et al.,"Query Processing and Optimization in distributed database",IJCSNS,vol 9,No.9,2009

[6] Bhuyar P.R. "Horizonatal Fragmentation technique in Distributed database",IJSRP,vol2,issue 5,2012

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

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

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

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

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

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

[13] Ţâmbulea L., Horvat-Petrescu M., Redistributing Fragments into a Distributed Database , International Journal of Computers Communications & Control, ISSN 1841-9836, 3(4):384-394, 2008