Journal of Innovative Engineering and Research (JIER)
Vol.- 1, Issue - 2, October 2018, pp. 1-5 (5 pages)

Fundamental Concepts & Techniques of Distributed Database

MEGHA SINGH[1*] and RAHUL SHARMA[2]

[1,2]*Department of Computer Science & Engineering, Dr. A. P. J. Abdul Kalam University, Indore*
**Corresponding Author Email: meghasingh@aku.ac.in*

Abstract - The purpose of this paper is to present an introduction to Distributed Databases which are becoming very popular now days. Distributed database systems provide an improvement on communication and data processing due to its data distribution throughout different network sites. Not only is data access faster, but a single-point of failure is less likely to occur, and it provides local control of data for users. We present a study of the fundamentals of distributed databases (DDBS). We discuss issues related to the motivations of DDBS, architecture, design, performance, and concurrency control, etc. and also we explore some of the research that has been done in this specific area of DDBS. In this paper we present some research on: query optimization, fragmentation.

Index Terms— Query Optimization, Concurrency Control, Recovery Fragmentation Optimization,

I. INTRODUCTION

In today's world of universal dependence on information systems, all sorts of people need access to companies' databases. In addition to a company's own employees, these include the company's customers, potential customers, suppliers, and vendors of all types. It is possible for a company to have all of its databases concentrated at one mainframe computer site with worldwide access to this site provided by telecommunications networks, including the Internet. Although the management of such a centralized system and its databases can be controlled in a well-contained manner and this can be advantageous, it poses some problems as well. For example, if the single site goes down, then everyone is blocked from accessing the databases until the site comes back up again. Also the communications costs from the many far PCs and terminals to the central site can be expensive. One solution to such problems, and an alternative design to the centralized database concept, is known as distributed database. The idea is that instead of having one, centralized database, we are going to spread the data out among the cities on the

In distributed network, each of which has its own computer and data storage facilities. All of this distributed data is still considered to be a single logical database. When a person or process anywhere on the distributed network queries the database, it is not necessary to know where on the network the data being sought is located. The user just issues the query, and the result is returned. This feature is known as location transparency. This can become rather complex very quickly, and it must be managed by sophisticated software known as a distributed database management system or distributed DBMS.

Definition

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. Or it can be described as, a system that runs on a collection of machines that do not have shared memory, yet looks to the user like a single machine.

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. Or it can be described as, a system that runs on a collection of machines that do not have shared memory, yet looks to the user like a single machine. A distributed database (DDB) is a collection of multiple, logically interrelated databases distributed over a computer network. A distributed database management system (distributed DBMS) is the software system that permits the management of the distributed database and makes the distribution transparent to the users. The term distributed database system (DDBS) is typically used to refer to the combination of DDB and the distributed DBMS. Distributed DBMSs are similar to distributed file systems (see Distributed File Systems) in that both facilitate access to distributed data.

II. CHARACTERISTICS

However, there are important differences in structure and functionality, and these characterize a distributed database system. Distributed file systems simply allow users to access files that are located on machines other than their own. These files have no explicit structure (i.e., they are flat) and the relationships among data in different files (if there are any) are not managed by the system and are the users responsibility. A DDB, on the other hand, is organized according to a schema that defines both the structure of the distributed data, and the relationships among the data. The

schema is defined according to some data model, which is usually relational or object-oriented.

A distributed file system provides a simple interface to users which allows them to open, read/write (records or bytes), and close files. A distributed DBMS system has the full functionality of a DBMS. It provides high-level, declarative query capability, transaction management (both concurrency control and recovery), and integrity enforcement. In this regard, distributed DBMSs are different from transaction processing systems as well, since the latter provide only some of these functions.

A distributed DBMS provides transparent access to data, while in a distributed file system the user has to know (to some extent) the location of the data. A DDB may be partitioned (called fragmentation) and replicated in addition to being distributed across multiple sites. All of this is not visible to the users. In this sense, the distributed database technology extends the concept of data independence, which is a central notion of database management, to environments where data are distributed and replicated over a number of machines connected by a network. Thus, from a user s perspective, a DDB is logically a single database even if physically it is distributed

Distributed database Governs storage and processing of logically related data over interconnected computer systems in which both data and processing functions are distributed among several sites.

III. DDBS ARCHITECTURE

Hardware

Due to the extended functionality the DDBS must be capable of, the DDBS design becomes more complex and more sophisticated. At the physical level the differences between centralized and distributed systems are:

Multiple computers called sites.

These sites are connected via a communication network, to enable the data/query communication.

Networks can have several types of topologies that define how nodes are physically and logically connected. One of the popular topologies 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 such as: printer server, mail server, file server, etc. these serves then are connected to a network of clients that can access the services of these servers. Stations (servers or clients) can have different design complexities starting from diskless client to combined server-client machine.

The server-client architecture requires some kind of function definition for servers and clients. The DBMS functions are divided between servers and clients using different approaches. We present a common approach that is used with relational DDBS, called centralized DMBS at the server level. The client refers to a data distribution dictionary to know how to decompose the global query in to multiple local queries. The interaction is done as follows:

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

We would like to discuss the different architectures of DDBS for the two main types, the client/server, and the distributed databases:

- ***The client/server:*** The file server approach: the simplest tactic is known as the file server approach. When a client computer on the LAN needs to query, update, or otherwise use a file on the server, the entire file must be sent from the server to that client. All of the querying, updating, or other processing is then performed in the client computer. If changes were made to the file, the entire file is then shipped back to the server. Clearly, for files of even moderate size, shipping entire files back and forth across the LAN with any frequency will be very costly. In terms of concurrency control, obviously the entire file must be locked while one of the clients is updating even one record in it. Other than providing a basic file-sharing capability, this arrangement's drawbacks render it not very practical or useful.

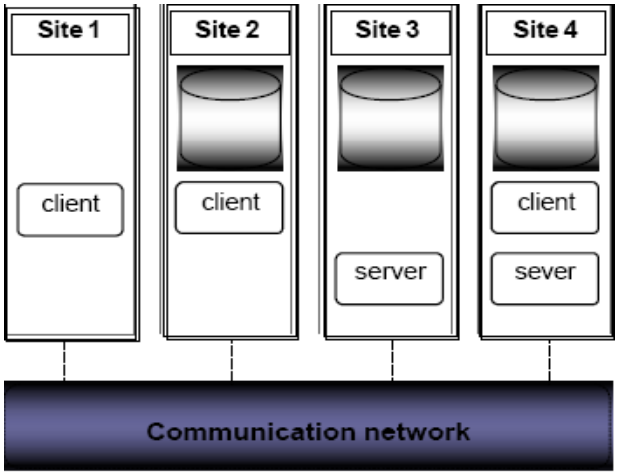

Fig 1: Client/Server Architecture

Advantages of Client/Server architecture include:

- More efficient division of labor,
- Horizontal and vertical scaling of resources,
- Better price/performance on client machines,
- Ability to use familiar tools on client machines,
- Client access to remote data (via standards),
- Full DBMS functionality provided to client workstations,
- Overall better system price/performance

Disadvantages of Client/Server architecture include:

- Server forms bottleneck,
- Server forms single point of failure,
- Database scaling is difficult.

It is preferable for a DDMBS to have the property of distribution transparency where the user's can issue global queries without knowing or worrying about the global distribution in the DDBS.

- ***DBMS server approach***: A much better arrangement is variously known as the database server or DBMS server approach. Again, the database is located at the server, but this time, the processing is split between the client and the server, and there is much less data traffic on the network. Say that someone at a client computer wants to query the database at the server. The query is entered at the client, and the client computer performs the initial keyboard and screen interaction processing, as well as initial syntax checking of the query. The system then ships the query over the LAN to the server where the query is actually run against the database. Only the results are shipped back to the client. Certainly, this is a much better arrangement than the file server approach! The network data traffic is reduced to a tolerable level, even for frequently queried databases. Also, security and concurrency control can be handled at the server in a much more contained way. The only real drawback to this approach is that the company must invest in a sufficiently powerful server to keep up with all of the activity concentrated there.
- ***Two-tier client/server***: Another issue involving the data on a LAN is the fact that some databases can be stored on a client PC's own hard drive while other databases that the client might access are stored on the LAN's server. This is also known as a two-tier approach, Software has been developed that makes the location of the data transparent to the user at the client. In this mode of operation, the user issues a query at the client, and the software first checks to see if the required data is on the PC's own hard drive. If it is, the data is retrieved from it, and that is the end of the story. If it is not there, then the software automatically looks for it on the server. In an even more sophisticated three-tier approach if the software doesn't find the data on the client PC's hard drive or on the LAN server, it can leave the LAN through a gateway computer and look for the data on, for example, a large, mainframe computer that may be reachable from many LANs.

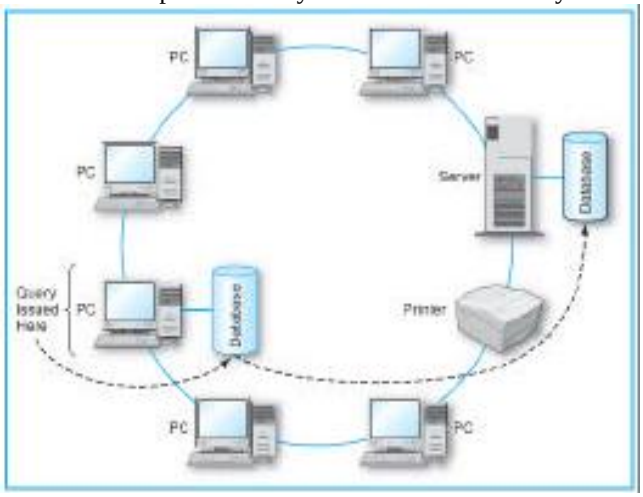

Fig 2: Two tier Client/Server

Three-tier approach: In another use of the term three-tier approach, the three tiers are the client PCs, servers known as application servers, and other servers known as database servers. In this arrangement, local screen and keyboard interaction is still handled by the clients, but they can now request a variety of applications to be performed at and by the application servers. The application servers, in turn, rely on the database servers and their databases to supply the data needed by the applications. The local processing on the clients is limited to the data input and data display capabilities of browsers such as Netscape's Communicator and Microsoft's Internet Explorer.

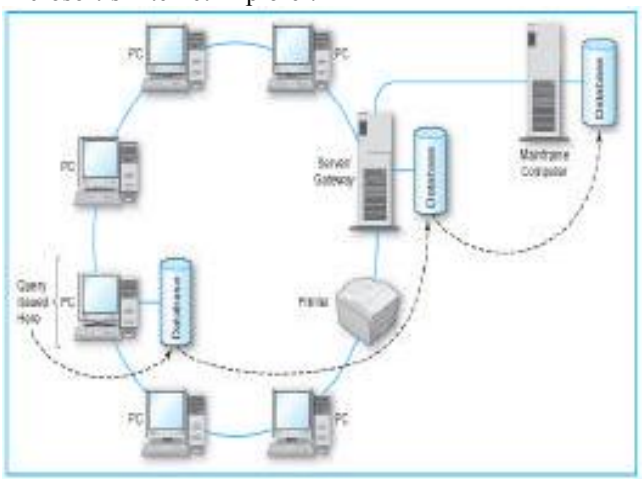

Fig 3: Three tier Client/Server

The application servers are the computers at company Web sites that conduct the companies' business with the "visitors" working through their browsers. The company application servers in turn rely on the companies' database servers to provide the necessary data to complete the transactions. For example, when a bank's customer visits his bank's Web site, he can initiate lots of different transactions, ranging from checking his account balances to transferring money between accounts to paying his credit card bills. The bank's Web application server handles all of these transactions. It, in turn, sends requests to the bank's database server and databases to retrieve the current account balances, add money to one account while deducting money from another in a funds transfer, and so forth

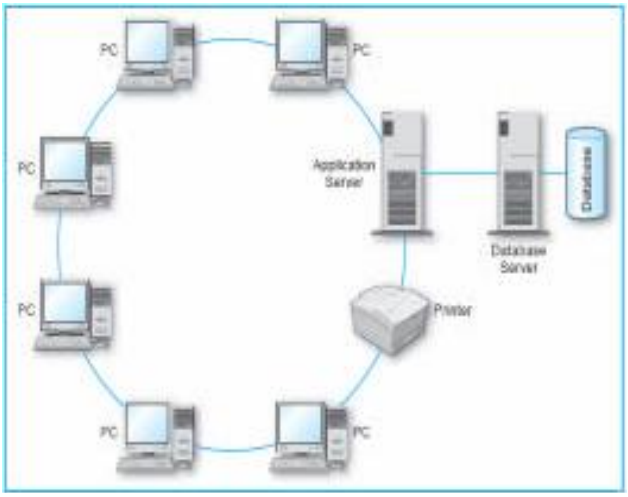

Fig 4: Another Version of Three tier

Journal of Innovative Engineering and Research (JIER)
Vol.- 1, Issue - 2, October 2018, pp. 1-5 (5 pages)

Software

In a typical DDBS, three levels of software modules are defined:

- The server software: responsible for local data management at site.
- The client software: responsible for most of the distribution functions; DDBMS catalog, processes all requests that require more than one site. Other functions for the client include: consistency of replicated data, atomicity of global transactions.
- The communications software: provides the communication primitives, used by the client/server to exchange data and commands of following fig 5.

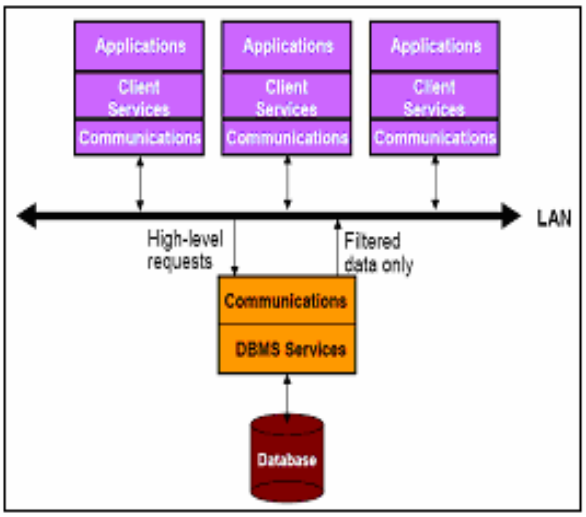

Fig 5: Client/Server Software

It is preferable for a DDMBS to have the property of distribution transparency (Shown in following fig.), where the user's can issue a global queries without knowing or worrying about the global distribution in the DDBS.

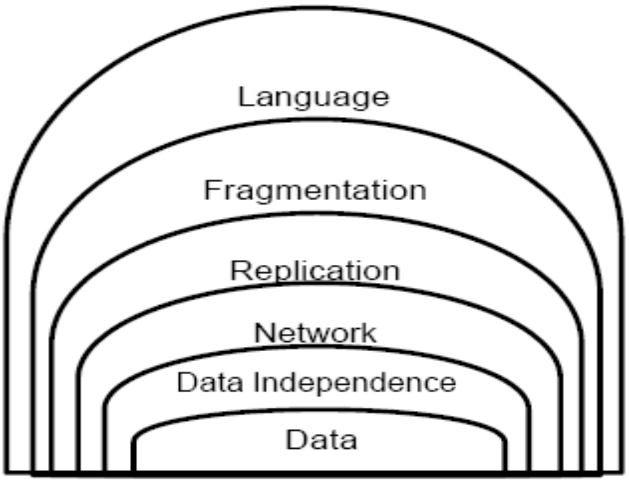

Fig 6: Layer Transparency

IV. FRAGMENTATION, REPLICATION

In distributing and allocating the database in the previous section, we assumed that the entire relations are kept intact. However, in DDBS we need to define the logical unit of DB distribution and allocation. In some cases it might be more efficient to split the tables into smaller units (fragments) and allocate them in different sites.

Fragmentation has three different types.

Horizontal Fragmentation:

The template is designed so that author affiliations are not repeated each time for multiple authors of the same affiliation. Please keep your affiliations as succinct as possible (for example, do not differentiate among departments of the same organization). This template was designed for two affiliations.

For author/s of only one affiliation (Heading 3): To change the default, adjust the template as follows.

Selection (Heading 4): Highlight all author and affiliation lines.

Change number of columns: Select Format > Columns >Presets > One Column.

Deletion: Delete the author and affiliation lines for the second affiliation.

For author/s of more than two affiliations: To change the default, adjust the template as follows.

Selection: Highlight all author and affiliation lines.

Change number of columns: Select Format > Columns > Presets > One Column.

Highlight author and affiliation lines of affiliation 1 and copy this selection.

Formatting: Insert one hard return immediately after the last character of the last affiliation line. Then paste the copy of affiliation 1. Repeat as necessary for each additional affiliation.

Reassign number of columns: Place your cursor to the right of the last character of the last affiliation line of an even numbered affiliation (e.g., if there are five affiliations, place your cursor at end of fourth affiliation). Drag the cursor up to highlight all of the above author and affiliation lines. Go to Format > Columns and select "2 Columns". If you have an odd number of affiliations, the final affiliation will be centered on the page; all previous will be in two columns.

Vertical Fragmentation:

Headings, or heads, are organizational devices that guide the reader through your paper. There are two types: component heads and text heads.

Component heads identify the different components of your paper and are not topically subordinate to each other. Examples include Acknowledgments and References and, for these, the correct style to use is "Heading 5". Use "figure caption" for your Figure captions, and "table head" for your table title. Run-in heads, such as "Abstract", will require you to apply a style (in this case, italic) in addition to the style provided by the drop down menu to differentiate the head from the text.

Text heads organize the topics on a relational, hierarchical basis. For example, the paper title is the primary text head because all subsequent material relates and elaborates on this one topic. If there are two or more sub-topics, the next level head (uppercase Roman numerals) should be used and, conversely, if there are not at least two sub-topics, then no

subheads should be introduced. Styles named “Heading 1”, “Heading 2”, “Heading 3”, and “Heading 4” are prescribed.

Hybrid Fragmentation:

Positioning Figures and Tables: Place figures and tables at the top and bottom of columns. Avoid placing them in the middle of columns. Large figures and tables may span across both columns. Figure captions should be below the figures; table heads should appear above the tables. Insert figures and tables after they are cited in the text. Use the abbreviation “Fig. 1”, even at the beginning of a sentence.

V. Concurrency Control

DDBS design of concurrency and recovery has to consider different aspects other than of those of centralized DBS. These aspects include:

- Multiple copies of data: concurrency has to maintain the data copies consistent. Recovery on the other hand has to make a copy consistent with others whenever a site recovers from a failure.
- Failure of communication links
- Failure of individual sites
- Distributed commit: during transaction commit some sites may fail, so the two phase commit is used to solve this problem.
- Deadlocks on multiple sites.

The following two sections describe two suggestions to manage concurrency control.

Distinguished Copy of a Data Item

There are three variations to this method: primary site technique, primary site with backup site, and primary copy technique. These techniques are described as follows:

- *Primary site* In this method, a single site is designated as the coordinator site. All locks and unlocks for all data units are controlled by this site. One advantage is, easy to implement. However two downsides of this method are: overloading of the coordinator site, and this site forms a single point failure for the entire DDBS.
- *Primary site with backup site* This technique addresses the second disadvantage in the 1st technique (primary site) by designating a backup site, that can take over as the new coordinator in case of failure, in which case, an other backup site has to be selected.
- *Primary copy technique* This method distributes the load to the sites that have a designated primary copy of a data unit as opposed to centralizing the entire data units in one coordinator site. This way if a site goes down, only transactions involving the primary copies residing on that site will be affected.

VI. Query Processing

DDBS adds to the conventional centralized DBS some other types of processing expenses, because of the additional design (hardware & software) to handle the distribution. These expenses present as the cost of data transfer over the network. Data transferred could be, intermediate files resulting from local sites, or final results need to be sent back to the original site that issued the query. Therefore, database designers are concerned about query optimization, which target minimizing the cost of transferring data across the network. One method to optimize query on DDBS is, the *simijoin*, where a relatio n R1 can send the entire join-column CR1 to the target relation R2, then the site containing R2 would perform the join on CR1, and project on the passed attributes. The resulting tuples are then shipped back to R! for further processing. This can significantly enhance the query efficiency, since the data transferred on the network is minimized

VII. Conclusion

We presented an introduction to distributed database design through a study that targeted two main parts: in the first part we presented an exploration of the fundamentals of DDBS, and the alternatives of their design. These alternatives addressed issues such as, architecture, distribution, query processing, concurrency control, etc. Through this paper, we want to attract readers towards the advantageous side of distributed databases. We also mentioned the software architecture being used for the distributed database .We also described Fragmentation, replication and recovery aspect also in order to make readers completely aware about the topic being described here. Besides having a fruitful side of DDBs, It also attracts researchers for finding the new scope in it.

REFERENCES

[1] Patrick O’Neil, and Goetz Graefe. 1995. Multi-Table Joins Through Bitmapped Join Indices. SIGMOD Record,

[2] Ambrose Goicoechea. 2000. Requirements Blueprint and Multiple Criteria For Distributed Database Design. International Council on Systems Engineering (INCOSE)

[3] Yin-Fu Huang, and Jyh-Her Chen. 2001. Fragment Allocation in Distributed Database Design. Journal of Information Science and Engineering 17, 491-506 (2001).

[4] Ramez Elmasri, and Shamkant B. Navathe. 1999. Fundamentals of Database Systems. Addison Wesley Longman, Inc.

[5] Tamer Özsu, and Patrick Valduriez. 1998. Distributed Database Management Systems. Purdue University, Computer Science department.

[6] The University of Queensland, School of Information Technology and Electrical

[7] Mark L. Gillenson. 2004. Fundamentals of Database Systems. Wiley E-Books. www.wiley.com/.

[8] Zhili Zhang and WiliamPerrizo. 2000. Distributed Query Processing Using Active Networks. ACM 1-58113-239-5/00/00