International Journal of Information and Computation Technology.
ISSN 0974-2239 Volume 4, Number 2 (2014), pp. 207-214

http://www. irphouse.com /ijict.htm

An Overview of Distributed Databases

Parul Tomar[1] and Megha[2]

[1]Department of Computer Engineering, YMCA University of Science & Technology, Faridabad, INDIA.
[2]Student Department of Computer Science, YMCA University of Science and Technology, Faridabad, INDIA.

Abstract

A Database is a collection of data describing the activities of one or more related organizations with a specific well defined structure and purpose. A Database is controlled by Database Management System(DBMS) by maintaining and utilizing large collections of data. A Distributed System is the one in which hardware and software components at networked computers communicate and coordinate their activity only by passing messages. In short a Distributed database is a collection of databases that can be stored at different computer network sites. This paper presents an overview of Distributed Database System along with their advantages and disadvantages. This paper also provides various aspects like replication, fragmentation and various problems that can be faced in distributed database systems.

Keywords: Database, Deadlock, Distributed Database Management System, Fragmentation, Replication.

1. Introduction

A Database is systematically organized or structuredrepository of indexed information that allows easy retrieval, updating, analysis, and output of data. Each database may involve different database management systems and different architectures that distribute the execution of transactions [1]. A distributed database is a database in which storage devices are not all attached to a common processing unit such as the CPU. It may be stored in multiple computers, located in the same physical location; or may be dispersed over a network of interconnected computers. A distributed database system consists of loosely-coupled sites that share no physical components [2]. In Centralized systems (Fig. 1), Data, Process and Interface components of an information system are central[3].

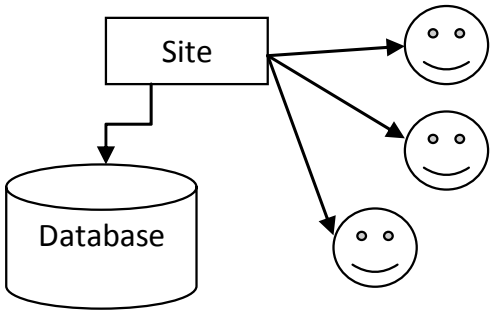

Fig. 1: Centralized Database System.

In order to work on the system end users uses terminals or terminal emulators. In Distributed System [3, 4, 5] (Fig. 2), Data, Process, and Interface components of an information system are distributed to multiple locations in a computer network. Accordingly, the processing workload is distributed across the network. Distributed Systems are required for Functional distribution, Inherent distribution in application domain, Economics, Better performance, and increased Reliability.

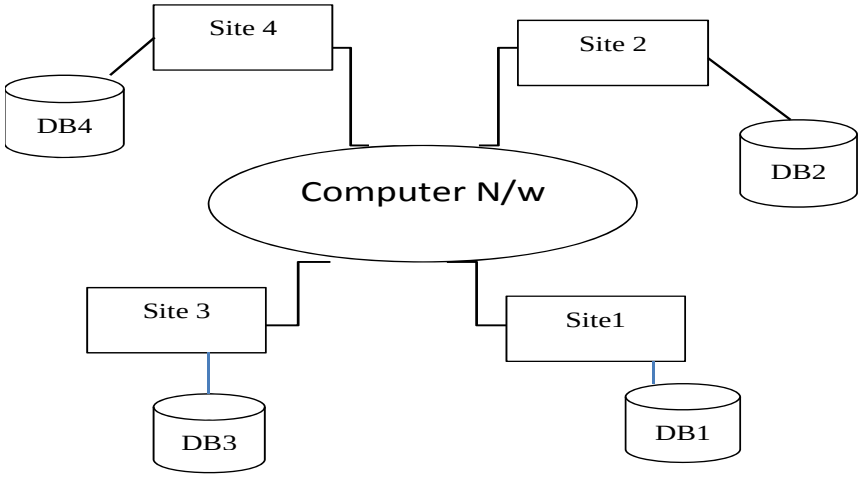

Fig. 2: Distributed Database System.

2. Requirement of Distributed Database Systems

One of the major objectives of Distributed database system is providing the appearance of centralized system to end user. The eight transparencies are believed to incorporate the desired functions of a distributed database system [6]. Such an image is accomplished by using the following transparencies: Location Transparency, Performance Transparency, Copy Transparency, Transaction Transparency, Transaction Transparency, Fragment Transparency, Schema Change Transparency, and Local DBMS Transparency. Other objective of distributed database is free object naming. Free object naming is basically allowing different users to access the same object with different names, or different objects with the same internal name. This will provide complete freedom to name the objects while sharing data without naming

conflicts. Another objective of distributed system is Concurrency control. Concurrency control is the activity of coordinating concurrent accesses to a database in a multi-user database management system (DBMS).

3. Types of Distributed Database Systems

Distributed Database Systems are broadly classified into two types[7,8]:

- Homogeneous Distributed System - In Homogenous distributed database system, the data is distributed but all servers run the same Database Management System(DBMS) software
- Heterogeneous Distributed System–In Heterogeneous distributed databases different sites run under the control of different DBMSs, These databases are connected somehow to enable access to data from multiple sites.

4. Advantages of Distributed Databases

Following are the various advantages of distributed databases[9,10]:

- Robust–A problem in one part of the organization will not stop other branches working.
- Security- Staff access can be restricted to only their portion of databases.
- Network traffic is reduced, thus reducing the bandwidth cost.
- Local database still works even if the company network is temporarily broken.
- High Performance–Queries and updates are largely local so that there is no network bottleneck.
- In distributed systems it is easier to keep errors local rather than the entire organization being affected.

5. Disadvantages of Distributed Databases

Following are the various disadvantages of distributed databases [9, 10]:

- Complexity-A distributed database is more complicated to setup and maintain as compared to central database system.
- Security–There are many remote entry points to the system compared to central system leading to security threats.
- Data Integrity–In distributed system it is very difficult to make sure that data and indexes are not corrupted.
- In distributed database systems, data need to be carefully placed to make the system as efficient as possible.
- Distributed databases are not so efficient if there is heavy interaction between sites.

6. Component of Distributed Database Systems

- Distributed Database System consists of the various components (Fig. 3). Database manager is one of major component of Distributed Database systems. Database Manager is software responsible for handling a segment of the distributed database. User Request Interface is another important component of distributed database systems. It is usually a client program which acts as an interface to the Distributed Transaction Manager.
- Distributed Transaction Manager is a program that helps in translating the user requests and converting into format required by the database manager, which are typically distributed. A distributed database system is made of both the distributed transaction manager and the database manager.

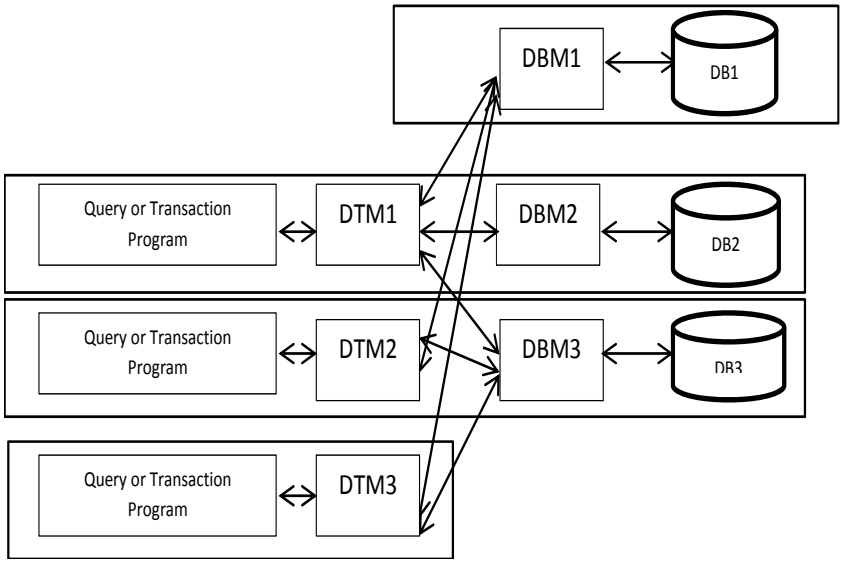

Fig. 3: Component Diagram of Distributed Databases.

7. Types of Distributed Database Systems

In order to access the data stored at remote location with less message passing cost, data should be distributed accordingly. Distribution of data is done through Fragmentation or Replication.

Fragmentation [11]: Fragmentation consists of breaking a relation into smaller relations or fragments and storing the fragments, possibly at different sites. Fragmentation of data in distributed database has four major advantages:

Efficiency: Data are stored close to where they are used and separate from other data used by other users or applications.

Local optimization: Data can be stored to optimize performance for local access.

Ease of querying: Combining data across horizontal partitions is easy because rows are simply merged by unions across the partitions.There are two types of fragmentation namely –

- Horizontal fragmentation: Each fragment consists of a subset of rows of the original relation(Fig. 4). It divides the "Rows" of R where R = R1 U R2 U …

R(a, b, c) -> R1(a, b, c), R2(a, b, c), … (1)

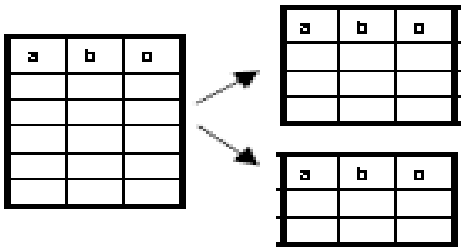

Fig. 4: Horizontal Partitioning.

- Vertical fragmentation: Each fragment consists of a subset of columns of the original relation (Fig. 5). It divides the "Columns" of R. R =R1 ⋈R2 ⋈R3… [3]

R(a, b, c) -> R1(a, b), R2(a, c), … (a is the primary key)R = R1 U R2 U … [4]

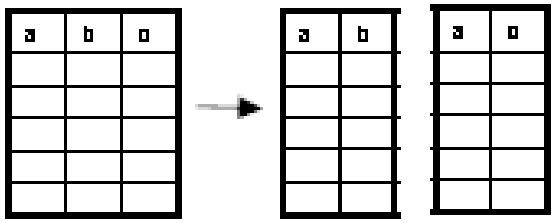

Fig. 5: Vertical Partitioning.

Replication [12, 13]: In Replication several copies of a relation are stored at different sites. Replication will help in increasing reliability, locality and performance. Various advantages of replication are as follows:

- Reliability: If one of the sites containing the relation (or database) fails, a copy can always be found at another site without network traffic delays
- Fast response: Each site that has a full copy can process queries locally, so queries can be processed rapidly.
- Possible avoidance of complicated distributed transaction integrity routines: Replicated databases are usually refreshed at scheduled intervals, so most forms of replication are used when some relaxing of synchronization across database copies is acceptable.
- Node decoupling: Each transaction may proceed without coordination across the network. Thus, if nodes are down, busy, or disconnected (e.g., in the case of mobile personal computers), a transaction is handled when the user desires.
- Reduced network traffic at prime time: Often updating data happens during prime business hours, when network traffic is highest and the demands for

rapid response greatest. Replication, with delayed updating of copies of data, moves network traffic for sending updates to other nodes to non-prime-time hours.

There are two types of replication which are as follows:

1. Synchronous Replication: All copies of a modified relation (fragment) must be updated before commit. Here, the most up to date value of an item is guaranteed to the end user. There are two different methods of synchronous replication.
   a) *Read-Any, Write-All:* This method is beneficial in case well when reads are much more frequent than writes.
      Read-Any: when reading an item, access any of the replicas.
      Write-All: when writing an item, must update all of the replicas.
   a) Voting:
   - When writing, update some fraction of the replicas.

When reading, read enough copies to ensure you get at least one copy of the most recent value.

Use a version number to determine which value is most recent the copies "vote" on the value of the item

2. Asynchronous Replication: Asynchronous replication allows different copies of the same object to have different values for short periods of time. Data is updated after a predefined interval of time.
   a) *Primary Site:* In primary-site replication, one copy of data is assigned as the master copy. Updation of data is possible only with in the master copy. The secondary copies of data can only be read. Changes to the master are periodically propagated to the secondary copies.
   b) *Peer-to-Peer:* In peer-to-peer replication, more than one replica is updatable. In addition, a conflict resolution strategy must be used to deal with conflicting changes made at different sites.

8. Problems In Distributed Database Systems

One of the major problems in distributed systems is deadlock. A deadlock is a state where a set of processes request resources that are held by other processes in the set and none of the process can be completed [14, 15, 16]. One process can request and acquire resources in any order without knowing the locks acquired by other processes. If the sequence of the allocations of resources to the processes is not controlled, deadlocks can occur. Hence we focus on deadlock detection and removal.

- **Deadlock Detection**
  In order to detect deadlocks, in distributed systems, deadlock detection algorithm must be used. Each site maintains a local wait for graph. If there is any cycle in the graph, there is a deadlock in the system.
  Even though there is no cycle in the local wait for graph, there can be a deadlock. This is due to the global acquisition of resources. In order to find the

global deadlocks, global wait for graph is maintained. This is known as centralized approach for deadlock detection.

The centralized approach to deadlock detection, while straightforward to implement, has two main drawbacks. First, the global coordinator becomes a performance bottleneck, as well as a single point of failure. Second, it is prone to detecting non-existing deadlocks, referred to as phantom deadlocks.

- **Deadlock Recovery**

A deadlock always involves a cycle of alternating process and resource nodes in the resource graph. The general approach for deadlock recovery is process termination. In this method, nodes and edges of the resource graph are eliminated. In Process Termination, the simplest algorithm is to terminate all processes involved in the deadlock. This approach is unnecessarily wasteful, since, in most cases, eliminating a single process is sufficient to break the deadlock. Thus, it is better to terminate processes one at a time, release their resources, and check at each step if the deadlock still persists. Before termination of process following parameters need to be checked:

a) The priority of the process:
b) The cost of restarting the process
c) The current state of the process

9. Conclusion

In the current scenario of the fast changing world, distribution of data became the necessity. Distribution of data has its own advantages and disadvantages. This paper presents a complete review on distributed databases. It is clear from the study that distribution of data involves the problem of deadlock. We need to find out the methods to data distribution and accessing which leads to minimization of deadlock and thus resulting in proper utilization of resources.

References

[1] www.businessdictionary.com/definition/database.html
[2] en.wikipedia.org/wiki/Database
[3] ib2012itgs.wikispaces.com/file/.../ITGS%20databasde%20homework.pdf.
[4] www.webopedia.com/TERM/D/distributed_database.html
[5] www.inf.unibz.it/dis/teaching/DDB/ln/ddb01.pdf
[6] www.teach-ict.com/as_a2_ict.../A2.../distributed_database/
[7] www.webopedia.com/TERM/D/distributed_database.htm
[8] wps.pearsoned.co.uk/wps/media/objects/.../Chapter%2012_WEB.pdf
[9] https://www.dlsweb.rmit.edu.au/toolbox/Database/.../dist_sys_def.htm
[10] www.csc.liv.ac.uk/~dirk/Comp332/COMP332-DDB-notes.pdf

[11] http://stackoverflow.com/questions/5777234/horizontal-vs-vertical-fragmentation-in-distributed-database-management-systems.
[12] www.cse.iitb.ac.in/~sudarsha/db-book/slide-dir/ch22.ppt
[13] Xiangning Liu, Bharat K. Bhargava, “Data Replication in Distributed Database Systemsover Large Number of Sites”,Computer Science Technical Reports. Paper 1229
[14] en.wikipedia.org/wiki/Deadlock
[15] X. M. Chandy and J. Misra, "A Distributed Algorithm for Detecting Resource Deadlocks in Distributed Systems " in ACM, 1982.
[16] B. M. M. Alom, F. Henskens, and M. Hannaford, "Deadlock Detection Views of Distributed Database," in International conference on Information Technology & New Generartion (ITNG- 2009) Las Vegas, USA: IEEE Computer Society, 2009.