Volume : 4 | Issue : 11 | November 2015 • ISSN No 2277 - 8179

Research Paper

An Overview of Distributed Data Base

Computer Science

KEYWORDS : Database, Distributed Database Management System, Fragmentation, Replication, Deadlock

Ms. Tanvi Rajeshbhai Raval

Asst. Professor, Laxmi Institute of Commerce & Computer Application

ABSTRACT

In database distribution system anything can be considered as data e.g. number, person name, phone no etc. when a data is meaningful, it is called information. The database is an organized collection of related information. Database management system is collection of interrelated data and set of application program to access those data. A database is controlled by Database Management system (DBMS) by maintaining and utilizing large collection 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 brief a distributed database is a collection of databases that can be stored at different computer network sites. In this paper I have included an overview of Distributed Database System along with their advantages and disadvantages, their characteristics and its implication in practical working. Further in this paper I have emphasized and worked out on various aspects like fragmentation, replication and various problems that can be faced in Distributed Database Systems.

Introduction:

A Database is systematically organized or structured repository of indexed information that allows easy retrieval, updating, analysis, and output of data. Each database may involve different database management systems and different architecture that distribute the execution of transactions. A distributed database is a database in which storage devices are not all attached to a central 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 Means a Distributed Database is a collection of multiple logically interrelated databases distributed over a computer network.

A distributed databases 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 = DDB + D-DBMS)

In Centralized Systems Data, Process and interface components of an information system are central. [Diagram 1]

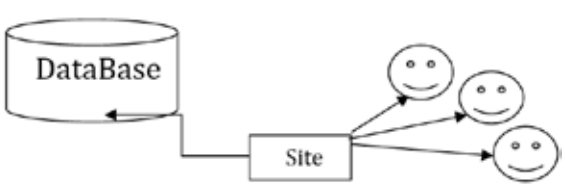


Diagram 1: Centralized Database System.

In order to work on the system end users uses terminals or terminal emulators. In Distributed

System [3, 4, 5] [Diagram 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.

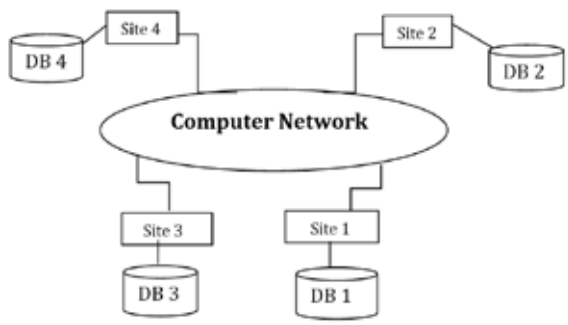


Diagram 2: Distributed Database System.

What is not DDBMS?

- A timesharing computer system
- A loosely or tightly coupled multiprocessor system
- A database system which resides at one of the nodes of a network of computers.

Requirement of Distributed Database System:

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. 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).

Characteristics of a DDBMS:

A DDBMS developed by a single vendor may contain:

- Data independence
- Concurrency Control
- Replication facilities
- Recovery facilities
- Coordinated Data Dictionary
- Authorization System
- Shared Manipulation Language

Volume : 4 | Issue : 11 | November 2015 • ISSN No 2277 - 8179

Also:
- **Transaction Manager (TM)**
- **Data Manager (DM)**
- **Transaction Coordinator (TC)**

Distributed Data Processing System is a system where the application programs run on distributed computers which are linked together by a data transmission network.

Types of Distributed Database System:
Homogeneous DDBMS: This is the case when the application programs are **independent** of how the db is distributed; i.e. if the distribution of the physical data can be altered without having to make alterations to the application programs. Here, all sites use the same DBMS product - same schemata and same data dictionaries.

In Short, In Homogeneous Distributed Database Management system, the data is distributed but all servers run **the same Database management system** software.

Heterogeneous DDBMS: This is the case when the application programs are **dependent on** the physical location of the stored data; i.e. application programs must be altered if data is moved from one site to another. Here, there are different kinds of DBMSs (i.e. Hierarchical, Network, Relational, Object., etc.), with different underlying data models.

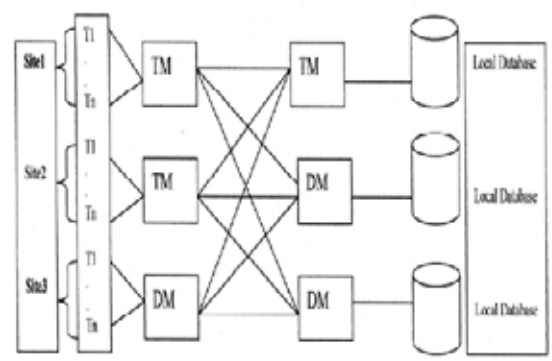
In short, In Heterogeneous Distributed Databases different sites run under the control of **different Database management system** software, these databases are connected somehow to enable access to data from multiple sites.

Advantages of Distributed Databases:
Following are the various advantages of distributed databases:
- 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.

Disadvantages of Distributed Databases:
Following are the various disadvantages of distributed databases:
- 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.

Architecture of the Distributed Database Management System:
Each computer (site) in a **distributed system** may contain a **Transaction Manager (TM)** and a **Data Manager (DM)** there is also a **Transaction Coordinator (TC)**. The TM is responsible for the Transactions received by the computer. The DM manages the db access on the local computer.

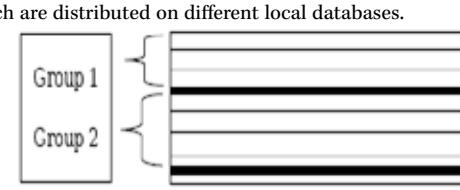
When a Transaction arrives at the TM, the TM divides the transaction into sub-transactions which are transmitted to those DMs containing the data needed by the Transaction. (In some cases the TC is responsible for this.) The TM processes the collected received data from the sub-transactions' responses and produces the final result. Any TM can communicate with all DMs and vice versa.



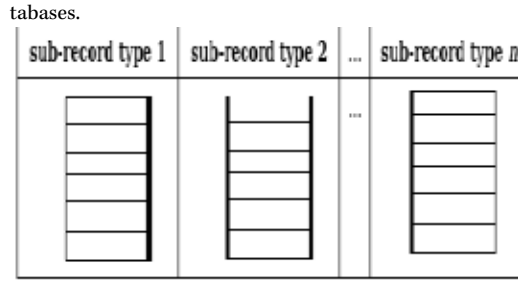
The DMs cannot transmit data to other DMs and the same applies to TMs, except in certain cases where it is convenient to transfer the total responsibility of a Transaction from one TM to another (i.e. if a Transaction runs as a local Transaction on another computer.)

Principles of Data Distribution:
The **distribution of data** can be based on entire files or on the subdivisions of files. There are the following subdivision types: Horizontal, Vertical and Hybrid subdivisions.

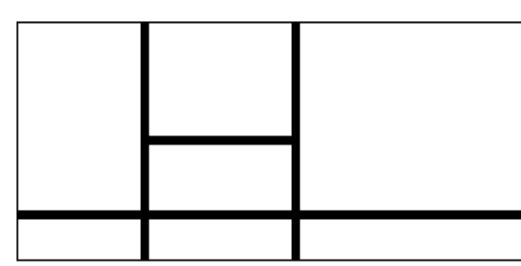
Horizontal: The records of the files are divided into groups which are distributed on different local databases.



Vertical: A logical record type is divided into two or more physical record types, which can be distributed on different local Databases.



Hybrid: Uses both horizontal and vertical



Fragmentation:
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: 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 is stored close to where it is most frequently used. In addition, data that is, not needed by' local applications is not stored.

Parallelism
With fragments as the unit of distribution, a transaction can be divided into several sub queries that operate on fragments. This should increase the degree of concurrency, or parallelism, in the system, thereby allowing transactions that can do so safely to execute in parallel.

Security
Data not required by local applications is not stored, and consequently not available to unauthorized users.

Disadvantages of fragmentation:
Fragmentation has two primary disadvantages, which we have mentioned previously:

Performance
The performance of global application that requires data from several fragments located at different sites may be slower.

Integrity
Integrity control may be more difficult if data and functional dependencies are fragmented and located at different sites.

The types of fragmentations are:
Division of relation r into fragments r1 , r2 , ..., rn which contain sufficient information to reconstruct relation r.

Horizontal fragmentation: Each fragment consists of a subset of rows of the original relation [Diagram 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), ...

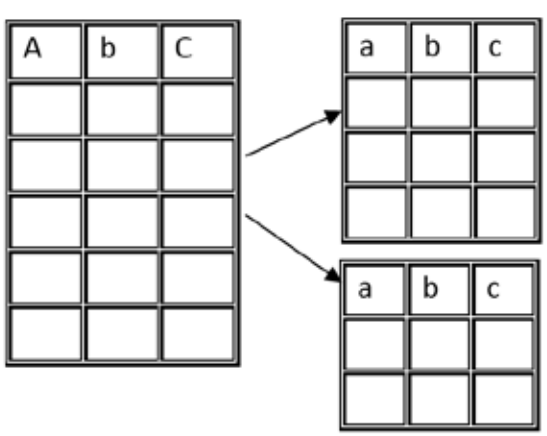


Diagram 4: Horizontal Partitioning

Vertical fragmentation: Each fragment consists of a subset of columns of the original relation [Diagram 5]

It divides the "Columns" of R.

R =R1 × R2 × R3...

R(a, b, c) -> R1(a, b), R2(a, c), ... (a is the primary key) R = R1 U R2 U ...

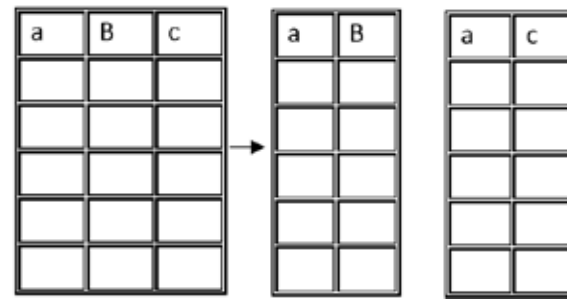


Diagram 5: Vertical Partitioning

Mixed Fragmentation: Vertical and horizontal fragmentation can be mixed. Fragments may be successively fragmented to an arbitrary depth.

Example of Fragmentation:
Suppose we have the relation

Deposit (Branchname, Account#, CustomerName, Balance, Tupleid)

BranchName	**Account#**	**CustmerName**	**Balance**	**Tupleid**
Wonderland	305	H.H.	10	1
Wonderland	226	C.B.	5	2
Moonland	177	C.B.	10	3
Moonland	402	B.B.	2	4
Wonderland	155	B.B.	10	5
Moonland	408	B.B.	100	6
Moonland	639	M.M.	1	7

Horizontal Fragmentation Would Produce:
Deposit 1

BranchName	Account#	CustmerName	Balance	Tupleid
Wonderland	305	H.H.	10	1
Wonderland	226	C.B.	5	2
Wonderland	155	B.B.	10	5

Deposit 2

BranchName	Account#	CustmerName	Balance	Tupleid
Moonland	177	C.B.	10	3
Moonland	402	B.B.	2	4
Moonland	408	B.B.	100	6
Moonland	639	M.M.	1	7

SUB1 BranchName	Account#	Tupleid
Wonderland	305	1
Wonderland	226	2
Moonland	177	3
Moonland	402	4
Wonderland	155	5
Moonland	408	6
Moonland	639	7

SUB2 CustmerName	Balance	Tupleid
H.H.	10	1
C.B.	5	2
C.B.	10	3
B.B.	2	4
B.B.	10	5
B.B.	100	6
M.M.	1	7

Vertical Fragmentation Would Produce:

Replication:

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.

Two types of replications:

Synchronous replication – all copies of a modified relation are updated before the modifying transaction commits.

Asynchronous replication – copies of modified a relation are updated over a period of time, and a transaction that reads different copies of the same relation may see different values.

- Widely used in commercial distributed DBMSs
- Users must be aware of distributed databases

Synchronous Replication:

Voting technique – a transaction must write a majority of copies to modify an object; read at least enough copies to make sure one of the copies is current.

- 10 copies, 7 are updatable, 4 are read
- Each copy has a version number, the highest is the most current.
- Not attractive and efficient, because reading an object requires reading several copies. Objects are read more than they are updated.

- **Read-any-write-all technique** – a transaction can read only one copy, but must write to all copies.
- Reads are faster than writes especially if it's a local copy
- Attractive when reads occur more than writes
- Most common technique
- Read-any-write-all cost - Before an update transaction can commit, it must lock all copies
- Transaction sends lock requests to remote sites and waits for the locks to be granted, during a long period, it continues to hold all locks.
- If there is a site or communication failure than transaction cannot commit until all sites are recovered
- Committing creates several additional messages to be sent as part of a commit protocol
- Since synchronous replication is expensive, Asynchronous replication is gaining popularity even though different copies can have different values.

Asynchronous Replication:

Allows modifying transactions to commit before all copies have been changed

- Users must be aware of which copy they are reading, and that copies may be out-of-sync for short periods of time.

Two approaches: Primary Site and Peer-to-Peer replication.

- Difference lies in how many copies are ``updatable" or ``master copies".

Peer to Peer Asynchronous Replication:

- More than one copy can be designated as updateable (master copy).
- Changes to a master copy must be propagated to other copies somehow.
- Conflict resolution is used to deal with changes at different sites.
- Each master is allowed to update only 1 fragment of the relation, and any two fragments updatable by different masters are disjoint
- Updating rights are held by one master at a time.

Primary site – one copy of a relation is the master copy

Problems in Distributed Database:

In a database environment, a deadlock is a situation where transactions are waiting for one another. Any lock based concurrency control algorithm and some timestamp based concurrency control algorithms may result in deadlocks, since these algorithms require transactions to wait for one another. In lock based concurrency control algorithms, locks on data items are acquired in mutually exclusive manner, thus, it may cause deadlock situation. Whenever a deadlock situation arises in a database environment, the outside interference is required to continue the normal execution of the system. Therefore, the database system requires special procedures to resolve the deadlock situation. Deadlock situation can be characterized by **wait-for graphs**, directed graphs that indicate which transactions are waiting for which other transactions. In a wait-for graph, nodes of the graph represent transactions and edges of the graph represent the waiting for relationships among transactions. An edge is drawn in the wait-for graph from transaction Ti to transaction Tj if the transaction Ti is waiting for the lock on a data item that is currently hold by the transaction Tj. Using wait-for graph, it is very easy to detect whether a deadlock situation has occurred in a database environment or not. There is a deadlock in the system if and only if the corresponding wait-for graph contains a cycle. The resolution of deadlock situation is much easier in centralized DBMS than that of distributed DBMS. In a centralized DBMS, only one local wait-for graph is drawn to detect the deadlock situation. The detection of deadlock in a distributed DBMS is more complicated, because the circular waiting situation which determines a deadlock situation may involve several different sites. Thus, in a distributed DBMS it is not sufficient to draw a local wait-for graph for each local DBMS only, but it is also necessary to draw a **global wait-for graph** for the entire system to detect deadlock situation. In a distributed database, a local wait-for graph is a portion of the global wait-for graph which consist only those nodes and edges that are completely contained at a single site.

Conclusion:

In the current situation of the fast changing world, distribution of data becomes the essential. Distribution of data has its own

advantages, disadvantages and characteristics. This study presents a complete review on distributed databases and its implication. It also includes the components of distributed database, principle of distribution, fragmentation of the database. It also represents the replication, types of replication and problem in distributed database, this study also measure the need to find out the methods which leads to minimization of deadlock and thus resulting in proper utilization of resources.

REFERENCE

[1] cs.njit.edu/cis731s06/presentations/Distributed_Databases.ppt [2] https://docs.oracle.com/cd/A58617_01/server.804/.../ch_repli.htm [3] http://searchsqlserver.techtarget.com/definition/database-replication [4] https://cs.uwaterloo.ca/~tozsu/courses/cs856/F02/lecture-1-ho.pdf [5]http://www.uio.no/studier/emner/matnat/ifi/INF5100/h11/undervisningsmateriale/distributed_dbs.pdf [6] https://www.cse.iitb.ac.in/~sudarsha/db-book/slide-dir/ch22.pdf [7] https://www.dlsweb.rmit.edu.au/toolbox/Database/.../dist_sys_def.htm [8] www.csc.liv.ac.uk/~dirk/Comp332/COMP332-DDB-notes.pdf [9] http://stackoverflow.com/questions/5777234/horizontal-vs-verticalfragmentation-in-distributed-database-management-systems [10] en.wikipedia.org/wiki/Deadlock [11] http://codex.cs.yale.edu/avi/db-book/db6/slide-dir/\ [12] http://www.scribd.com/doc/18110136/Distributed-Database-Management-Notes-2#scribd [13] X. M. Chandy and J. Misra, "A Distributed Algorithm for Detecting Resource Deadlocks in Distributed Systems " in ACM, 1982. [14] http://www. irphouse.com /ijict.htm