OVERVIEW OF DISTRIBUTED DATABASE SYSTEM

Okardi, Biobele[1]
Department of Computer Science
Federal University Otuoke,
Bayelsa State, Nigeria
bokardi2001@gmail.com[1]

Asagba, O. Prince[2]
Department of Computer Science
University of Port Harcourt
Rivers State, Nigeria
asagba.prince@uniport.edu.ng[2]

Abstract

Distributed Database System (DDBS) is a collection of multiple, logically interrelated databases distributed over a computer network. It is a schema that facilitates database decentralization for the management and manipulation of data in a database using same or common language. The business environment is presently in need of distributed database and Client/server applications for reliability, scalability and accessibility of information which rises in Steady manner. Distributed database systems provide improved communication in data processing throughout different network sites. The Database is controlled by Database Management System (DBMS) in maintaining and utilizing large collections of data. This paper tends to present an overview and introduction to Distributed Database System and their advantages and disadvantages, and also provides various aspects like replication, fragmentation and various problems that can be faced in distributed database systems. This research proposes a systematic review on distributed databases systems with the aid of three distribution strategies respectively - Data fragmentation, Data allocation and Data replication. Some problems encountered in the design and use of these strategies has been pointed out. For Distributed System, the hardware and software components in the network communicate and coordinate their activity only by passing messages.

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

Introduction

Databases in today's world are very essential to every business. In every major website visited, such as - Google, Yahoo, Amazon, Jumia, Konga, or other thousands of small sites that provide information, there is a database behind the scene serving up the information requested. The strength of a database comes from a body of knowledge and technology known as Database Management System or A DBMS (Garcia-Molina 2009). For any organization to function properly there is need for a well-maintained database system to enable its business processes and ease storage of data. In recent past, centralized database were used for data storage and retrieval. Later the distributed database introduction came into limelight and it has shown high level performance and efficiency in data processing and management within large organizations. The introduction of Distributed database system was due to increase in globalization and organizations tend to diversify across the globe (Tutorialpoint 2016).

The distributed database is said to be centrally administered as a corporate resource while providing local flexibility and customization. The network tend to allow the users to share the data, such that a user (or program) at location A must be able to access (and also update) data at location B. The sites of a distributed system may be span over a large geographical area (e.g., the United States or the world) or over a small area (e.g., a building or campus). Computers in the network range from PCs to large-scale servers or even supercomputers.

The Evolution of Distributed Database Management Systems ((DDBMS)

Distributed database management system (DDBMS) tends to govern the storage and processing of data that are logically related over interconnected computer systems where both the data and its processing are distributed among several sites. In order to understand how and why the Distributed Database Management System (DDBMS) is different from the DBMS, it is useful to briefly view and examine the changes in the business environment that set the platform for the development of the DDBMS.

In the early 1970s, organizations tend to implement centralized database management systems to meet their well-structured information needs. Its use requires that, corporate data is stored in a single central site, usually a mainframe computer. And data access was provided through dumb terminals. Two decades after the introduction centralized database pave way for various crucial social and technological changes (such as customers' demands and market needs, increase in sophistication of mobile devices) that affected both the systems and its data usage (Coronel & Morris 2019). The long term impart of the internet and mobile revolution on the design of a distributed database is been felt and, the business environment and

the shortfalls of the centralized database has given rise to the need for an application that would access data from different sources at multiple location. And such database is best managed by a DDBMS.

The centralized approach, illustrated in Figure 1, worked well to fill the structured information needs of corporations, but it fell short when quickly moving events required faster response times and equally quick access to information. In dynamic environment the slow progression from information request to approval to specialist to user simply does not serve decision makers well. What was needed was quick, unstructured access to databases, using ad hoc queries to generate on-the-spot information.

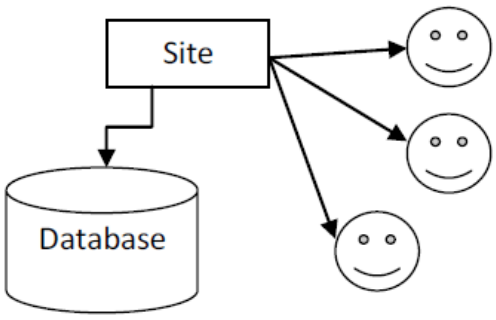

Figure 1: Centralized Database (Tomar & Megha 2014).

Databases are very essential tool in data organization and management. They are considered as the nucleus of most system, especially management systems. Databases are used as repository for the user's data. It is also responsible for restructuring data in a more efficient manner. A database capable of maintaining data consistently and also store data efficiently.

What is a Distributed Database System?

A Distributed Database System (DDBS)is simply a collection of multiple, and interrelated databases that are distributed over a computer network. It is a relational schema that facilitates database decentralization for the management and manipulation of data in a database with the aid of same or common language **(Abbas et al 2016).** Distributed database is a single logical database that span physically across computers in multiple locations and it is connected through data communication links (Sagar & Swapnil 2016). It is simply a virtual database whose component are physically stored in a number of distinct real databases in distinct locations, enabling users to access data anywhere within the network (Kaur & Kaur 2013). According to Katembo et al (2019), most organizations are motivated to implement efficient Distributed Database Systems (DDBS) for the purpose of scalability.

What is Distributed Database Management System DDBMS**?**

(D–DBMS) is a software that manages the distributed database system (DDB) and provides an access mechanism that makes this distribution transparent to the users. Distributed database system (DDBS) = DDB + D–DBMS (Ozsu & Valduriez 2019). The objective of DDBMS is to control the management of a distributed database (DDB) in such a way that makes it appears to the user as a centralized database.

Various DBMS solutions from various vendors exist in the market. Among the vendors, three of them shares two- third of the market with regards to database systems. Their popularity has spread all over the global environment. They include Oracle, Microsoft server and IBM DB2 Universal Database. These three vendors tend to offer different versions or editions of their solution which is capable of catering for different businesses' needs. The three main vendors Oracle 11g, Microsoft SQL server 2005 and IBM DB2 version 9.5 (Grech 2009).

Distributed Processing and Distributed Databases

In distributed processing, the processing (logical) is shared among two or more physically independent sites that are connected through various communication links. For example, the data input/output (I/O), , data validation and data selection might be performed on one computer, and a report based on the data is created on another computer. A distributed database, on the other hand, stores related database over two or more physically independent sites. The sites are connected through a computer (Coronel & Morris 2019).

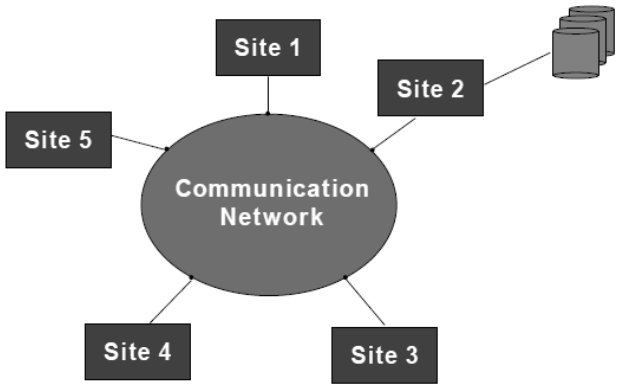

Figure 2: Centralized DBMS on a Network (Ozsu & Valduriez 2019)

Advantages of Distributed Database System

Comparing Distributed database with centralized Database system with Distributed database shows that Distributed database system has numerous advantages. The following are some important advantages of Distributed databases:

- **Increased reliability and availability**

  When a centralized system fails, the database is unavailable to all users but a distributed system will continue to function at some reduced level, however, even when a component fails. For instance, if the data and DBMS software tend to be distributed over several sites, and if a site fails, the other sites will continue to function properly and we can equally access data that exist on the failed site. This basically leads to improvement in the reliability and availability of the database system.

- **Modular growth**

  If an organization expands to a new location or adds a new work group. It is basically easier and more economical to add a local computer and its associated data to the distributed network than to expand a large central computer.

- **Lower communication costs**

  A distributed system data can be located closer to their point of use and that can reduce communication costs, compared with a central system.

- **Local control**

  Distributing the data encourages local groups to exercise greater control over their data, which promotes improved data integrity and administration. Users can access non local data when necessary.

- **Faster response**

  In distributed database requests for data by users at a particular site can be satisfied by data stored being stored in the site. This speeds up query processing since communication and central computer delays are minimized

- **Easier Expansion**

  It is much easier to expand the distributed system environment in terms of adding more data, increase the database size and add more processor capability.

- **Improved Performance**

  Multiple queries can be executed at different sites, by breaking them into a number of subqueries that basically executes in parallel. This can be achieved through interquery and intraquery parallelism by executing multiple queries at different sites by breaking up a query into a number of subqueries that simply executes in parallel which basically leads to improvement in performance.

Disadvantages of Distributed Database System

Distributed database system also faces certain costs and disadvantages:

• **Software cost and complexity**

A distributed database is more complicated to maintain and setup as compared to central database system. A more complex form of softwares (especially the DBMS) are required for a distributed database environment.

• **Processing overhead:** It is mandatory for various sites to exchange messages and perform additional calculations to ensure proper coordination among data at the different sites.

• **Data integrity: The** Increased complexity of a distributed database is obvious and it becomes very difficult in ensuring data indexes are not corrupted.

• **Slow response:** Response to request for data can be extremely slow due to the fact that data are not properly distributed according to their usage.

• **Security**: The security of a distributed database system tend to be vulnerable due to the fact that there are many entry points to the to the system..

• The efficiency of the distributed database system tend to reduce due to heavy interaction between sites in the database.

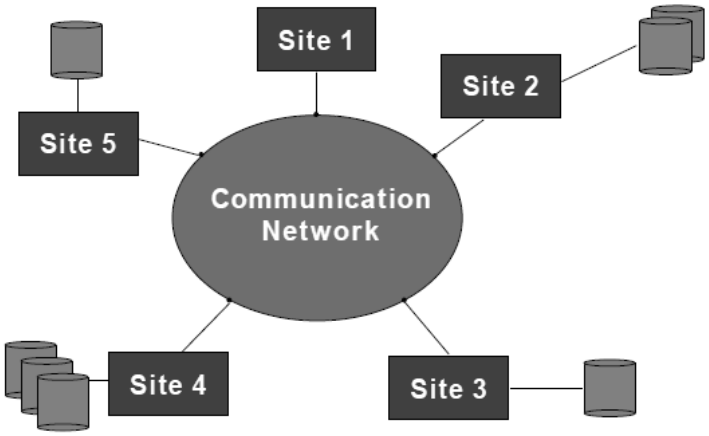

Figure 3: Distributed DBMS Environment (Özsu & Valduriez 2019)

Component of Distributed Database Systems

Distributed Database System consists of the various components

- **Database manager:** a software responsible for handling a segment of the distributed database. It is a major components of Distributed Database systems.
- **User Request Interface:** Is a program (Client) that acts as an interface to the Distributed Transaction Manager.

- **Distributed Transaction Manager**: It is a program that translates the user requests and also convert it into a 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 (Tomar & Megha (2014).

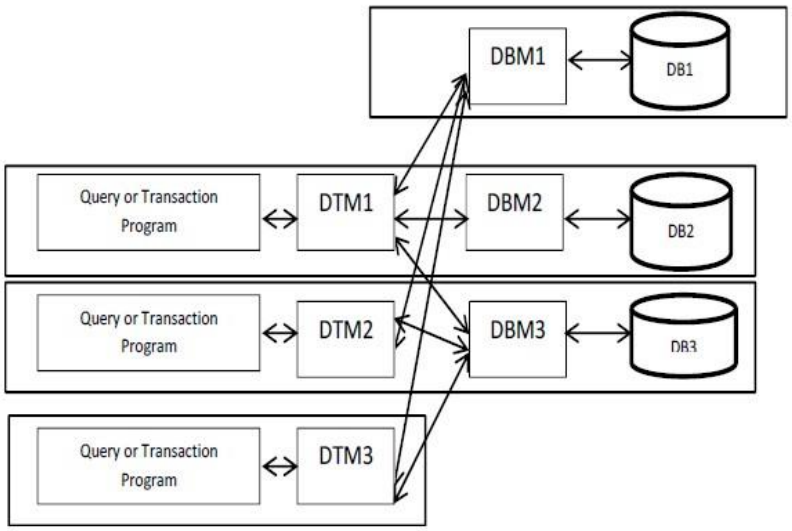

Figure 4: Component Diagram of Distributed Databases (Tomar & Megha 2014).

Issues Affecting Distributed DBMS

Distributed Database Design

- how to distribute the database
- replicated & non-replicated database distribution
- a related problem in directory management

Query Processing

- convert user transactions to data manipulation instructions
- optimization problem

Concurrency Control

- synchronization of concurrent accesses
- consistency and isolation of transactions' effects
- deadlock management

Reliability

- how to make the system resilient to failures
    - atomicity and durability

Concurrency Control

- synchronization of concurrent accesses
- consistency and isolation of transactions' effects
- deadlock management

Distributed Database

For Distributed systems, unlike a centralized database system data is distributed among different database systems of an organization. These database systems are connected to one another via communication links. Such links help the end-users to access the data easily. We basically have two types of Distributed Database which include Homogenous and Heterogeneous DDBS.

1. Homogeneous DDB

These are database systems tend to execute on the same operating system, use same application process, and carries same hardware devices. Every site runs same type of DBMS**.** The same DBMS is used at each node. Types of Homogeneous DDB include Autonomous and Non Autonomous DDB

a. Autonomous DDB: Each DBMS works independently, passing messages back and forth to share data updates.

b. Non autonomous*:* A central, or master, DBMS coordinates database access and updates across the nodes.

2. Heterogeneous DDB

The heterogeneous Distributed DB systems tend to execute on different operating systems under varying application procedures, and carries different hardware devices. Different sites run different DBMSs different RDBMSs or even non-relational DBMSs. RDBMS = Relational DBMS. Potentially different DBMSs are used at each node.

2A. *Systems* Supports some or all of the functionality of one logical database.

i. *Full DBMS functionality:* *It* Supports all of the functionality of a distributed database

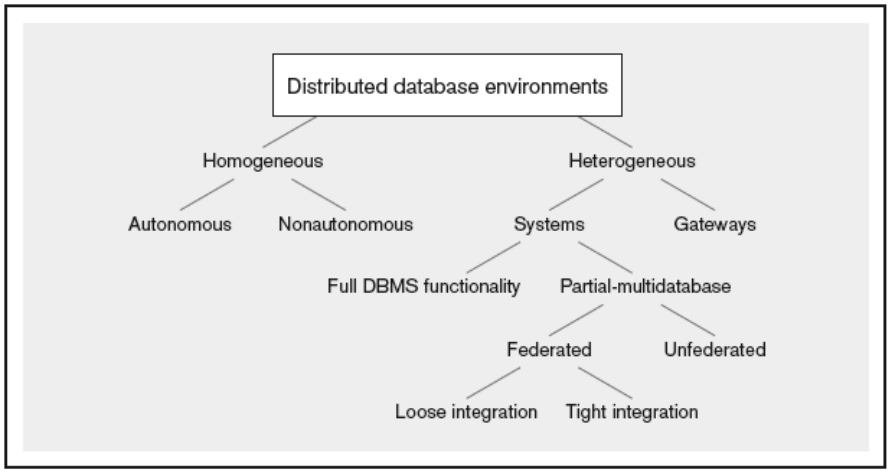

Figure 5: Distributed Database Environment

ii. Partial-multidatabase. It Supports some features of a distributed database,

a. Federated Distributed Database Supports local databases for unique data requests.

i. Loose integration: Many schemas exist, for each local database, and each local DBMS must communicate with all local schemas.

ii. Tight integration: A global schema exists that defines all the data across all local databases.

b. Unfederated: Requires all access to go through a central coordinating module.

2B. Gateways: Simple paths are created to other databases, without the benefits of one logical database.

A homogeneous distributed database environment is depicted in Figure 5.

The environment is typically defined by the following characteristics (related to the non-autonomous category described previously):

• Data are distributed across all the nodes.

• The same DBMS is used at each location.

• All data are managed by the distributed DBMS (so there are no exclusively local data).

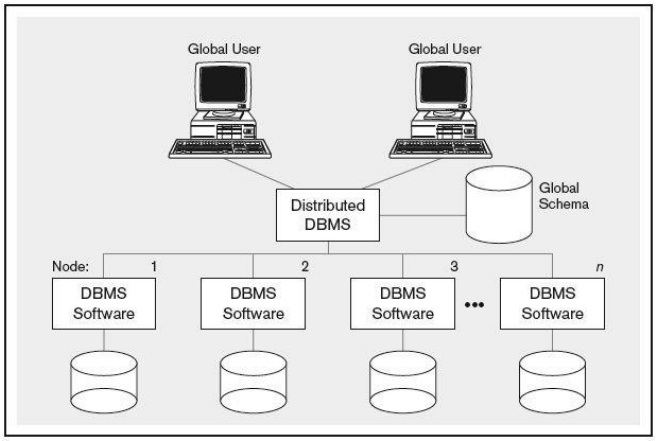

Figure 6: Homogeneous Distributed database environment (Jeffrey et al 2005)

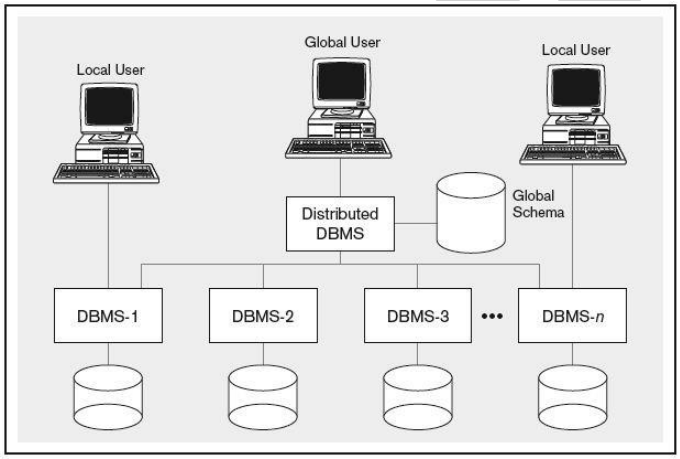

Figure 7: Heterogeneous Distributed database environment (Jeffrey et al 2005)

• All users access the database through one global schema or database definition.

• The global schema is simply the union of all the local database schemas. It is difficult in most organizations to force a homogeneous environment, yet heterogeneous environments are much more difficult to manage. There are many variations of heterogeneous distributed database environments. However, a heterogeneous environment will be defined by the following characteristics (as depicted in Figure 7).

• Data are distributed across all the nodes.

• Different DBMSs may be used at each node.

• Some users require only local access to databases, which can be accomplished by using only the local DBMS and schema.

• A global schema exists, which allows local users to access remote data.

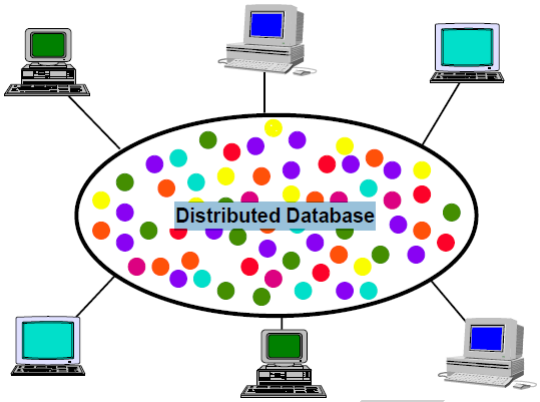

Figure 8: Distributed Database –User View

In distributed processing, a database logical processing is shared among two or more physically independent sites that are connected through a network.

A basic distributed processing environment shares the database processing chores among three sites connected through a communications network. Although the database resides at only one site (i.e Abuja), each site can access the data and update the database. A distributed database, on the other hand, stores a logically related database over two or more physically independent sites, that are connected via a computer.

Distributed Database Design

The design principles and concepts described are still applicable, whether the database is centralized or distributed. However, the design of a distributed database tends to introduce the following issues.

• How do we partition the database into fragments?

• Which fragments do we replicate?

• Where to do we locate those fragments and replicas

Data fragmentation and data replication really deal with the first two issues and data allocation deals with the third issue. Data in a distributed database should be evenly distributed to increase availability (reduce bottlenecks), maximize performance, and provide location awareness, which is an increasing requirement for mobile applications.

Data Fragmentation

Data fragmentation enables one to break a single object into two or more fragments, or segments. The object might be a user's database, a system database, or a table. Each fragment can be stored at any site on the computer network. Information gathered about data fragmentation is stored in the distributed data catalog (DDC), from which it is accessed by the TP to process user requests. Data fragmentation strategies

are based at the table level and consist of dividing a table into logical fragments. You will explore three types of data fragmentation strategies: horizontal, vertical, and mixed.

• **Horizontal fragmentation** is simply division of a relation into subsets (fragments) of tuples (rows). Each fragment is stored at a different node, and each fragment has unique rows. However, the unique rows all have the same attributes (columns). That is, each fragment tend to represent the equivalent of a SELECT statement, with the WHERE clause on a single attribute.

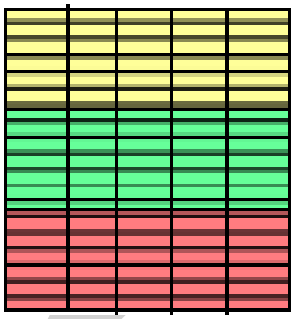

Figure 9: Horizontal Fragmentation (Goebel 2011).

• **Vertical fragmentation** is the division of a relation into attribute (column) subsets. Each subset (fragment) is stored at a different node, and each fragment has unique columns with the exception of the key column, which is common to all fragments. This is the equivalent of the PROJECT statement in SQL.

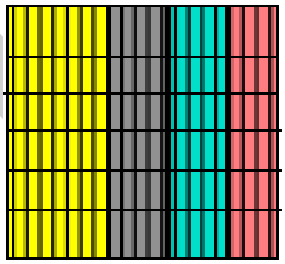

Figure 10: Vertical Fragmentation

• **Mixed fragmentation** is a combination of horizontal and vertical strategies. For instance, a table may be divided into several horizontal subsets (rows), each one having a subset of the attributes (columns) Goebel V. (2011).

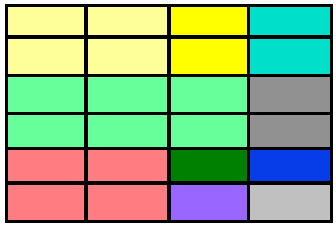

Figure 11: Mixed Fragmentation

Data Replication

Data replication can be defined as the storage of data copies at multiple sites served by a computer network. Fragment copies can be stored at various sites to serve specific information requirements. The existence of fragment copies can enhance data availability and response time, data copies can help to reduce communication and total query costs. For instance, if database A is divided into two fragments, A1 and A2. Within a replicated distributed database, the scenario depicted in Figure 11 is possible. Fragment A1 is stored at Sites S1 and S2, while fragment A2 is stored at Sites S2 and S3. Replicated data is subject to the mutual consistency rule, where is required that all copies of data fragments have be identical. The DDBMS must ensure that a database update is performed at all sites where replicas exist, in order to maintain data consistency among the replicas.

There are basically two styles of replication

We have the Push Replication and the Pull Replication

• ***Push replication*.** After a data update, the originating DP node sends the changes to the replica nodes to ensure that data is immediately updated. This type of replication focuses on maintaining data consistency. However, it decreases data availability due to the latency involved in ensuring data consistency at all nodes.

• ***Pull replication***. After a data update, the originating DP node sends "messages" to the replica nodes to notify them of the update. The replica nodes decide when to apply the updates to their local fragment. In this type of replication, data updates propagate more slowly to the replicas. The focus is on maintaining data availability. However, this style of replication allows for temporary data inconsistencies. Although replication has some benefits, such as improved data availability, improved data failure tolerance, better load distribution, and reduced query costs, it also imposes additional DDBMS processing overhead because each data copy must be maintained by the system.

Several Factors Influence the Decision to Use Data Replication

• **Database size**. The amount of data replicated will have an impact on the storage requirements and the data transmission costs.

• **Usage frequency.** How frequently the need for data to be updated is determined by the frequency of data usage. Frequently used data should be updated more often, especially on the large data sets that are used only on quarter basis.

• **Costs.** By Costs, we refer to cost for performance, software overhead, and management associated with synchronizing transactions and their components versus fault-tolerance benefits that are associated with replicated data. When the usage frequency of remotely located data is high and the database is large, data replication can reduce the cost of data requests.

Data Allocation

Data Allocation is a database distribution approach where DDB fragments are assigned on a distributed network sites. The data allocated can be or not replicated (single copy) or replicated (double copies) Katembo et al (2019). Data allocation tends to describe the process of deciding where data could be located.

Asynchronous distributed database Versus Synchronous distributed database

Asynchronous distributed database technology keeps copies of replicated data at different nodes so that local servers can access data without reaching out across the network. With asynchronous technology, there is usually some delay in propagating data updates across the remote databases, so some degree of at least temporary inconsistency is tolerated. Asynchronous technology tends to have acceptable response time because updates happen locally and data replicas are synchronized in batches and predetermined intervals, but may be more complex to plan and design to ensure exactly the right level of data integrity and consistency across the nodes. Compared with centralized databases, either form of a distributed database has numerous advantages. The following are the most important of them:

With synchronous distributed database technology, all data across the network are continuously kept up to date so that a user at any site can access data anywhere on the network at any time and get the same answer. With synchronous technology, if any copy of a data item is updated anywhere on the network, the same update is immediately applied to all other copies or it is aborted. Synchronous technology ensures data integrity and tends to minimize the complexity of knowing where the most recent copy of data is located.

DISTRIBUTED DBMS ADVANTAGES AND DISADVANTAGES	
ADVANTAGES	**DISADVANTAGES**
Data is located near the site of greatest demand. The data in a distributed database system is dispersed to match business requirements.	***Complexity of management and control.*** Applications must recognize data location, and they must be able to stitch together data from various sites. Database administrators must have the ability to coordinate database activities to prevent database degradation due to data anomalies.
Faster data access. End users often work with only the nearest stored subset of the data.	***Technological difficulty.*** Data integrity, transaction management, concurrency control, security, backup, recovery, and query optimization must all be addressed and resolved.
Faster data processing. A distributed database system spreads out the system's workload by processing data at several sites.	***Security.*** The probability of security lapses increases when data is located at multiple sites. The responsibility of data management will be shared by different people at several sites.
Growth facilitation. New sites can be added to the network without affecting the operations of other sites.	***Lack of standards.*** There are no standard communication protocols at the database level. For example, different database vendors employ different and often incompatible techniques to manage the distribution of data and processing in a DDBMS environment.
Improved communications. Because local sites are smaller and located closer to customers, local sites foster better communication among departments and between customers and company staff.	***Increased storage and infrastructure requirements.*** Multiple copies of data are required at different sites, thus requiring additional storage space.
Reduced operating costs. It is more cost-effective to add nodes to a network than to update a mainframe system. Development work is done more cheaply and quickly on low-cost PCs and laptops than on mainframes.	***Increased training cost.*** Training costs are generally higher in a distributed model than they would be in a centralized model, sometimes even to the extent of offsetting operational and hardware savings.
User-friendly interface. Client devices are usually equipped with an easy-to-use graphical user interface (GUI). The GUI simplifies training and use for end users.	***Higher costs.*** Distributed databases require duplicated infrastructure to operate, such as physical location, environment, personnel, software, and licensing.
Less danger of a single-point failure. When one of the computers fails, the workload is picked up by other workstations. Data is also distributed at multiple sites.	
Processor independence. The end user can access any available copy of the data, and an end user's request is processed by any processor at the data location.	

Figure 12 : Advantages & Disadvantages of Distributed DBMS (Coronel & Morris 2019)

Problems Associated With Distributed Database Systems

One major problem associated with distributed database system is Deadlock. A Deadlock is simply a state where request made for a set of process request are being held by other processes and none of the processes can be completed. Requesting and acquiring of resources in any order without knowing the resource is acquired by other process. If the allocation of resources is not properly controlled, Deadlock may occur. In order to reduce the occurrence of deadlock in the distributed database system, one must focus more attention on the following techniques – Deadlock Detection, Deadlock, Avoidance Deadlock prevention and deadlock recovery e.t.c. .

• **Deadlock Detection**

In order to detect deadlocks in distributed database systems, a deadlock detection algorithm must be used. Where each site on the distributed system maintains a local wait for graph. If there is any cycle in the graph, it means deadlock exist in the system. Even though there is no cycle in the local wait for graph, there can be a deadlock due to the global acquisition of resources. In order to find the global deadlocks, global wait for graph is maintained. A centralized deadlock detection approach is used to detect the occurrence of deadlock in the system **(**Tomar & Megha 2014).

• **Deadlock Recovery**

Deadlock involves a cycle of alternating process and resource nodes in the resource graph. The general approach in recovering deadlock is process termination. In this method, nodes and edges of the resource graph are eliminated. A process termination algorithm is used to terminate all processes involved in the deadlock **(**Tomar & Megha 2014). This approach is said to be wasteful in most cases, because eliminating a single process is sufficient to break the deadlock so, it is better to terminate processes one at a time, release their resources, and check at each step if the deadlock still persists.

Before terminating a process, the 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

Conclusion

The importance of Distributed database system is so enormous and its usage have penetrated into most large organization and it has yield much profit in terms of data access, integrity, availability and speedy recovery of data processing. The huge data processed have enable businesses over the globe to improve.

The various issues with distributed database system cannot be compared to the progress and upliftment in our day to day data processing activities.

Factors affecting the implementation of a distributed database in the operating system are the different version and types of OS environment. Though distributed database systems have no problem and do execute well in some operating system, but it is still considered as challenges in DDBS due to the different specification of OS especially mobile environment such as android and iOS that make up the distributed database.

The deadlock handling can be controlled using deadlock detection agents. However, replication control and the integration of distributed databases in various operating systems are the challenges that still persist in recent years. The analysis suggests that these issues are continuously developing due to the rapid changes of technology so that there is no absolute solution for these challenges.

References

Abbas Q., Shafiq H., Ahmad I., & Tharanidharan I. (2016). Concurrency control in distributed database system. 2016 International Conference on Computer Communication and Informatics (ICCCI). Coimbatore, India: IEEE.

Coronel C. and Morris S. (2019). Database Systems Design, Implementation, and Management 13 Edition ISBN10: **1-337**-62790-9 https://www.cengagebrain.co.uk/shop/isbn/9781337627900

Garcia-Molina H., Ullman J. D., & Widom J. (2009). Database Systems: The Complete Book Second Edition Pearson Prentice Hall, Upper Saddle River, New Jersey

Goebel V. (2011). Distributed Database Systems Department of Informatics University of Oslo https://pdf4pro.com/amp/view/distributed-database-systems-forsiden-50afdb.html

Grech J. 2009 Designing and Implementing a Distributed Database for a Small Multi-Outlet Business https://epublications.regis.edu/cgi/viewcontent.cgi?ar

Jeffrey A. H., Mary B. P., & Fred R. M., (2005) Distributed Databases Modern Database Management 7th Edition file:///C:/Users/USER/Downloads/mdm7e_Ch13.pdf

Katembo K. E., Shri K. & Ruchi A., (2019). A systematic review on Distributed Databases Systems and their techniques: *Journal of Theoretical and Applied Information Technology*. 96(1) http://www.jatit.org/volumes/Vol97No1/21Vol97No1.pdf

Kaur M., & Kaur H., (2013) Concurrency Control in Distributed Database System. *International Journal of Advanced Research in Computer Science and Software Engineering Research* ISSN: 2277 128X Volume 3, Issue 7, https://d1wqtxts1xzle7.cloudfront.net/43755912/V3I7-0526.pdf

Mole P. (2020) Empirical Research on the Challenges of Distributed Databases: A Literature Review in Distributed and Parallel Databases. file:///C:/Users/USER/Downloads/EmpiricalResearchonthe Challenges ofDistributedDatabases.pdf

Ozsu M. T. & Valduriez P. (2019). Principles of Distributed Database Systems Third Edition Springer ISBN 978-1-4419-8833-1 e-ISBN 978-1-4419-8834-8 DOI 10.1007/978-1-4419-8834-8

Ozsu M. T. & Valduriez P. (2020). Principles of Distributed Database Systems Fourth Edition Springer https://arxiv.org/ftp/arxiv/papers/1111/1111.2852.pdf

Sagar B., & Swapnil S.,(2016). Fundamentals of Distributed Database System 1 ISSN: 2455-2631 IJSDR Volume 1, Issue 9 https://www.ijsdr.org/papers/IJSDR1609038.pdf

Tomar P., & Megha (2014). An Overview of Distributed Databases: *International Journal of Information and Computation Technology*. ISSN 0974-2239 Volume 4, Number 2 (pp. 207-214 © International Research Publications House https://www.ripublication.com/irph/ijictspl/ijictv4n2spl15.pdf

Tutorialpoint(2016). Distributed Database System: Simplyeasylearning.https://www.tutorialspoint.com/dbms/ index.htm