Distributed Database Problems, Approaches and Solutions — A Study

Md. Shohel Rana, Mohammad Khaled Sohel, and Md. Shohel Arman

***Abstract*—The distributed database system is the combination of two fully divergent approaches to data processing: database systems and computer network to deliver transparency of distributed and replicated data. The key determination of this paper is to achieve data integration and data distribution transparency, study and recognize the problems and approaches of the distributed database system. The distributed database is evolving technology to store and retrieve data from several location or sites with maintaining the dependability and obtainability of the data. In the paper we learn numerous problems in distributed database concurrency switch, design, transaction management problem etc. Distributed database allows to end worker to store and retrieve data anywhere in the network where database is located, during storing and accessing any data from distributed database through computer network faces numerous difficulties happens e.g. deadlock, concurrency and data allocation using fragmentation, clustering with multiple or single nodes, replication to overcome these difficulties it is essential to design the distributed database sensibly way.**

***Index Terms*—DDBMS, network, DDBMS rules, problems, approaches, PCDM, SCDM, DDA.**

I. Introduction

A logically interconnected group of collective data, materially distributed over a computer network or a system contains ofa group of sites linked together via communications network, in which each site is a database structure site where all the sites allow to work together, so that a user at any site can access data anywhere in the network where data stored at the user's own site that looks like one centralized database to the end user. Thus, instead of having one central database tolerate the entire load, it is shared by a collection of machines/computers over the communication networks or medium. A set of server machines working in synchronization to provide the desires to numerous users where these machines are connected to each other either through the wireless connection or through various communication media that serve data transfer at high rate forming a distributed database system. In the distributed system the processors used in those machines may differ from microcomputers to work position to minicomputers to computers used in day to day life [1].

According to Paulina Borsook [2], Distributed databases differ from traditional distributed processing where collaborative computing also is done by remote access. In distributed processing, users can work from remote locations, where the application, the database management program and parts of the data are located elsewhere. Micro-to-mainframe links and time-sharing are examples, but in terms of distributed database, the physical and logical location of the software and data are inappropriate to users conduct business without courtesy for the location of the database, knowing that they will have access to that data.

The chapter is organized in the following way. In the second chapter rules of the distributed database system are presented and in chapter three has been discussed on problems of DDBMS and four design approaches & distributed database architecture is explained. Chapter six is the conclusion.

Manuscript received July 15, 2018; revised September 6, 2018. The outcome of this study in Distributed storage system was implemented by a project work under School of Computing, The University of Southern Mississippi, United States.

Md. Shohel Rana is with School of Computing, The University of Southern Mississippi, Hattiesburg, MS 39406 USA (e-mail: md.rana@usm.edu).

Mohammad Khaled Sohel and Md. Shohel Arman are with Daffodil International University, Dhaka, Bangladesh (e-mail: khaledsohel@daffodilvarsity.edu.bd, arman.swe@diu.edu.bd).

II. Rules of Distributed Database System

A. Date's Rule

To make the assessment of distributed databases easier, Chris Date executive vice president of the Codd and Date Consulting Group, San Jose, Calif came up to formulate twelve directives plus a fundamental principle to describe DDBS. Though no current DDBS follows to all of them, they establish a convenient goal. The twelve rules are as follows [2]-[4]:

Rule 0 - The Fundamental Principal: To the user, a distributed database should look exactly like a non-distributed database.

Rule 1 - Local Autonomy: Each local site can play a role as an independent, autonomous, centralized DBMS with having the responsibility to these vital DMB functions (security, concurrency control, backup, and recovery).

Rule 2 - Central Site Autonomy: No site in the network relies on a central site or any other site. All sites should have the same competences, even though some sites may not unavoidably exercise all these competencies at a specified point in time.

Rule 3 - Failure Autonomy: The system is not affected by node failures. The system is in the nonstop act even in the case of a node failure or an enlargement of the network. In a comparable manner, the DDBMS should continue to work if new nodes are added.

Rule 4 - Location Autonomy (Transparency): The user does not need to know the location of data stored physically to

doi: 10.18178/ijmlc.2018.8.5.731

retrieve those data and be able to act as if all the data are stored locally.

Rule 5 - Fragmentation Autonomy: Data fragmentation is crystal clear to the user, who perceives only one logical database. The user does not need to know the name of the database fragments to retrieve them.

Rule 6 - Replication Autonomy: Relations and fragments can be represented at the physical level by multiple, distinct, stored copies or replicas at distinct sites, transparent to the user.

Rule 7 - Distributed Query Processing: A query should be capable of being executed at any node in the DDBMS that contains data relevant to the query. Many nodes may contribute in the response to the user's query without the user's being conscious of such contribution.

Rule 8 - Distributed Transaction Management: A transaction may update data at numerous sites, and the transaction is executed transparently.

Rule 9 - Hardware Independence: The distributed database system can run on any kind of hardware platform with all machines participating as equal partners where appropriate.

Rule 10 - Operating System Independence: The distributed database system can run on any kind of operating system (e.g. Windows, Linux, MacOS, Android).

Rule 11 - Network Independence: The DDB and its associated DDBMS should be capable of being implemented on any suitable network platform.

Rule 12 - Database Independence: The system must support any vendor's database product. That means the DDBS should be able to work with different database if they have the same interfaces.

B. Stonebraker's Rule

Michael Stonebraker, professor of computer science at the University of California, Berkeley and chief theoretician at Relational Technologies Inc. proposed a set of six rules that create a model for distributed databases. This set of rules described below [2], [5].

Rule 1 - Retrieval Transparency: Transactions should be able to be retrieved from any site, from which the transaction was submitted. The equal consequences should be produced, regardless of the site submitting the transaction.

Rule 2 - Update Transparency: Transactions should be able to be updated form any site, regardless of the site from which the transaction was submitted. The equal consequences should be produced, regardless of the site submitting the transaction.

Rule 3 - Scheme Transparency: Any authorized user can issue, from any site, data-definition changes so that those changes become observable everywhere.

Rule 4 - Performance Transparency: The performance of any command at one site should be comparable to the performance of that facility distributed from any further site.

Rule 5 - Copy Transparency: Distributed databases should support redundant copies of database objects.

Rule 6 - Tool Transparency: All tools provided by a vendor must support location transparency so that users need not concern which tools can be used to manipulate remote data.

By summarizing the two rules we can easily observe that both emphasize the following goals:

Distributed database can be alike a non-distributed database (Date Rules: 0, Stonebraker Rules: 6)

Independence of individual sites within the system from other sites and non-dependence of the system on any one site (Date Rules: 1-3, Stonebraker Rules: 1-2)

Transparency, to users, of the operations of the system and the distribution of the data (Date Rules: 4-6, Stonebraker Rules: 3-5)

Distributed nature of query and transaction processing (Date Rules: 7-8, Stonebraker Rules: 1-5)

Independence of the system with respect to hardware, operating systems, network software, and database management systems (Date Rules: 9-12, Stonebraker Rules: 7).

III. Problems in Distributed Database System

A. The Ideal Situation

The distribution can be geographical or local in which every single application should be able to work transparently on data that is:

- spread across a variety of different DBMS's
- running on a variety of different machines
- supported by a variety of different operating systems
- connected by a variety of different communication networks.

B. Problems Areas of DDBMS

The problems of Distributed database [6]-[8] can be described as follows:

Distributed Concurrency Control: The integrity of the database is maintained by specifying the synchronization of access to the distributed database to manage concurrency different locking techniques uses based on the mutual exclusion of access to data.

Replication Control: Replication techniques only applicable to distributed systems where a database is supposed to be replicated if the whole database or a percentage of it is copied and the copies are stockpiled at dissimilar sites. Having more than one copy of a database, the issue is continuing the communal uniformity of the copies ensuring with all copies are identical schema and data.

Deadlock Handling: Numerous users request for the same resources from the database if the resources are obtainable at that time, then database allow permission for the resources to that user if not obtainable the user has to wait until the resources are released by another user. Sometimes the users do not release the resources blocked by some other users.

OS Environment: To implement distributed database environment a specific operating system is required as per organizational requirements. The operating system plays an important role to manage the distributed database because sometimes it doesn't support for distributed database.

Transparent Management: The major problem area in the distributed database where data located in numerous locations and number of users are used that database. So, the transparent management of data is important to maintain the integrity of distributed database.

Security and Privacy: A great issue in the distributed

system that how to apply the security policies to the interdependent system. Since distributed systems contract with sensitive data and information ensuring with the strong security and privacy measurement system exists. The important issues in the distributed system are the protection of distributed system assets together with the fundamental resources, storage, communications and user-interface I/O, higher-level compounds of these resources, like processes, files, messages, display windows and more complex objects, etc.

Resource Management: Resources are in different places in the distributed system. Routing is an issue at the network layer of the distributed system and at the application layer. To cooperate with these resource in a distributed system.

IV. Design Approaches and Distributed Database Architecture

This section describes the approaches for designing distributed database to meet the following goals [9]:

- to provide high performance
- to provide reliability
- to provide functionality
- to fit into the existing environment
- to provide cost-saving solutions

Importance of design Reasons of poor efficiency: Hardware: 10%, DBMS: 15%, Database design: 25%, Applications design: 55% and Costs of improvement: Hardware: 10%, DBMS: 20%, Database design: 40%, Applications design: 30% [9].

A. General Design Steps

- Group of logically-related communal data.
- Data split into fragments.
- Fragments may be replicated.
- Fragments/replicas assigned to sites.
- Sites connected by a communications network.
- Data at each site is under control of a DBMS.
- DBMSs grip local applications autonomously.

B. Top Down Approach

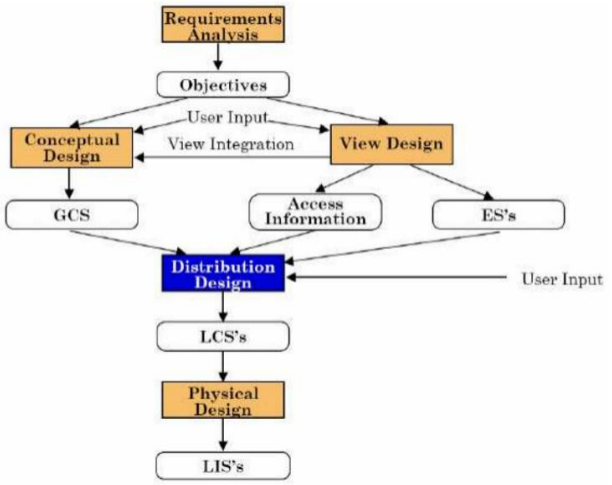

Fig. 1. Top-down approach.

This approach shown in the Fig. 1 is typically used during the implementation of the distributed database from a foundation. Starting from a requirement analysis phase with the analysis of the company situation, defining problems and restraints, defining purposes, and designing scope and margins. The subsequent activities are conceptual design and view design. The conceptual design deals with entity relationship modeling and normalization focusing on the data requirements [9]-[11].

C. Bottom Up Approach

This approach shown in Fig. 2 is used only when the distributed database already exists, and we just add another database to an existing setting. In this situation, we can improve further features into the existing database [9]-[11].

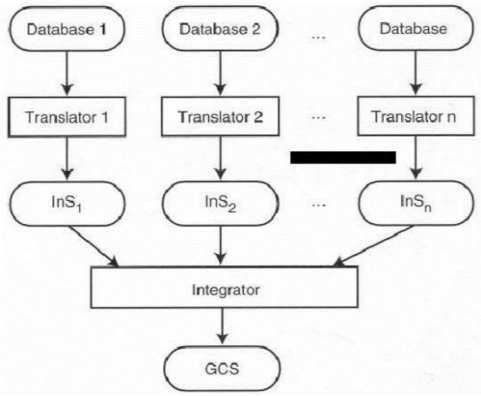

Fig. 2. Bottom-up approach.

The top-down technique is more accepted if the design is constructed from a scratch or the bottom-up technique is typically used if the design matches to the existing systems or some modules are yet ready.

V. Solutions in DDBMS

According to Michael Stonebraker [12] Distributed Database management systems offer advanced level user support than conventional operating systems where numerous operating system services examined with an opinion toward their applicability to support of database management roles including buffer pool management, file system, scheduling, interposes management and communication with consistency control. Task switching overhead solution for an operating system to create a special scheduling class for the DDBMS and other users by proposing model called 'Server Model' became feasible if the operating system provided a message facility which allowed n processes to originate messages to a single destination process by its own scheduling and multitasking.

C. Sunil Kumar [13] addressed security features when escalating a distributed database. Several factors had been considered during the choice between the object-oriented and the relational data model where the most significant of these issues were single and multilevel access controls (MAC), protection and integrity maintenance. The choice didn't make solely based on existing security features while defining which distributed database replica be more secure for a specific job. One did also request the effectiveness and efficiency of the delivery of these features. In this paper, they

provided a solution for in Relational Database Protection by proposing access control called SQL View, because Users can read or modify data in their view, but the view prohibits users from accessing data at a classification level above their own. A user at a lower classification level will be unaware that data exists at a higher classification level if the view is properly designed. Another is Global Views which is fruitful at data control to a lesser extent at implication protection because their usage can be computationally costly, and the addition of global views adds computational time to a process too long.

D. Cohen [14] addressed in this paper, in the setting of transaction processing including multicopy updates, concurrency control, and crash recovery. A variety of the primary node idea for multicopy updates was implemented. Data inconsistencies, formed by early termination of transaction processing, are detected and removed. In this paper, PCDM accepts three types of transactions. (i) Customer subnetwork transactions issued by an administrator to exercise control over the subnetwork. (ii) Process control transactions issued by maintenance staffs to monitor and control PCDM functionality. (iii) Response transactions issued by SDCMS in response to PCDM initiated update transactions. Having a uniform high priority process control and response transactions processed before all new subnetworks management transactions. Customer subnetwork transactions are processed by three priorities specifying by the administrators of each transaction submitted. If not specified, a default priority is assigned by the system uniformly for all customers. The SCDM accepts three types of transactions. (i) Update transactions issued by PCDMS to create, remove, or modify profiles in the SCDB. (ii) Process control transactions issued by maintenance personnel to monitor and control SCDM functionally. (iii) Enforcement transactions issued by other processes to verify the capabilities of a user in the context of a service request. Having a uniform high priority, Process control and enforcement transactions are processed before all new update transactions. All update transactions are handled with a single-priority issued by the PCDM. The update algorithm grips transactions to add, modify and remove the profile. Each update transaction is routed to the PCDM at the account's home service area.

The following components used for the Recovery mechanism. (i) Duplex hardware provided to protect service against single hardware failures. (ii) The operating system maintains two copies of each data element on distinct disk drives. (iii) The node creates to take backup copy of the database for checkpointing. (iv) During transaction processing, the system maintains on disk the lock table and a completed update transaction log. The completed update transaction log maintained at the physical page level and used only by the recovery process. (v) Node maintained the synchronization of transactions for restoring database consistency at all levels. Using these recovery components specified service restored in the following way:

PCDM's responsibility to

- Check file consistency on disk.
- If both disk copies are lost, install backup copy, and internally run the completed update transaction log.
- Use the lock table to back out rashly terminated transactions.
- Renew subnetwork management service.
- Detect variations during usual transaction processing to notify node.

SCDM's responsibility to

- Check file consistency on disk.
- If both copies of the database on disk are lost, install backup, and internally run the completed update transaction log.
- Renew communication service.
- Inconsistencies detected during normal transaction processing are promoted to a node.

Natalija K. [15] proposed a new deadlock detection algorithm for the distributed database with dynamically creating deadlock detection agents (DDAs) shown in figure 3 that is responsible for detecting deadlocks in one connected component of the global wait-for-graph (WFG). Distribution of information about the global WFG between different agents, called DDAs for every cycle in the graph is the main idea of the algorithm where one DDA have the complete information about it. Each cycle goes to one linked section of the universal WFG for detecting cycles by allowing transactions to be associated with at most one DDA. Since only one DDA detects deadlocks one transaction is involved and the same deadlock will not be detected twice, which is a characteristic problem another algorithm for distributed deadlock detection encounter. The proposed algorithm dynamically creates DDAs when they are needed. A DDA starts detecting a deadlock each time it receives new dependencies send to a DDA contain one transaction waiting for a set of other transactions. Consequently, the waiting transaction being a member of all cycles currently executing as part of the WFG the DDA is responsible for. The deadlock detection follows depth-first search, starting from the waiting transaction. If a cycle is found, the DDA resolves it. At present, this is done by choosing the youngest transaction from the cycle to be terminated. If more than one cycle found through the new dependencies, the transaction whose wait dependencies newly arrived is terminated, it seems all cycles get resolved.

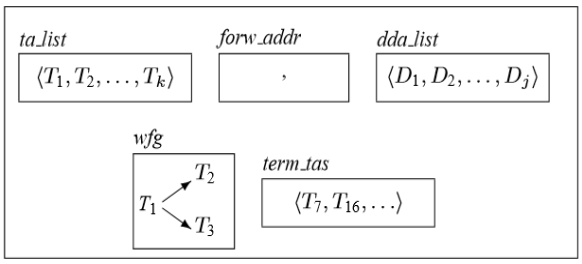

Fig. 3. DDA structure.

In distributed databases replication control procedure used to store numerous copies of data at different locations. The problem with having numerous copies in multiple locations is the overhead to maintain data reliability, predominantly during update actions. To maintain mutually reliable data in all sites, replication control procedures need to be implemented. Some of the replication control algorithms including (i) Master-slave replication control consists of is one master site and 'N' slave sites where a master algorithm

runs at the master site to detect conflicts and a copy of slave algorithm runs at each slave site with two phases: transaction acceptance/rejection phase and transaction application phase, (ii) Distributed voting algorithm includes of 'N' peer sites, before starting execution a transaction all sites have to ensure "OK", (iii) Majority consensus where a transaction is permitted to execute when a majority of the peers "OK", (iv) Circulating token where the transactions in the system are serialized using a circulating token and executed accordingly against every replica of the database [16], [17].

Among numerous algorithm Yuan Wei [18] proposed dynamic replication control algorithm called ORDER algorithm designed for gaining competence over replication strategies by dynamically changing the update frequency and update duration of replicas. When receiving a transaction, this algorithm assesses the data needs of the incoming transaction and creates data replicas for the transaction. It also registers the update frequency and duration to the primary sites of these replicas. This algorithm job is to register active replicas to their primary sites from the receiving site also to push updates to the active replicas at the extended frequencies requested by the incoming transactions. When a local site acknowledges a transaction, the algorithm computes the appropriate update frequency and update duration for each remote data item quantified by the transaction. The algorithm keeps track of all transactions using the data replicas until they expire to maintain the minimum update frequency for all active replicas. The algorithm also requires to re-calculate the update frequency of replicas accessed by an expiring transaction.

VI. Conclusion

The distributed database is one of evolving technology in the research field and business organization. In this paper, we learn rules of distributed database proposed by Date and Stonebraker. After merging these two rules we figure out the standard that follow can help in any distributed database system to build a new model for distributed storage environment. In this paper, we also discuss various problem areas, approaches and numerous solutions of the distributed database. The problem areas declared in the paper are very useful while implementing distributed database so that concurrency, deadlock, replication control, security, and privacy is easily managed. In this paper, we also learn the architecture of distributed database for implementing distributed database system.

References

[1] D. S. Hiremath and S. B. Kishor, "Distributed database problem areas and approaches," *IOSR Journal of Computer Engineering (IOSR-JCE)*, pp. 15-18, 2016.

[2] P. Borsook, "New pains, new gains: Distributed database solutions are on their way," *Data Communication*, March 1988.

[3] C. Date, "Twelve rules for a distributed database," *Computer World*, June 1987.

[4] C. J. Date, *An Introduction to Database Systems*, 5th ed., Reading, MA: Addison-Wesley, vol. 1, 1990, ch. 23.

[5] M. Stonebraker, "The integration of rule systems and database systems," *TKDE*, vol. 4, no. 5, pp. 415-423, 1992.

[6] A. S Tanenbaum, "Distributed systems principles and paradigms," *PHI*.

[7] S. Ceri and G. Pelagatti, *Distributed Databases: Principles and Systems*, McGraw-Hill Computer Science Series, 1984.

[8] A. B. Gadicha, A. S. Alvi, Vijay B. Gadicha, and S. M. Zaki, "Top-down approach process built on conceptual design to physical design using LIS, GCS schema," *International Journal of Engineering Sciences & Emerging Technologies*, vol. 3, issue 1, pp. 90-96, August 2002.

[9] Design of Distributed Databases. (December 12, 2017). [Online]. Available: http://mazsola.iit.unimiskolc.hu/tempus/discom/doc/db/tema12.pdf

[10] L. Braz de Oliveira Macaferi, *Top-Down Approach in Distributed Databases*, Barra Mansa, November 2007.

[11] S. K. Rahimi and F. S. Haug, *A Distributed Database Management System a Practical Approach*, Wiley Publication.

[12] M. Stonebraker, *Operating System Support for Database Management*, University of California, Berkeley.

[13] C. S. Kumar, J. Seetha, and S. R. Vinotha, "Security implications of distributed database management system models," *International Journal of Soft Computing and Software Engineering*, vol. 2, no. 11, 2012.

[14] D. Cohen, "Implementation of a distributed database management system to support logical subnetworks," *The Bell System Technical Journal*, vol. 61, no. 9, November 1982.

[15] N. Krivokapic, A. Kemper, and E. Gudes, "Deadlock detection in distributed database systems: A new algorithm and a comparative performance analysis," *The VLDB Journal*, vol. 8, pp. 79-100, 1999.

[16] Distributed DBMS - Replication Control. (December 12, 2017). [Online]. Available: https://www.tutorialspoint.com/distributed_dbms/distributed_dbms_replication_control.htm

[17] B. Gavish and M. W. Suh, "Configuration of fully replicated distributed database system over wide area networks," *Annals of Operations Research*, vol. 36, pp. 167-192, 1992.

[18] Y. Wei, A. A. Aslinger, S. H. Son, and J. A. Stankovic, "ORDER: A dynamic replication algorithm for periodic transactions in distributed real-time databases," Department of Computer Science, University of Virginia, USA.

Md. Shohel Rana was born in Sirajganj, Rajshahi, Bangladesh, in 1987. He received his B.Sc. (Engg.) and M.Sc. (Engg.) both in computer science and engineering from the Mawlana Bhashani Science and Technology University, Tangail, Bangladesh in 2010 and 2014 respectively. Currently he is pursuing his Ph.D. in computational science under School of Computing at The University of Southern Mississippi, Mississippi, United States. His research area includes digital image processing and computer vision, data science, machine learning, human computer interaction, e-learning/remote learning, distributed database system and web technology.

Mohammad Khaled Sohel was born in Mymensingh, Bangladesh in 1972. He received his M.S in management information system from Daffodil International University, Dhaka, Bangladesh in 2007. He pursued his B.Sc. (Hons.) in computing and information systems from London Metropolitan University, United Kingdom. Currently he is working as an assistant professor in the Department of Software Engineering under the Faculty of Science and Technology at The Daffodil International University, Dhaka, Bangladesh. His research AREA includes RFID technology, computer networks, distributed database system, blockchain technology, information systems management.

Md. Shohel Arman is a lecturer and alumni of Department of Software Engineering under Faculty of Science & Information Technology in Daffodil International University, Dhaka, Bangladesh. He is an energetic and focused man since his student life. Currently he is focusing on his interested research area. His research interests are distributed database system, machine learning, data mining, internet of things (IOT), software security and management information system (MIS).