SEEE DIGIBOOK ON ENGINEERING & TECHNOLOGY, VOL. 01 (2), JUN 2020

ENGINEERING AND TECHNOLOGY

# Data Preservation and Concurrency Management in Distributed Database

**Ananthi P, Madan A Sendhil, Kanimozhi R, Maheswari R, Rathika C**

**Rathinam Technical Campus, Coimbatore, India**

**Abstract**

A decentralized database has the ability to be hosted anywhere, including the Internet, the intranets or extranets of a company, or even on workstations that are dispersed around the company itself. There is also the potential that information is stored on servers that are a part of a network that is well organized. This is another option available to you. Because the information that distributed databases keep is dispersed across several computers, they have the potential to increase productivity at remote work sites. This is because distributed databases store data in a manner that is decentralized. This is because the data is stored in distributed databases in a decentralized form, which is the reason for this result. As opposed to being the only reason, the justification for this is that data storage makes it possible for transactions to be carried out on multiple workstations rather than just one. User data privacy has become a major worry over the past few years as a direct result of the proliferation of distributed databases, which are the technology that is currently considered to be the most cutting edge. This is a direct consequence of the widespread implementation of distributed database systems. On a regular basis, comprehensive inspections as well as routine maintenance must be performed on the system's access control and general integrity. This is an imperative necessity. In the following paragraphs, we will discuss a number of solutions to the problem of data loss, all of which are based on the characteristics of distributed database systems. Each of these solutions has its own advantages and disadvantages. Multilayer access control, concurrent access, reliability, integrity, and recovery are some of the qualities that fall under this category. The fundamental timestamp method, the distributed two phase locking protocol (2PL), the distributed pessimistic protocol, and the Deadlock algorithm will be the primary foci of this research into concurrency control-based algorithms.

Keywords: decentralised database, privacy legislation, multi-level access control, and concurrent access.

## Introduction

Distributed database management systems, also known as DDBMS, are a subset of database management systems (DBMS) that are responsible for maintaining a large number of databases that are housed in different physical locations but are linked together through the use of a computer network. [3] It offers features that enable users to view a distributed database as a single entity regardless of its location. This pattern has been influenced by globalization, which has increased the propensity of firms to expand their activities around the globe. As a result, this pattern has become more widespread. They might come to the conclusion that instead of keeping a centralized database, they would be better served by dispersing data among a number of separate dedicated servers in order to better meet their needs. As a direct consequence of this, the idea of using distributed databases came to be developed throughout time. In addition, every location is able to communicate with the others when it becomes necessary to do so. Through the utilization of a connection that is known as a database link, users on the local machine can have access to data that is kept on a database that is hosted on a remote machine. Each database that is a component of the distributed system needs to have a distinct global database name within the network domain so that these connections may be made. The objective of doing so is to facilitate the establishment of these connections. When working with a distributed system, it is possible to identify a database server in a way that gives the impression of being unique by making use of the name of the global database. A dispersed database management system, more commonly referred to as a DDBMS, is a sort of technology that enables users to operate dispersed databases while concealing the difficulties associated with the dispersal of data. This type of technology is generally referred to as the DDBMS. The fact that distributed databases are typically physically separated from one another and are administered independently between local and global transactions is the primary distinction that can be made between a centralized database and a distributed database. This is the most important distinction that can be made between the two different kinds of databases. If an event does not retrieve any data from any sites other than where it originated, then that event is regarded to be local. On the other hand, a transaction is regarded as global if it either attempts to analyze it from a location different from the one where it was initially created or simultaneously accesses data from a number of other locations.

In this study, we investigate the security concern of distributed databases, and we will also explore the security difficulties that were observed in both models. In addition, we will analyze the specific challenges posed by each system's security measures towards the end; the comparison will focus on the relative advantages offered by each model with regard to safety.

Concept of an Artificial Intelligence Distributed Database System

The idea of a database that was shared across multiple locations did not originate until the middle of the 1970s. Because it was expected that a large number of applications will be disseminated in the not too distant future, it became necessary to likewise distribute the database. A distributed database system, also known as a DDBS, is a collection of multiple databases that are conceptually interconnected but physically spread among numerous computers or locations via a computer network. These databases make up a distributed database system. Users of a distributed database have the misconception that the entirety of the database is locally accessible, with the possible exception of communication delays that may occur between locations. Users are oblivious to the fact that the database is distributed due to the fact that a distributed database is a logical union of all locations. A non-distributed or centralized database system should be avoided in favor of a distributed database system, often known as a DDBS. There are many good reasons for this. Work is frequently distributed across multiple locations within an organization.

One of the most important aspects of the information system is the planning and construction of a trustworthy distributed database platform. The most significant contributors to the total amount of time required to process a query or update in a network with a high bandwidth are the network's latency and the amount of processing that occurs locally. Processing in parallel is one way that can be utilized to lessen the impact of these problems, particularly if this tactic is taken into consideration from the very beginning of the design phase. The strategic use of replication paves the way for the implementation of parallelism in a way that is both effective and practical. Therefore, the design of a distributed database can be thought of as an optimization issue that requires solutions to a variety of interrelated challenges, such as the fragmentation of data, the allocation of data, and the optimization of the local environment. Controlling concurrency is an extra problem that database systems could face at any given time (CC). It enables users to access a distributed database in a multiprogrammed manner while maintaining the appearance that each user operates independently on a dedicated system. This is accomplished by maintaining the appearance of a dedicated system for each user. Due to the fact that the technology can be programmed in a variety of languages, this is feasible. An additional element of concurrency control (CC) is the coordination of concurrent database visits inside a multi-user database management system. This feature, which is sometimes referred to as "concurrent access coordination" in some contexts, can be found in multi-user database management systems (DDBMS). There are several different approaches to concurrency control that can be taken. These algorithms are illustrated by the following examples: two-phase locking, time stamping, multi-version timestamps, and optimistic non-locking methods. There are many various approaches to concurrency control, and some of them are better than others depending on the system.

Controlling concurrency is an extra problem that database systems could face at any given time (CC). It enables users to access a distributed database in a multiprogrammed manner while maintaining the appearance that each user operates independently on a dedicated system. This is accomplished by maintaining the appearance of a dedicated system for each user. Due to the fact that the technology can be programmed in a variety of languages, this is feasible. Concurrency control, often known as CC, is used in multi-user database management systems to help coordinate concurrent database accesses. This is one of the roles that CC plays. "concurrent access coordination" is what "CC" stands for as an acronym (DDBMS). There are several different approaches to concurrency control that can be taken. These algorithms are illustrated by the following examples: two-phase locking, time stamping, multi-version timestamps, and optimistic non-locking methods. Certain solutions for concurrency control are preferable to others, although this is very dependent on the system [5].

## III. DESIGN STRATEGIES OF DISTRIBUTED DATABASE

The idea of a database that was spread out across multiple locations was not conceived of for the first time until the late 1970s. It was imperative that the database be sent out as well given that it was anticipated that a considerable number of applications would be sent out in the near future. A distributed database system, also known as a DDBS, is a collection of multiple databases that are conceptually interconnected but physically spread among numerous computers or locations via a computer network. These databases make up a distributed database system. The term "distributed database system" refers to the collection of databases that make up this system. Users of a distributed database have the misconception that the database in its entirety is locally accessible, with the possible exception of any communication delays that may occur between locations. This is a common misunderstanding among people. Because a distributed database is a logical union of all sites, users are not aware that the database is spread out across multiple locations. As a direct consequence of this, it will be possible to access the database in a more timely manner. A non-distributed or centralized database system is inferior to its comparable distributed database system (DDBS), which is preferable for a variety of reasons. A distributed database system (DDBS) is superior to its non-distributed or centralized counterpart. Across an organization, work is often decentralized and carried out in a variety of locations.

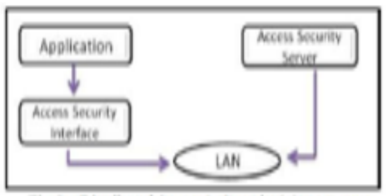

Fig 5 : Distributed System's Security Management

One of the most significant challenges that the information system must overcome is the development of a trustworthy distributed database system. Within a network that has a high bandwidth, the two variables that have the most impact on the entirety of the processing of a query or update are the network's latency and the local processor. One technique that may be utilized to lessen the impact of these aspects is known as parallel processing, and this strategy is especially effective when it is taken into consideration throughout the entirety of the process of designing the system. It is not possible to take advantage of concurrency in a productive manner without first employing replication in a planned and strategic fashion. As a consequence of this, the planning of a distributed database can be interpreted as an optimization problem that requires solutions to a range of problems that are reliant upon one another. A few of these problems are the fragmentation of data, the allocation of data, and the optimization of the local environment. In relational database management systems, concurrency control is an additional problem that could potentially occur (CC). Users are given access to a multi-programmed version of a distributed database, but it is made to appear as though each user is performing their tasks independently on a system that is solely dedicated to them. This is because the computer is capable of operating multiple programs all at once, which explains why this is the case. An additional function of concurrency management, known as "concurrent access coordination," is the coordination of concurrent accesses to a database that are regulated by multi-user database systems. "concurrent access coordination" is really the name of this function (DDBMS). Concurrency control can be accomplished using a great number of different strategies. Two-phase locking, time stamping, multi-version timestamps, and optimistic non-locking procedures are just a few examples of the several methods that fall under this category. When applied to certain types of systems, specific concurrency control strategies perform significantly better than others.

In relational database management systems, concurrency control is an additional problem that could potentially occur (CC). Users are given access to a multi-programmed version of a distributed database, but it is made to appear as though each user is performing their tasks independently on a system that is solely dedicated to them. This is because the system is capable of simultaneously operating a large number of programs, which is the reason behind this. The term "concurrent access coordination," which is shortened to "concurrency control" (CC), refers to the process of coordinating numerous users' simultaneous database accesses in a multi-user database management system. Concurrency control serves a variety of applications (DDBMS). Concurrency control can be accomplished using a great number of different strategies. Two-phase locking, time stamping, multi-version timestamps, and optimistic non-locking procedures are just a few examples of the several methods that fall under this category. Although some methods offer superior concurrency management to others [5,] the level of control provided by a given system can vary considerably.

## IV. DATA SECURITY IN DISTRIBUTED DATABASE

It was not until the 1970s when both the concept of a distributed database as well as its implementation were first conceived of by someone. Because it was anticipated that a significant number of service applications would be submitted in the not-too-distant future, it was imperative that both the database and the applications be disseminated. A distributed database system, also known as a DDBS, is a collection of multiple databases that are conceptually interconnected but physically spread among numerous computers or locations via a computer network. These databases make up a distributed database system. The term "distributed database system" refers to the collection of databases that make up this system. A distributed database system is what this collection of databases ultimately amounts to from a purely technical standpoint. Users of a distributed database have the misconception that the database in its entirety is locally accessible, with the possible exception of any communication delays that may occur between locations. This is a mistake that a lot of people make all the time. Because a distributed database would be a logical union of all locations, users are unaware of something that may be referred to as the database's distribution. As a direct consequence of this, it will be possible to access the database in a more timely manner. As a direct consequence of this change, it will be possible to access the database in a more timely manner. A non-distributed or centralized database system is not as good as its equivalent, a distributed database system (DDBS), which is desirable for a number of different reasons. The performance of a distributed database is known to be superior, which is one of these criteria. The work that needs to get done is typically divided up and distributed to a number of different areas within an organization.

One of the most serious challenges that the information system must overcome is the creation of a database management system (dbms) that is able to be held accountable. Latency and local computing are the two characteristics that have the biggest impact on the overall length of work necessary to execute a query or an update on a network that has a high bandwidth. This is because latency is the amount of time it takes for a request to be processed after it is sent. Parallelism is one strategy that may be implemented to lessen the impact of these problems, and it is especially effective in this capacity if parallel processing is taken into account from the very beginning of the system design process. When replication is used in a deliberate and organized manner, only then is it feasible to make effective use of parallelism. This is especially true for the processing of enormous amounts of data. As a consequence of this, the creation of a distributed database can be seen as an optimization task that calls for the solution of a number of different difficulties that are dependant on one another. Data fragmentation, data allocation, and local optimization are only some of the problems that arise as a result of this. Another problem that could appear in relational database systems is referred to by its acronym, concurrency control (CC). Users are provided with the capability to access a distributed database in a multi-programmed manner, while at the same time the illusion is preserved that each user is working independently on a dedicated server. This is due to the fact that the system is capable of executing many programs at the same time, which makes it possible for this to take place. One further use of concurrency control, also known as concurrent access coordination, is the process of coordinating the simultaneous accesses that several users have to a database that is maintained by a multi-user database system. This is one of the ways in which concurrency control can be applied (DDBMS). There is a great variety of different approaches one can take in order to acquire the skill of being able to manage concurrency. Approaches such as time stamping, optimistic non-locking methods, multi-version headers, and two-phase locking are all examples of the procedures that fall under this group of processes. When applied to certain systems, certain methods give more concurrency control than do others. Compare and contrast with other solutions.

The term "concurrency control" is an abbreviation for yet another challenge that might be encountered by relational database systems (CC). Users are provided with the capability to access a distributed database in a multi-programmed manner, while at the same time the illusion is preserved that each user is working independently on a dedicated server. This is due to the fact that the system is capable of executing numerous processes at the same time, which makes it possible for this to happen. Concurrency control (CC), which is an acronym for "concurrent access coordination," is a component of a multi-user database management system that is in charge of coordinating the ongoing database accesses of several users (DDBMS). There is a great variety of different approaches one can take in order to acquire the skill of being able to manage concurrency. The two phase locking, time sequencing, multi-version timestamps, and optimistic non-locking approaches are all examples of the algorithms that fall under this category of algorithms. Even though the answer to this question will change depending on the system, there are certain concurrency control solutions that are superior to others [5].

Aspects of client and server architecture pertaining to data security:

There are five different approaches to consider when talking about the safety of the customer model.

It's possible that the user workstation endorsement mechanism is only partially functional or doesn't even exist.

- The sign-in process has the potential to be made more automated, which is already possible.
- The location of the workstation could be in a potentially hazardous or public area, depending on the circumstances.
- It is possible that the workstation is attempting to circumvent the security protection by turning on highly effective utility services or development equipment.
- In severe situations, users have the ability to assume the identity of another user in order to get access to the system.

## V. DISTRIBUTED CONCURRENCY CONTROL

Concurrency control is the management of multiple users accessing the same database at the same time in a Distributed Database System (DBMS). Strong two-phase locking is the method that is used the most frequently for controlling concurrency in distributed systems. In database management systems and transaction processing, the term "distributed concurrency control" most often refers to the process of controlling the concurrency of a distributed database. The basic goal of distributed concurrency control is to ensure that multi-database systems are capable of distributed serializability. The process of managing several users' simultaneous database access within a multi-user management system is referred to as the concurrency problem (DBMS). Concurrency control is an additional challenge that comes along with database systems. It gives users the ability to access a distributed database in multiprogramming passion while keeping up the appearance that each user is operating independently on a dedicated system [10].

Serializability is achieved by the use of concurrency control, which modifies the behavior of concurrently running transactions so that they are unable to leave the database in an inconsistent state. In a shared database system, serializability is one method that can be used to assure consistency. Serializability requires that a set of concurrent transactions be carried out in a manner that produces the same results as serial execution. This must be accomplished in order to achieve serializability. [9] The purpose of this section is to study the influence that tiered security serializability has within a single-site database system.

The Time Stamp Ordering Algorithm in Its Most Primitive Form

A time stamp can be used to determine whether or not a request is out of date in relation to the data object it is working on, and it can also be used to rank occurrences in relation to one another. One of the mechanisms that is utilized in distributed databases to ensure that there is continuity [16] is called timestamp ordering.

A timestamp is a unique identifier that can be used to determine the details of a transaction. In this method, the transactions are arranged according to the timestamp values associated with each one. Despite the fact that transactions might sometimes clash with one another, the timestamp-ordering protocol ensures that they can always be processed in the correct sequence. tasks related to reading and writing

It is the responsibility of the protocol system to carry out the conflicting pair of tasks depending on the timestamp values of the tasks' respective timestamps.

Transaction T has a Request for Action (A) operation.

If WTS is greater than TS (T), then roll back T; else, successfully execute R (A).

and then set RTS (A) to equal the maximum of RTS (A) and TS (T).

T) The operation T issues the W (A) transaction.

If the RTS for A is higher than the TS for T, then rollback T.

If WTS (A) is greater than TS (T), then rollback T; otherwise, do nothing.

Carry out W (A) with success, and ensure that WTS (A) = TS (T)

Locking that is Distributed in Two Phases (2PL)

There have been many different approaches to concurrency control developed throughout the years in order to guarantee the serializability of transactions that are carried out in parallel. The locking procedure is taken into consideration to be one of these methods. There are a variety of approaches that can be taken to lock something. One of the key strategies that is utilized in order to manage concurrency in distributed database systems is referred to as the two-phase locking protocol. The maxim "Read any, write all" serves as the guiding philosophy behind this protocol.

When things are read by a transaction, a read lock is placed on those items. If the read lock later has to be modified, the transaction changes the read lock into a write lock. It is not necessary to lock the local copy of an item to read it; all that is required is to place a read lock on any duplicate of the item. However, in order to update an item, all copies of the item must have the "write locks" setting enabled. While the transaction is being carried out, write locks are obtained, and the transaction will not proceed with a write request until all of the to-be-modified copies of the object have been safely locked. This occurs at some point during the process of the transaction. Every lock is kept in place until either the transaction is finished in a successful manner or it is abandoned and deleted. The 2PL Protocol acts as a lock gatekeeper because it specifies the criteria under which transactions can acquire and release locks. This allows the 2PL Protocol to control who can acquire and release locks. In order to comply with the requirements of the 2PL protocol, each transaction must, in two separate steps, either trigger a lock or unlock request.

• A transaction may acquire locks while in the Growing Phase, but it will not be able to release those locks.

• A transaction can unlock doors during the Shrinking Phase, but it is unable to purchase new locks at this time.

At first, the transaction will enter the Growing Phase, which is the time period during which any necessary locks will be requested. After this, it will enter the Shrinking Phase, during which it will give up all locks and stop being able to send requests. During this time, it will also stop being able to read requests. Transactions utilizing the 2PL Protocol must first get all necessary locks before moving on to the unlock phase. The 2PL protocol guarantees that serialization will take place, but it does not check for or avoid deadlocks at any point throughout the process. As a direct consequence of this, this strategy may result in a deadlock. Each time a transaction becomes stuck, local deadlocks are investigated, and if necessary, they are broken by resuming the transaction with the most recent initial start-up time of any of the parties engaged in the deadlock cycle. As a result, there is the possibility of reaching an impasse. The "Snoop" process is in charge of determining whether or not there is a global impasse. It accomplishes this by routinely requesting waits-for information from all sites, and then searching for and resolving any potential global deadlocks that may have arisen as a result of this information.

**Optimistic Distributed Protocol Distributed (OPT)**

The third algorithm is an optimistic distributed concurrency control algorithm that is based on timestamps and exchanges certification information while the commit protocol is being carried out. The timestamps of read and written operations are maintained for each individual data item. Optimistic concurrency control is capable of delivering serializability even when working within the limits that are imposed by our two fundamental assumptions. If concurrency control is not put into place, concurrent transactions have the potential to produce execution orders that cannot be sequentialized [9].

Transactions are able to freely read and edit data objects, with any changes being saved in a local workspace up until the commit time. The transaction needs to be able to remember the version identifier (also known as the write timestamp) of the item at the precise moment that it performs each read. After all of the members of the transaction's cohorts have finished their work and returned their findings to the master, the transaction is given a timestamp that is completely unique across the entire world. This time stamp is provided to each cohort in the message that states it is "ready to commit," and it is utilized to locally validate all of the reads and writes performed by each cohort, as will be detailed below:

In order for a read request to be considered valid, the version of the item that is being requested to be read must continue to be the most up-to-date version.

There are no writes with a later date that have been authenticated in the immediate area. A write request is considered to be valid if there have not been any subsequent reads that have been validated and then committed.

There have been no following readings that have been validated in this area as of yet.

Algorithm for a Deadlock:

The distributed wound-wait locking algorithm is the one that comes in at number four on the list of algorithms. The 2 PL procedure serves as an example for this method, which largely follows that procedure. This technique takes a different approach to solving the deadlock problem than 2PL does. Whereas 2PL stores information about waits-for and then checks for local and global deadlocks, this solution uses timestamps to solve the problem and eliminate the possibility of deadlocks occurring. The key to understanding the difference lies in this distinction. Because each transaction is given a one-of-a-kind number that corresponds to the exact moment it was started, more recent transactions are unable to force older transactions to hold their place while they complete. If a younger transaction attempts to acquire a lock while an older transaction is already waiting for it, the younger transaction will be "wounded," which means that it will be restarted, unless the younger transaction has already entered the second phase of its commit procedure. If the request makes an older transaction wait for a younger transaction, this result will occur. The ability for more recent transactions to wait for older transactions reduces the likelihood of there being a stalemate in the system.

t(T1) came before t(T2) because: If the transaction [t(T1)] that is requesting for the lock is older than the transaction [t(T2)] that currently has the lock on the data item that is being requested, the transaction [t(T1)] that is asking for the lock must either abort or rollback.

## VI. CONCLUSIONS

centralized and decentralized administration systems for databases Distributed database systems are becoming increasingly important to the operations of businesses. As a result, we are left with no other option but to take measures to protect not only the authenticity of these systems but also the atmosphere in which they function. The protection of information in its confidentiality, integrity, and accessibility are the primary goals of security measures, regardless of the format the information may take. When it comes to helping with the management of the security of distributed databases, there is a wide variety of tools and approaches available to choose from. In this post, we are going to talk about the fundamental idea that lies behind parallel and distributed concurrency management, as well as the various approaches that may be used to control concurrency in distributed environments. In order for a database to function in an efficient manner, it is absolutely necessary for it to exhibit the ACID properties.

Currently, we are conducting research on a variety of methods for enhancing the speed of transactions that require a high level of security without sacrificing the safety of these transactions. In addition, we are concentrating on the defense of real-time distributed systems in an effort to discover methods that can improve the effectiveness of transactions requiring a high level of security without compromising that level of security.

## Reference

1. 1.Bernstein, P. A., & Goodman, N. (1984). An algorithm for concurrency control and recovery in replicated distributed databases. ACM Transactions on Database Systems (TODS), 9(4), 596-615.
2. 2.Rothnie Jr, J. B., Bernstein, P. A., Fox, S., Goodman, N., Hammer, M., Landers, T. A., ... & Wong, E. (1980). Introduction to a system for distributed databases (SDD-1). ACM Transactions on Database Systems (TODS), 5(1), 1-17.
3. 3.Bernstein, P. A., & Goodman, N. (1980). Fundamental Algorithms for Concurrency Control in Distributed Database Systems. COMPUTER CORP OF AMERICA CAMBRIDGE MA.
4. 4.Ray, C. (2009). Distributed database systems. Pearson Education India.
5. 5.Corbett, James C., Jeffrey Dean, Michael Epstein, Andrew Fikes, Christopher Frost, Jeffrey John Furman, Sanjay Ghemawat et al. "Spanner: Google's globally distributed database." ACM Transactions on Computer Systems (TOCS) 31, no. 3 (2013): 1-22.
6. 6.Badal, D. Z. (1979, November). Correctness of concurrency control and implications in distributed databases. In COMPSAC 79. Proceedings. Computer Software and The IEEE Computer Society's Third

International Applications Conference, 1979. (pp. 588-593). IEEE.

7. 7.Ramakrishnan, R., Gehrke, J., & Gehrke, J. (2003). Database management systems (Vol. 3). New York: McGraw-Hill.

8. 8.Rosenkrantz, D. J., Stearns, R. E., & Lewis, P. M. (1978). System level concurrency control for distributed database systems. ACM Transactions on Database Systems (TODS), 3(2), 178-198.

9. 9.Garcia-Molina, H. (1983). Using semantic knowledge for transaction processing in a distributed database. ACM Transactions on Database Systems (TODS), 8(2), 186-213.

10. 10.Pokorný, J. (2015, June). Database technologies in the world of big data. In Proceedings of the 16th International Conference on Computer Systems and Technologies (pp. 1-12).