Concurrency Control and Security Issue in Distributed Database System

Akshay M. Gupta[1], Yogesh R. Gore[2]
[1]Research Scholar, Department of Computer Science and Information Technology, Veermata Jijabai Technological Institute, Mumbai, Maharashtra, India.

***Abstract* - This paper reviews the security issues and concurrency control in distributed database system and data security aspects of client and server. As distributed database became more popular, the need for improvement in distributed database management system become even more important. The most important issue is security that may arise and possibly compromise the access control and the integrity of the system. In this paper, we propose some solution for some security aspects such as multi-level access control, confidentiality, reliability, integrity and recovery that pertain to a distributed database system. This paper review some algorithm based of concurrency control such as basic timestamp algorithm, distributed two phase locking protocol (2PL), distributed optimistic protocol, wound-wait algorithm.**

***Keywords* - Distributed Database, Security aspects, Multi-level access control, Concurrency control.**

I. INTRODUCTION

A Distributed [3] database is a collection of databases which are distributed and then stored on multiple computers (otherwise called sites) within a network. All sites participating in the distributed database enjoy local autonomy in the sense that the database at each site has full control over itself in terms of managing the data. Also the sites can inter-operate whenever required. A database [4] link connection allows local users to access data on a remote database for establishing these connections, each database in the distributed system must have a unique global database name in the network domain. The global database name uniquely identifies a database server in a distributed system. Which means users have access to the database at their location so, that they can access the data relevant to their task without interfering with the work of others? The Distributed database management system (DDBMS) is software that permits the management of the distributed database and makes the distribution transparent to the user. The main difference between centralized and distributed database is that the distributed databases are typically geographically separated and are separately administrated between local & global transactions. A local transaction is one that access data only from sites where the transaction originated , A global transaction on the other hand is one that either access data in a site different from the one at which the transaction was initiated or, accessed data in several different site .

In this paper we will review the security concern of databases and distributed databases in particular. The security problems found in both models will be examined. Moreover, we will evaluate the security problems unique to each system finally; the comparison is done relative merits of each model with respect to security.

II. DISTRIBUTED DATABASE SYSTEM CONCEPT

The concept of distributed database came into existence during mid – 1970. It was felt that many applications would be distributed in future and therefore the database had to be distributed also. Actually a distributed database system (DDBS) is a collection of several logically related databases which are physically distributed in different computers or sites over a computer network [17]. The users of distributed database have the impression that the whole database is local excepted for the possible communication delay between the sites. This is because a distributed database is a logical union of all the sites and the distribution is hidden from the users. DDBS is preferred over a non-distributed or centralized database system for various reasons. Distributed is quite common in an enterprise.

The design of responsible distributed database system is a key concern for information system. In high band-width network, latency and local processing are the most significant factors in query and update response time. Parallel processing can be used to minimize their effects, particularly if it is considered at design time. It is the judicious replication that enables parallelism to be effectively used. Distributed database design can thus be seen as an optimization problem requiring solutions to various interrelated problems: data fragmentation, data allocation and local optimization. Concurrency control (CC) is another issue among database system. It permits user to access a distributed database in a multi-programmed fashion which preserving the illusion that each user is executing alone on a dedicated system. Another activity of concurrency control (CC) is to "Co-ordinating, concurrent accesses to a database in a multi user database management system (DDBMS). There are numbers of algorithms that provides Concurrency control, such as two phase locking, Time stamping, Multi-version timestamp, and Optimistic non- locking mechanism. Some methods provide better concurrency control than other depending on the systems.

Concurrency control (CC) is another issue among database system. It permits user to access a distributed database in a multi-programmed fashion which preserving the illusion that each user is executing alone on a dedicated system. Another activity of concurrency control (CC) is to "Co-ordinating [18], concurrent accesses to a database in a multi user database management system (DDBMS). There are numbers of algorithms that provides Concurrency control, such as two phase locking, Time

stamping, Multi-version timestamp, and Optimistic non- locking mechanism. Some methods provide better concurrency control than other depending on the systems [5].

III. ARCHITECTURE OF DISTRIBUTED DATABASE

A distributed database (DDB) is a collection of data that logically belongs to the same system but is spread over the sites of a computer network. It is not necessary that database system have to be geographically distributed. The sites of the distributed database can have the same network address and may be in the same room but the communication between them is done over a network instead of shared memory. As communication technology, hardware, software protocols advances rapidly and prices of network equipment falls every day, developing distributed database systems become more and more feasible. Design of efficient distributed database is one of the major research problems in database & information technology areas.

A distributed database management system (DDBMS) is then defined as the software system that permits the management of the DDB and makes the distribution transparent to the users. Distributed database system (DDBS) is the integration of DDB and DDBMS. This integration is achieved through the merging the database and networking technologies together. Or it can be described as, a system that runs on a collection of machines that do not have shared memory, yet looks to the user like a single machine. Assumptions regarding the system that underlie these definitions are:

1. Data is stored at a number of sites. Each site is assumed to logically consist of a single processor. Even if some sites are multiprocessor machines, the distributed DBMS is not concerned ith the storage and management of data on this parallel machine.
2. The processors at these sites are interconnected by a computer network rather than a multiprocessor configuration.
3. To form a DDB, distributed data should be logically related, where the relationship is defined according to some structural formalism, and access to data should be at a high level via a common interface. The typical formalism that is used for establishing the logical relationship is the relational model.
4. The system has the full functionality of a DBMS.

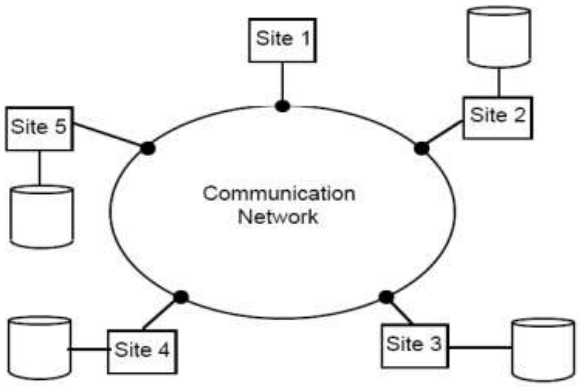

Figure 1: A Distributed Database Environment

IV. SECURITY CONCERN IN DISTRIBUTED DATABASE

Security in distributed database: Security means protection of information and information system from unauthorized access, modification and misuse of information or destruction. The basic of distributed database security is to deal with protecting data from people or software having malicious intentions. Distributed systems pose four main security components, security authentication, authorization, access control and encryption. Security breaches are typically categories as unauthorized data observation, incorrect data modification and data unavailability [2].

DBMS have many of the same security requirements as operating systems, but there are significant differences since the former are particularly susceptible to threat of improper disclosure, modification of information and also denial of service [10].

Security Authentication: Authentication stands for verifying the identity of someone as user or device who wants to use data, resources, or applications. Authentication also enables accountability by making it possible to link access and actions to specific identities. Authentications is realized by "smart token" which is a hardware device in the size of a pocket computer or credit card that creates a password and transfer it to the authentication server that is linked up to the network.

Authorization: Authorization is a security mechanism. It is used to determine user/client privileges or access levels related to system resources. The only authorized users are allowed to access the service of system.

Unauthorized data observation result in the disclosure of information to users not entitles to gain access to such information [2].

Encryption: It is the technique of encoding data that only authorized users can understand it. A number of industry standard encryption algorithms are useful for the encryption and decryption of data on the server, some most popular algorithms are RSA, DES, PGP.

Multi-level access control: In a multi-level access system, user are limited from having complete data access. Policies restricting user access to certain data parts may result from secrecy requirement or, they may result from loyalty to the principal of least privileged (a user only has access to relevant information). Access policies for multi-level system are typically to as either open or, closed. In an open system, all the data is considered unclassified unless access to a particular data element is expressly prohibited. A closed system is just the opposite: in this case access to all data is prohibited unless the user has specific access privileged. Security in distributed, database system has focused on multi-level security. Specifically approaches based on distributed data and

centralized control architectures were proposed .Prototypes based on these approaches were also developed during the late 1980s and early 1990s.

Data security aspects of client / server:

There are five approaches in security aspect of client/server

- The work station approval mechanism of users may be partial or non – existent.
- It is possible to carry out automation of the Login procedure.
- The work station may be installed in a public area or in a high risk area.
- The work station may active strong utilities or development devices and thereby tries to bypass the security mechanism.
- In extreme cases the users may pretend to be another user and infiltrate the system.

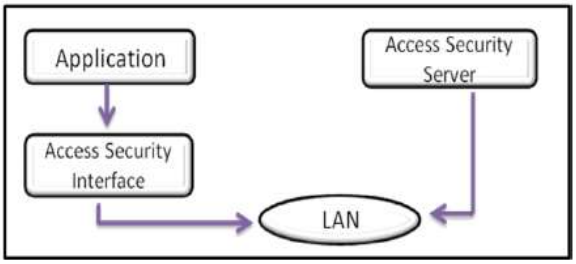

Fig 5 : Distributed System's Security Management

V. DISTRIBUTED CONCURRENCY CONTROL

Concurrency control: It is the activity of processing concurrent accesses to a database in Distributed Database System. The most common distributed concurrency control technique is strong strict two-phase locking. In database systems and transaction processing, distributed concurrency control refers primarily to the concurrency control of a distributed database [19]. The major goal for distributed concurrency control is distributed serializibility for multi database systems. The Concurrency problem is the activity of coordinating concurrent access to a database in a multi user Management System (DBMS) [14]. Concurrency control is another issue among database system. It permits users to access distributed database in multiprogramming passion while preserving the illusion that each user is executing alone on a dedicated system [10].

Serializability: Concurrency control constrains the behaviour of transactions that operate concurrently so they cannot interfere with each other and cause inconsistent database states. Serializability is one means for achieving consistency in a shared database system. Serializability requires a set of concurrent transactions to be executed in such a way that they have the same result as some serial execution of the transactions. This section examines the effects of multilevel security serializability in a single-site database system [9].

Basic Timestamp Ordering Algorithm

Time Stamp can be used to determine the out datedness of a request with respect to the data object it is operating on and to order events with respect to one another. Timestamp ordering is one of the consistency preserving algorithms that is used in distributed database [16].

Timestamp is a unique identifier used to identify a transaction. In this algorithm transaction is ordered based on their timestamp values. The timestamp-ordering protocol ensures serializability among transaction in their conflicting Read and Write operations [20]. This is the responsibility of the protocol system that the conflicting pair of tasks should be executed according to the timestamp values of the transactions.

Rule:

- Transaction T issue R (A) operation.

  If WTS>TS (T)
  Then rollback T
  Otherwise
  Execute R (A) successfully
  and set RTS (A) = max {RTS (A), TS (T)}.

- Transaction T issues W (A) operation.

  If RTS (A) > TS (T)
  Then rollback T
  If WTS (A) > TS (T)
  Then rollback T
  Otherwise
  Execute W (A) successfully
  and set WTS (A) =TS (T)

Distributed Two-Phase Locking (2PL)

In order to ensure serializability of parallel executed transactions elaborated different methods of concurrency control. One of these methods is locking method. There are different forms of locking method. Two phase locking protocol is one of the basic concurrency control protocols in distributed database systems. The main approach of this protocol is "read any, write all".

Transactions set read locks on items that they read, and they convert their read locks to write locks on items that need to be updated. To read an item, it suffices to set a read lock on any copy of the item, so the local copy is locked; to update an item, write locks are required on all copies. Write locks are obtained as the transaction executes, with the transaction blocking on a write request until all of the copies of the item to be updated have been successfully locked. All locks are held until the transaction has successfully committed or aborted. The 2PL Protocol oversees locks by determining when transactions can acquire and release locks. The 2PL protocol forces each transaction to make a lock or unlock request in two steps:

- Growing Phase: A transaction may obtain locks but may not release any locks.
- Shrinking Phase: A transaction may release locks but not obtain any new lock.

The transaction first enters into the Growing Phase, makes requests for required locks, then gets into the Shrinking phase where it releases all locks and cannot make any more requests. Transactions in 2PL Protocol should get all needed locks before getting into the unlock phase. While the 2PL protocol guarantees serializability, it does not ensure that deadlocks do not happen. So deadlock is a possibility in this algorithm, Local deadlocks are checked for any time a transaction blocks, and are resolved when necessary by restarting the transaction with the most recent initial start-up time among those involved in the deadlock cycle. Global deadlock detection is handled by a "Snoop" process, which periodically requests waits-for information from all sites and then checks for and resolves any global deadlocks.

Distributed Optimistic Protocol (OPT)

The third algorithm is the distributed, timestamp-based, optimistic concurrency control algorithm which operates by exchanging certification information during the commit protocol. For each data item, a read timestamp and a write timestamp are maintained. Optimistic concurrency control can provide serializability under the restrictions imposed by our two basic assumptions. If concurrency control is not used, non-serializable execution orders can be produced by concurrent transactions [9].

Transactions may read and update data items freely, storing any updates into a local workspace until commit time. For each read, the transaction must remember the version identifier (i.e., write timestamp) associated with the item when it was read. Then, when all of the transaction's cohorts have completed their work, and have reported back to the master, the transaction is assigned a globally unique timestamp. This time stamp is sent to each cohort in the "prepare to commit" message, and it is used to locally certify all of its reads and writes as follows [20].

A read request is certified if-:

- The version that was read is still the current version of the item.
- No write with a newer timestamp has already been locally certified.

A write request is certified if-:

- No later reads have been certified and subsequently committed.
- No later reads have been locally certified already.

Wait-Die & Wound-Wait Algorithm:

The forth algorithm is the distributed wound-wait locking algorithm. It follows the same approach as the 2 PL protocol. The difference lies in the fact that it differs from 2PL in its handling of the deadlock problem: unlike 2PL protocol, rather than maintaining waits-for information and then checking for local and global deadlocks, deadlocks are prevented via the use of timestamps in this algorithm. Each transaction is numbered according to its initial start-up time, and younger transactions are prevented from making older ones wait. If an older transaction requests a lock, and if the request would lead to the older transaction waiting for a younger transaction, the younger transaction is "wounded" – it is restarted unless it is already in the second phase of its commit protocol. Younger transactions can wait for older transactions so that the possibility of deadlocks is eliminated.

Example-: Wound-Wait algorithm

	T1 is allowed to
t(T1) > t(T2) ⟶	Wait
t(T1) < t(T2) ⟶	Abort and rolled back

t(T1) > t(T2) -: If requesting transaction [**t(T1)**] is younger than the transaction [**t(T2)**] that has holds lock on requested data item then requesting transaction [**t(T1)**] has to wait. **t(T1) < t(T2) -:** If requesting transaction [**t(T1)**] is older than the transaction [**t(T2)**] that has holds lock on requested data item then requesting transaction [**t(T1)**] has to abort or rollback.

VI. CONCLUSIONS

Distributed database systems are a reality. Many organizations are now deploying distributed database systems. Therefore, we have no choice but to ensure that these systems operate in a secure environment and integrity. Security is concerned with the assurance of confidentiality, integrity, and availability of information in all forms. There are many tools and techniques that can support the management of distributed database security. We discuss the basic concept of concurrency control in distributed database systems and also discussed the various techniques for concurrency control in distributed environments. It is really important for database to have the ACID properties to perform.

We are in the process of investigating schemes by which the performance of high security level transactions can be improved without compromising with the security. Further we are looking to secure real time distributed systems by which the performance of high security level transactions can be improved without compromising the security.

VII. REFERENCES

[1]. Philip A. Bernstein, Vassos Hadzilacos and Nathan Goodman, "Concurrency Control and Recovery in Database Systems", Addison-Wesley (1987).

[2]. Elisa Bertino And Ravi Sandhu, "Database Security—Concepts, Approaches, and Challenges", IEEE Transactions On Dependable And Secure Computing, Vol. 2, No. 1, January-March 2005.

[3]. Bell, David and Jane Grisom, "Distributed Database Systems. Workinham", England: Addison Wesley, 1992.

[4]. Charles P. Pfleeger and Shari Lawrence Pfleeger, Security in Computing, Prentice Hall Professional Technical Reference, Upper Saddle River, New Jersey, 2003.

[5]. Charles P. Pfleeger, "Security in Computing", New Jersey: Prentice Hall. 1989.

[6]. Navathe Elmasri, "Database Concepts", Pearson Education, IV edition (2003).

[7]. M. Tamer Ozsu and Patrick Valduriez, "Principles of Distributed Database Systems", Prentice-Hall, II edition,(1999).

[8]. Gupta V.K., Sheetlani Jitendra, Gupta Dhiraj and Shukla Brahma Datta, "Concurrency control and Security issues in Distributed database system", Vol. 1(2), 70-73, August (2012).

[9]. Alan R. Downing, Ira B. Greenberg, and Teresa F. Lunt , "Issues In Distributed Database Security", SRI International(1990).

[10]. Zakira Suliman Zubi "On Distributed Database Security Aspects", IEEE, 2009.

[11]. Abdou R. Ali and Hany M. Harb , "Two Phase Locking Concurrency Control in Distributed Database with N-Tier Architecture", IEEE0-7803-8575-6/04.

[12]. Resilient Concurrency Control in Distributed Database Systems IEEE TRANSACTIONS ON RELIABILITY, VOL. R-32, NO. 5, DECEMBER 1983.

[13]. The Concurrency Control Mechanism of SDD-1: A System for Distributed Databases (The Fully Redundant Case) IEEE Transactions On Software Engineering, Vol. Se-4, No. 3, May 1978.

[14]. Obaidah A. Rawashdeh, Hiba A. Muhared and Nedhal A. Al-Sayid, "An Optimistic Approach in Distributed Database Concurrency Control", 2013 5th International Conference on Computer Science and Information Technology (CSIT).

[15]. Database Security—Concepts, Approaches, and Challenges, Elisa Bertino, Fellow, IEEE, and Ravi Sandhu, Fellow, IEEETRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, VOL. 2, NO. 1, JANUARY-MARCH 2005

[16]. Subir Verma, "Performance Evaluation of the Timestamp Ordering Algorithm in a Distributed Database", IEEE Transactions On Parallel And Distributed Systems, Vol. 4, No. 6, June 1993.

[17]. A. A. Akintola, G. A. Aderounmu, and A. U. Osakwe", Performing Modeling of an Enhanced Optimistic Locking Architecture for Concurrency Control in a Distributed Database System",ACM vol.37, No.4, November 2005.

[18]. Simon Wiseman, DERA, Database Security: Retrospective and Way Forward, 2001.

[19]. Sheetlani Jitendra and Gupta V.K., "Concurrency Issues of Distributed Advance Transaction Proces", Res. J. Recent Sci., 1(ISC-2011), 426-429 (2012).

[20]. Bernstein P and Goodman N, "Timestamp-Based Algorithms for Concurrency Control in Distributed Database Systems," Proc. 6th VLDB Cot& Mexico City, Mexico,Oct.1980