A Review on Synchronization and Concurrency Control Techniques of Distributed Databases

Sajawal ur Rehman Khan[1], Muhammad Sheraz Arshad Malik[2], Muhammad Waleed Ashraf[1], Ayesha Saif Ullah[1], Ikraam Asghar[1], Nafeesa Razzzaq[1]
College of Computing Riphah International University Islamabad Pakistan[1]
Department of Information Technology, Government Collage University Faisalabad Pakistan [2]

Abstract
Use of distributed databases is greater than ever. Distributed database is an arrangement of databases that can be put away at various physical areas however intelligently inter related. As data is spread at several sites in distributed databases, it needs to be synchronized. Synchronization is important due to the fact that it allows data to be consistent. When multiple transactions are to be performed on a variable at the same time, the need for concurrency control mechanism emerges. This paper is concerned with different techniques used for the synchronization and concurrency control of distributed databases.

Keywords:
Distributed Database, Synchronization, Concurrency Control, Clock synchronization, Locking, Timestamp.

1. Introduction

A database is a set of data related to one or more organizations with a definite goal to achieve. Databases can be categorized into a centralized database and distributed database [1]. In the centralized database, Database and Database Management System (DBMS) are set on a solitary PC or site. While in distributed database, Database and DBMS are on various locales. DBMS is software which handles different operations storage, update, retrieval etc. in a computer system.
Figure 1 demonstrates a distributed database (DDB) which is a gathering of various; intelligently interrelated databases dispersed over a PC arrange [3] to deal with the DDB we require software as distributed database management system (DDBMS). Distributed database enables asset sharing between systems [2]. It basically fills in as an interface between the database and clients or application programs. DDB can further be partitioned into homogeneous and heterogeneous distributed databases. In homogeneous the data is distributed but all sites run the same Database Management System (DBMS) software [1], while in heterogeneous distinctive DBMS for different network sites?
In DDB, information is conveyed over various PC systems utilizing replication in which a successive duplicating of information starting with one database then onto the next database is performed. Be that as it may, replication of information does not guarantee a dispersion of completely reliable information. To conquer this issue of inconsistency we utilize synchronization [5]. Synchronizing the conveyed information in DDB is imperative as various operations are performed on the information, so information should stay reliable.

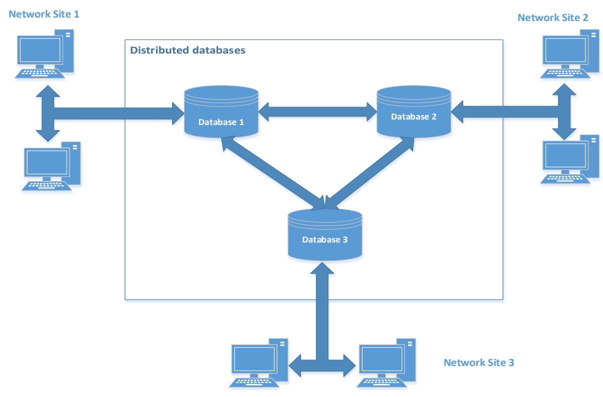

Fig. 1 Distributed Database

Synchronization is a part of replication which guarantees each duplicate of the database on different sites contains similar data and object [6]. Synchronization maintains data integrity and it keeps multiple copies of a dataset [8]. Synchronization is of two types; one-way and two-way synchronization. In one-way sync data is transferred from master to slave while in two-way synchronization data is transferred from master to slave and other way around [9].
Concurrency control allows several transactions to be performed simultaneously. It basically implies synchronization of concurrent transactions. It guarantees that database is always in a consistent state by ensuring a specific access order for data items. Is also amplifies the concurrency of exchanges.
Remaining paper is organized as follows: Section 2 comprised of data synchronization techniques, Section 4 discusses some algorithms of clock synchronization, Section 4 describes the synchronization for homogeneous databases using audit log, Section 5 introduces the

Manuscript received February 5, 2019
Manuscript revised February 20, 2019

concurrency control in distributed databases, Section 6 concludes the paper and at the end references are given.

2. DATA Synchronization Techniques

In spite of the fact that synchronization of information in distributed databases is not a simple task but rather a considerable measure of work has been done as of late on synchronization. There are two main methods of data synchronization in distributed databases named as incremental synchronization and complete synchronization [4]. In incremental synchronization just refreshed or updated data is transferred to a central database to ensure data synchronization. While, in complete synchronization data of all the databases is needed to be transferred to a central database for one time, and transmitted data is updated synchronously by the central databases. This is straightforward yet requires a huge network overhead. Incremental synchronization in this sense is superior to complete synchronization because the prior diminishes the system overhead, but acquiring incremental data is a challenge. Trigger, Timestamp and scanning are some of the methods to obtain incremental data.

3. Clock Synchronization

As there are numerous devices involved in distributed databases so they regularly require accurate timing in order to encourage synchronization and data correlation. Clock synchronization involved a system network which should maintain universal time. For web-based applications it is necessary to have a universal notion. Each system would also have its own particular clock but clock drift makes it inaccurate and inconsistent. Clock drift means a change in a clock's perspective of time. To ensure the understanding between different clocks of various systems in a network, the drifts among individual clocks which can be achieved by sharing messages containing information about current state of their clocks [11].

Cristian's algorithm, Berkeley algorithm, Network Time Protocol (NTP) and Lamport's Algorithm are some of the algorithms which would be talked about in this paper.

3.1 Cristian's algorithm

Cristian's algorithm is a method for clock synchronization using physical clocks. It was introduced in 1989 by Flaviu Cristian. This algorithm works on the principle of getting a common view of time among different network entities of a distributed system by sending a request to a time server and obtaining the time [11]. But this obtained time cannot be used for network delays and processing. Figure 2 shows that this can be done by measuring the time at which request is sent to the time server (T0) and the time at which response is received (T1). Overhead can be calculated by (T1-T0)/2. New time can now be represented as:

T new= T-server + [(T1-T0)/2]

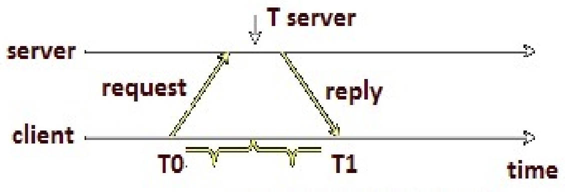

Fig. 2 Cristian's Algorithm

If the smallest time interval for a message to travel from client to server and vice-versa is Tmin, and the earliest time at which a timestamp is generated by the server is T 0 + Tmin. The latest time at which a timestamp could be generated by the server is T1 - Tmin. The range will be as follows:

Range: T1 – T0 - 2Tmin

Many requests are sent concurrently to the time server so, one of the requests might be delivered faster than other requests. Accuracy is improved by using this technique. The major disadvantage of this algorithm is that it depends on a single time server, if it fails whole synchronization process will fail.

3.2 Berkeley algorithm

This algorithm was developed by Zatti and Gusella in 1989 [11]. It does not depend on single time server so the advantage is that if time server fails whole synchronization process will not be at stake. It keeps clocks synchronized with each other. One system is master and others are slaves. It has no Universal time Coordinated (UTC) instead system manages its own time. There is a time daemon process which is running in order to calculate the time of individual systems. Time of all the systems is sent to a master system which calculates the average time and sends it to all the systems to synchronize. It is used for the internal synchronization of a group of computers.

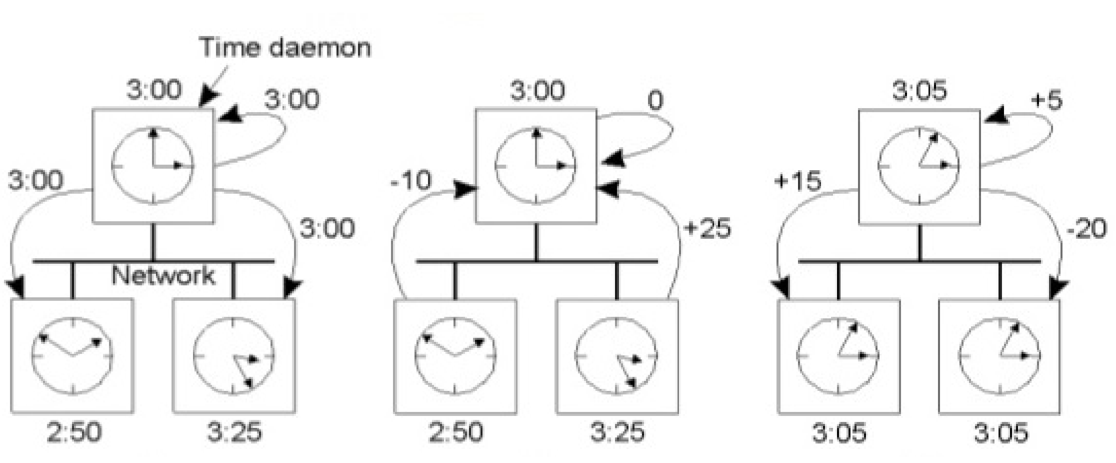

Fig. 3 Berkeley's Algorithm

Figure 3 shows that time demon asks all machines for their time clock. Machines answer and time demon tell them how to synchronize their clocks. In this case they are finally synchronized to 3.05 clock value. The main advantage of using Berkeley algorithm is that in the case of master system's failure a new master will be elected.

3.3 Network Time Protocol

It is a method of physical clock synchronization. It enables clients across the Internet to be accurately synchronized to UTC (universal time coordinated). Synchronization across the internet is difficult. NTP needs some reference clock that defines the true time to operate. All other clocks are set towards that time. It will make all systems agree on some time, and also make them agree to use a standard time.

Following is an example:

- Client across the internet sends request at time A=120.
- The server receives a request at time X=170.
- The server is slow, so it doesn't send out the response until time Y=180.
- The client receives the request at time B=140.
- Client determines the time spend on the network is B-A-(Y-X)=140-120-(180-170)=10 seconds
- Client assumes the amount of time it took to get a response from the server to the client is 10/2=5 seconds.
- The client adds that time to the time when the server sent the response to estimate that it received the response at time 185 seconds.
- The client now knows that it needs to add 45 seconds to its clock.

Simple Network Transfer Protocol (SNTP) is a subset of NTP not a new protocol. It is like Critian's algoritm.. It sets clock from the server.

3.4 Lamport's Algorithm

In distributed databases, data synchronization is a big challenge. It is not possible to determine which version of data is most recently updated just by using physical timestamp. Rather one can use logical clocks which are based on the order of events. Suppose we want event's order in such a way that it shows their relationship e.g. If an event A happens before event B, then A is not definitely caused by B but A might have influenced B. This can be shown as "happened-before" relations on events. This relation was proposed by Leslie Lamport and is defined as follows:

$A \longrightarrow B$: it describes A & B are within the same process and A occurred before B.

$A \longrightarrow B$ and $B \longrightarrow C$ then, $A \longrightarrow C$

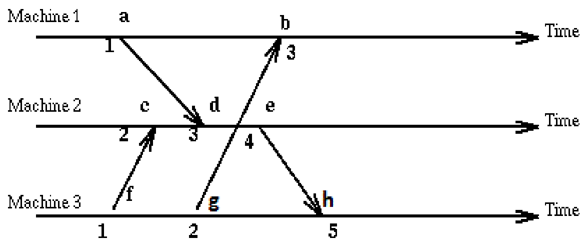

Fig. 4 Lamport's Algorithm

4. Synchronization in the Homogeneous database using Audit log

Distributed system has a major advantage to handle data. So that replication is one of the major useful ways to distribute data. So, synchronization is used to handle replicated data. It ensures that each database has same data and updating of data in database occurs periodically. One method of applying synchronization is Audit log which is different in each database. The audit log can be utilized to identify who accessed the database, what kind of activities and what data has been changed [10]. Every database has different audit log. So, Audit log table is used for the audit log.

4.1 Audit Log Table

Audit log table used all types of triggers of like INSERT, UPDATE, and DELETE. However trigger action cannot record Data Definition Language (DDL) activity and user activity. While audit trigger is the special database object which works on conditions automatically. In synchronized database, the table is created with audit-log prefix and then database name is used. Figure 5 shows the activity diagram of the creation of Audit log table [5].

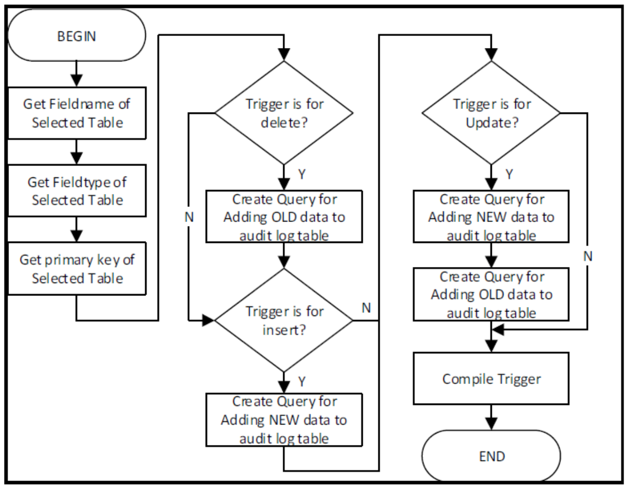
BEGIN
Get Fieldname of Selected Table
Get Fieldtype of Selected Table
Get primary key of Selected Table
Trigger is for delete?
Y
N
Create Query for Adding OLD data to audit log table
Trigger is for insert?
N
Y
Create Query for Adding NEW data to audit log table
Trigger is for Update?
Y
N
Create Query for Adding NEW data to audit log table
Create Query for Adding OLD data to audit log table
Compile Trigger
END

Fig. 5 Audit Log Creation

It will use different triggers, primary key, field name and field type of table will also be used for creating a table. Every trigger has different contents. Delete trigger will use old contents. Insert trigger will add new contents while update trigger uses both old and new contents. Every trigger will be marked I for insert, U for Update and D for delete.

4.2 Synchronization Process

Before synchronization process, synchronized database in early initial state at each site must be comparable. This is done to ensure synchronized data is still consistent. Synchronization starts when this condition satisfies. Audit log, id and last time stamp are necessary to record and start synchronization. The time stamp is used to check whether there is new data, changed data, or deleted data from the table at synchronized database according to id record and timestamp, also to make sure if there is overhaul towards the entire data in audit table which certainly will take a long time.

Table 1: Audit Log Table

ID	Timestamp		Cost in RS
01	2017-5-1	7:18:50	105
02	2017-5-1	7:18:40	104
03	2017-4-15	6:50:00	106
04	2017-3-16	5:51:05	103
05	2017-2-2	8:16:56	102

Table 1 shows an audit log table in which cost of the item in RS is changing according to time_stamp and is recorded. Hence you can easily retrieve the latest cost. Heterogeneous service is an integrated component within the Oracle database server and the enabling technology for the Oracle transparent gateway products.

5. Concurrency Control

In distributed databases concurrency control is a concept that is used to resolve conflicts with the concurrent accessing or changing of data. There are two types of concurrency control techniques in distributed databases; one is lock based protocol while other is timestamp protocol.

5.1 Lock-Based Protocols

Lock based protocols are divided into two types:

i. Binary Lock
ii. Shared/Exclusive Lock

In binary lock there are two conceivable states: Locked & Unlocked, which implies either a lock is acquired or not. Figure 6 shows that T1 is accessing the variable X which means T1 has acquired a lock on X so, T2 should not be allowed to access X. If the lock is released by T1 only then T2 should be allowed to access X.

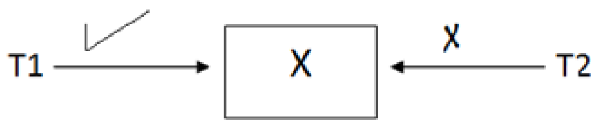

Fig. 6 Binary Locking

Figure 7 shows a shared lock which allows all transactions to read, which simply means that any transaction (T1, T2, and T3) can acquire the lock on that same data item (X) for reading purpose [13].

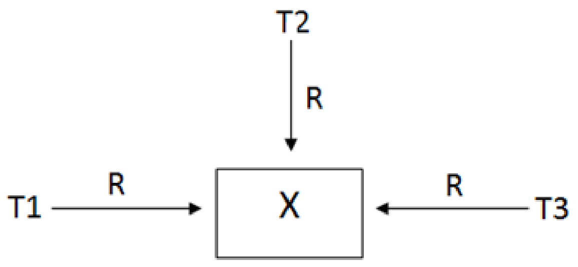

Fig. 7 Shared Lock

Figure 8 represents an exclusive lock which handles both read and write operations. For this situation if transaction T1 has acquired a R/W lock on data item X, no other transaction (T2, T3, and T4) can acquire either shared or exclusive lock on that data item X [13].

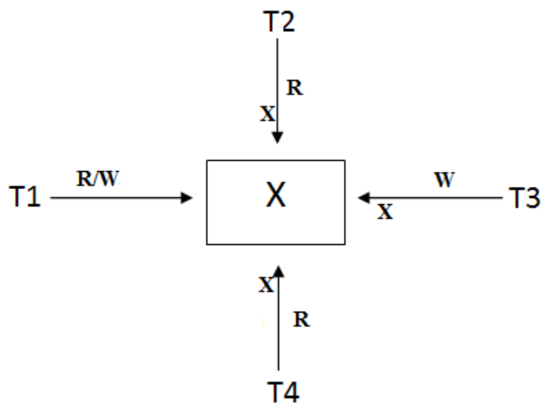

Fig. 8 Exclusive Lock

The relationship between shared and exclusive lock can be clarified using a lock matrix as in table 2. It demonstrates that when a shared lock is acquired, another shared lock can also be acquired. However, when an exclusive lock is acquired no other lock can be acquired.

Table 2: Lock Matrix

	Exclusive	**Shared**
Exclusive	0	0
Shared	0	1

5.1.1 Two Phase Locking (2PL)

It is a concurrency control technique. It also serves as synchronization technique [7]. The main purpose of 2PL is that it synchronizes the conflicts between current operations by detecting the read and write operations. Before executing any transaction the operations must own a lock. Following are two rules to grant lock(s) to any transaction:

1. The different transactions cannot simultaneously possess conflicting locks.
2. Once the transaction releases any locks, it may never obtain the extra locks.

There are two phases of 2PL; Growing phase and shrinking phase. In growing phase lock is acquired on a data item which is needed to be accessed. Locks are acquired either by shared or exclusive mode [14]. While, in shrinking phase acquired locks are released. Figure 9 illustrates the two phases of 2PL.

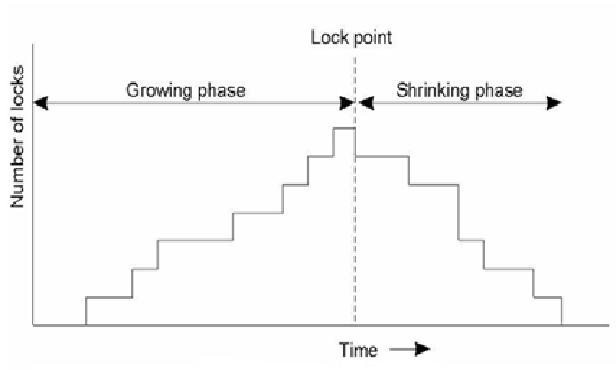

Fig. 9 Two-phase locking

5.2 Timestamp Based Protocols

Concurrency control in databases can best be accomplished by locking algorithms. But the drawback of locking algorithms is that they may lead to deadlocks which will then require deadlock detection, prevention and avoidance processes. Such complications can be dealt with utilizing timestamp based protocols [12]. A timestamp is associated with each transaction. Suppose there are two transactions T1 and T2, Figure 10 shows that:

Initially Timestamp of variable Y is,	T.S=0
When transaction T1 is accessing,	T.S=T
When transaction T2 is accessing,	T.S=T`

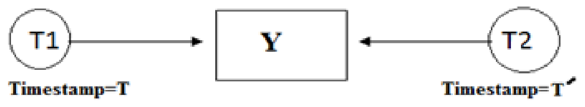

Fig. 10 Timestamp based concurrency control

In a timestamp based protocol timestamp of transactions is compared and then access is granted.

6. Conclusion

In recent years researchers have shown a lot of interest towards solving the issue of inconsistency and concurrency of data in distributed databases. Synchronization is one of the best possible solutions to provide a consistent data across different network sites. While concurrency control techniques solve the problem of concurrent data access. There are different techniques to achieve synchronization and concurrency control which are discussed in the paper.

References

[1]. Megha, P.T.a., An overview of distributed database. International Journal of Information and Computation Technology., 2014. Volume 4.

[2]. Ghodrati, Masoomeh, and Ali Harounabadi. "Provide a New Mapping for Deadlock Detection and Resolution Modeling of Distributed Database to Colored Petri Net." International Journal of Computer Applications 95.5 (2014).

[3]. Joshi, Himanshu, and G. R. Bamnote. "Distributed database: A survey." International Journal Of Computer Science And Applications 6.2 (2013).

[4]. Lu, Z. Y., et al. "A Method of Data Synchronization Based on Message Oriented Middleware and Xml in Distributed Heterogeneous Environments." Proceedings of International Conf. on Artificial Intelligence and Industrial Eng. no. Aiie. 2015.

[5]. GUDAKESA, RAI, et al. "TWO-WAYS DATABASE SYNCHRONIZATION IN HOMOGENEOUS DBMS USING AUDIT LOG APPROACH." Journal of Theoretical & Applied Information Technology 65.3 (2014).

[6]. Mazilu, Marius Cristian. "Database replication." Database Systems Journal 1.2 (2010): 33-38.

[7]. Yadav, Arun Kumar, and Ajay Agarwal. "A distributed architecture for transactions synchronization in distributed database systems." International Journal on Computer Science and Engineering 2.6 (1984): 2010.

[8]. Malhotra, Naveen, and Anjali Chaudhary. "Implementation of Database Synchronization Technique between Client and Server." International Journal of Engineering Science and Innovative Technology (2014).

[9]. Morgan, Daniel Patrick. Knowing heaven: Astronomy, the calendar, and the sagecraft of science in early imperial China. The University of Chicago, 2013.

[10]. Murray, Meg C. "Database security: What students need to know." Journal of Information Technology Education: Innovations in Practice 9 (2010): IIP-61.

[11]. Sharma, Chetna, and S. Pooja. "Distributed Clock Synchronization Algorithms: A Survey." (2016).

[12]. Rinki Chauhan and Preeti Bhati “Illustration of Timestamp Ordering in Controlling Concurrency in Distributed Database” International conference on Advanced Computing, Communication and Networks’11

[13]. Gajendra Singh “Demystifying Concurrency Control in DBMS” International Journal of ICT and Management

[14]. Rachael Harding, Andrew Pavlo, Dana Van Aken and Michael Stonebraker “An Evaluation of Distributed Concurrency Control” Proceedings of the VLDB Endowment, Vol. 10, No. 5, 2017