International Journal of Latest Trends in Engineering and Technology
Special Issue ICRMR-2018, pp. 075-083
e-ISSN:2278-621X

ANALYSIS OF DATABASE REPLICATION PROTOCOLS

Katembo Kituta Ezéchie[1], Shri Kant[2], Ruchi Agarwal[3]

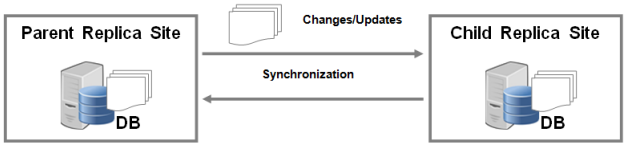
Abstract- In this paper it has been analyzed standard database replication protocols. The database replication consists of maintaining consistent multiple copies of database tables or relations, called replicas, on separate sites. First it has been rapidly presented the general context of the study followed by a review on generic database replication, with its famous synchronization algorithm, which is the unique strategy to manage exchange of data between databases in Distributed Database Systems (DDBS). To concretize the aim of this study, it has been presented some deficiencies collected during this study as follow: serialization of update transactions, reconciliation of updates, update of unavailable replicas in Eager or synchronous replication, sites autonomy and the independence of synchronization algorithm. Finally, this has been our motivation to propose an effective approach for synchronization of distributed databases over a decentralized peer-to-peer architecture.

Keywords – Database replication, Consistency, Eager and Lazy Synchronization

1. INTRODUCTION

In computing, Data replication involves sharing information between two or more redundant data sources those are linked by a computer network to ensure data consistency among themselves, to improve availability and performance [1]. In this logic, these data must be duplicated beforehand on the aforementioned sources. When we are referring to data, our interest is fixed on a Database that is a file for recording data in an organized and structured way.

However, the replication is one of the most used strategy for distributing or sharing data across Distributed Database Systems (DDBSs). So we can say that a DDBS is a set of logically connected or networked Databases, managed by different sites and appearing to the user as a single Database [2], [23]. In this way, the Database replication is the interchange of data between at least two Databases stored at different sites. According to [3], [4], [5], Database replication is a set of technologies for distributing Databases objects and copying or transferring data from one Database called "Master" or "Parent" to one or more than one "Slave (s)" or "Child (Children)" in order to maintain the consistency between them by the mean of synchronization procedures as shown here below in Fig.1.



Data replication protocol

It stores separate copies of the database or their portions at two or more sites. It is a popular fault tolerance technique of distributed databases [6], [23]. In a replication model the major problem turns out to be the maintenance of consistency between the data [7]. Therefore, setting up a replication procedure involves deciding which fragments or tables of the database will be replicated [8].

Nowadays, these technologies, the more usually, run under Database Management Systems (DBMSs). A DBMS is a software that manages the Database and provides access mechanisms or connection possibilities to users to connect their data shared and replicated on several locations. Since when it provides mechanisms to users to connect their Databases distributed over a computer network it is qualified Distributed Database Management System (DDBMS) [9], [23]. There are several DDBMSs of which we can name a few: SQL Server, FoxPro, MySQL, Oracle DB, PostgreSQL, Informix, Paradox, Sybase, Dbase, etc.

This paper has as aim to analyze the general database replication protocols. So to reach this aim, the armature of this document is structured as follows: the first section has serving as introduction to present the context under which this investigation is running, the second section will present the basic methods of database replication, the third will show some deficiencies and proposed solutions and finally the fourth section will conclude this study.

[1] Department of Computer Science & Engineering, School of Engineering and Technology, Sharda University, Greater Noida, India
[2] Department of Computer Science & Engineering, School of Engineering and Technology, Sharda University, Greater Noida, India
[3] Department of Computer Applications, JIMS Engineering Management Technical Campus, Greater Noida, India

2. GENERIC METHODS OF REPLICATION

Replication is not to be confused with a simple data backup because with this last method the data do not change over time and therefore reflect a fixed state, while the replicated data constantly evolves as changes are made on the data source. This is why replication sets itself the goals of improving data availability, improving performance and scalability and local autonomy [10].

2.1 Fundamental principles of replication

In the old database distribution environment, the only data distribution approach was based on a Client-Server architecture. In this model, the client connects to the DBMS to place orders. These orders are of two types: read (it is then a query, the SQL Select command) and update (these are transactions, the SQL Insert, Update and Delete commands) [11]. For transactions, there is a modification of the data on the server, but it remains short orders. On the other hand, in the case of a reading, there is no modification of the data but the treatments can be long and relate to a large mass of data [12]. It is therefore easy to understand that, in this approach, a large number of requests may partially (or completely) overload the server. There are several solutions to this kind of problem, thus the replication is one of them. There are four replication configurations: fully or complete replicated Database, partially or fragmented replicated Database, unreplicated or centralized database and selected replicated Database [15], [17], [18], [23].

The main goal of the replication is to facilitate access to data by increasing availability from one hand and reliability from another hand. Either because the data is copied to different sites for dispatching queries, or because a site can take over when the primary server crashes. Currently, the principle of replication involves at least two DBMSs. It is quite simple and takes place in three stages in general [13], [32]:

The Master database receives an update command (Insert, Update or Delete);

Changes made to the data are detected and stored (in a table, a file, a queue) for propagation;

A synchronization process propagate updates made on the Master to another (s) base (s) called slave (s).

2.2 Synchronization process

The synchronization is a process that establishes consistency between data from a Source or Master or Parent to a Destiny or Slave or Child and vice versa. This technology is parameterized and runs in time according to any harmonization that can make it trigger automatically or run manually, as appropriate to copy the data changes in one or two directions [19], [25], [26], [27].

The synchronization algorithm [32] has already overcome the problem which client-server knew in past few time. This algorithm solves the problem of Centralized Database Systems, such when all clients depend on a single central server and the database becomes inaccessible due to scheduled failures or Central Server failures. In this approach, all connected workers will no longer have access to their data. Therefore, the synchronization is a technique for working "online" and "offline", in the absence of the network or during server failures so that Data are stored in the user's site Database (Local Server), in order to be automatically synced with the server (Central Server) in serial order when the system recover from the failure [19], [20].

However, although the synchronization of copies has solved the problem of the Centralized Database Systems, but it remains the main difficulty of replication because it is responsible to maintain consistency between replicas and data availability. Thus, there are several synchronization techniques: synchronization timestamp based [10], triggers based [16], XML Based [20], stored procedure based [21], Message digest based [22], mathematical, flag and log based [28], etc.

2.3 Approaches for broadcasting updates

As soon as updates are made to a copy of the source Database "Master" then the next decision is to know by which method they will be propagated to the destination copies "Slave (s)" in order to maintain the reliability. Replica reliability is obviously a part of the data replication, which is in turn a method to overcome the problem of slow data access, low availability, fault tolerance, etc., in DDBSs. Two general approaches to manage updates propagation exist. These strategies depend on "when" parameter i.e. the need of knowing when updates are propagated. Thus, it emerges two update strategies as follow [1], [9], [10], [14], [15], [29]:

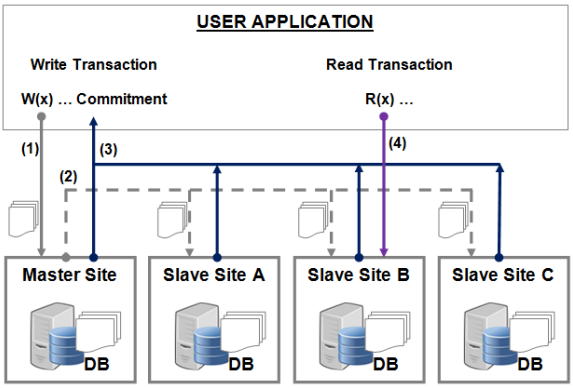
Eager synchronization or synchronous replication: This method recommends that all data replicas be updated in the same transaction as the write transaction. This transaction typically presents itself as a basic Two-Phase-Commit (2PC) Protocol. But after the operation all the replicas are coherent and bear the same physical state. Consequently, it is clear that disconnected sites can still block an update procedure. This strategy provides strong mutual consistency or reliability.

Lazy synchronization or asynchronous replication: The second method, in turn, introduces a new approach to overcome the difficulty of distributed locks. Its technicality prone to update a subset of replicas during the execution of an update transaction and then transmit the modification to the other replicas a little later. Only a part of replicas is updated. Other replicas are fetched up-to-date lazily after the commitments of the transaction. This process can be triggered by the commitment of the transaction or another periodically executing transaction. This approach provides weak mutual consistency or reliability.

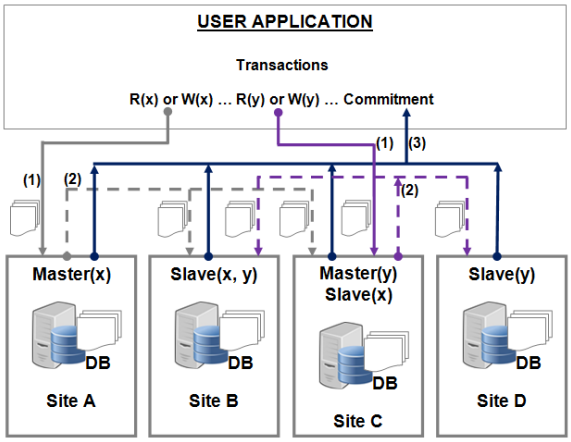
These replication dimensions are orthogonal with "where" parameter i.e. we need to know where updates are going to take

place. From this we have [1], [10], [14], [15]:
Single Master or Primary Copy with Limited or Full Replication Transparency (centralized): only one copy of the data is updated (master copy) and all others (secondary copies) are subsequently updated to reflect the changes in the master copy.
Update everywhere (decentralized or distributed): updates are done on any data copy i.e. all sites that have a copy of the data can perform the data update and the changes are replicated to all other copies.
Types of Replication Protocols
Assume a fully replicated database and each site work under a Two-Phase-Locking (2PL) concurrency control technique. Therefore, we have four possible combinations [1], [10], [14], [15], [29]:

2.4 Eager centralized
In this approach, on a master site, operations (mostly Write transactions) on a data element are conducted. These procedures are combined with strong consistency methods, so that the single update transaction which is committed using the Two-Phase-Commit (2PC) protocol performs logical data item updates.
First case: Single Master with Limited Replication Transparency
Let W(x) be a write transaction and R(x) be a read transaction where x is the replicated data item. The Fig.2, here below depicts the Eager Single Master Replication Protocol in the logic of Read-Any, Write-All (RAWA) or Read-One, Write-All (ROWA) using Time-Stamping Algorithm. The user submits the Write Transaction on the Master Site only and the Master Transaction Manager System forward synchronously these updates/Changes to Slaves. A Read-only Transaction can be submitted to anyone of Slaves Sites or the Master Site itself.



A Write transaction is performed on the DB on the Master Site;
Write is then despatched to other replicas;
At the commitment time updates become permanent;
Read transactions are routed to any slave copy.
Eager Single Master Replication Protocol
This case present one major drawback of overloading the Master Site by Write Transactions. As presented in the Fig.2, here above, every Write Transaction from each user application need to be deferred to the Master copy before being dispatching to Slaves. Moreover, one important issue that persist is to make the difference between an "update" transaction and a "read-only".
Second case: Single Master with Full Replication Transparency
This case overcomes the issue of Master overloading by Write Transactions from users. Thus apart from the Eager replica control algorithms coupled with Time-Stamping algorithm, the concurrency control which uses the coordinating Transaction Manager has been introduced.
However, the logic of Read-Any, Write-All or Read-One, Write-All is still keeping but using Transaction Management algorithm. The user application is alleviated to know the Master Site. Even if the implementation of this case is more complicated than the first alternative discussed but it is responsible to provide Full Replication Transparency, keeping the same schema depicted by the Fig.2, but introducing the Transaction Management algorithm.
Third case: Primary Copy with Full Replication Transparency
Let W(x) and W(y) be a write transactions and R(x) be a read transaction where x and y are replicated data items, successively first routed to Master(x) and Master(y). The Fig.3, here below depicts the Primary Copy with Full Replication Transparency with the supposition of fully replication. A is the Master Site storing the data x and B and C are Slave Sites containing replicas; in the same way, C is the Master Site that holds the data y with B and D its Slave Sites.



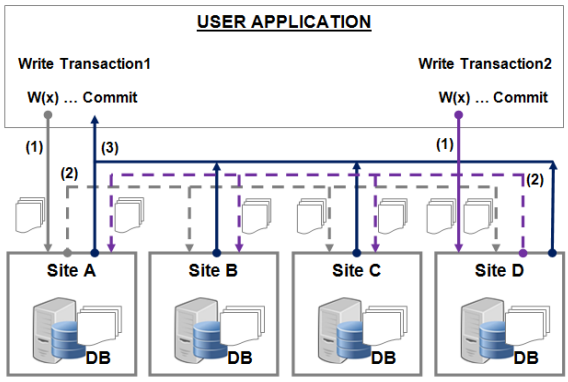
Read or Write transactions for all element of data are directed to that master of data items. A Write transaction is first performed at the Master;
Updates are then dispatched to the other replicas;
The commitment of Updates is performed.

2.5 Eager Primary Copy Replication Protocol

Primary copy method requires a sophisticated repertory at all sites, as well as a joint replica concurrency control technique, but it also overcomes some issues discussed in the previous approaches such that to reduce the load of the Master Site without producing a great volume of transmission among the transaction managers and the lock managers.

2.6 Eager distributed

Changes or updates can come from anywhere to first update the local replica and then send to other replicas. If the updates result from a site where there is no data element, it is sent to one of the replica sites, which in turn harmonizes the execution. The update transaction is responsible for fulfilling all these possibilities. The user is notified and the updates become permanent when the transaction commit.
Let W(x) be a write transaction where x is a data item duplicated at sites A, B, C and D. The Fig.4, here below depicts how two operations modify different copies (at two sites A and D). This procedure turns with the logic of Read-Any, Write-All or Read-One, Write-All constructed on the concurrency control techniques.

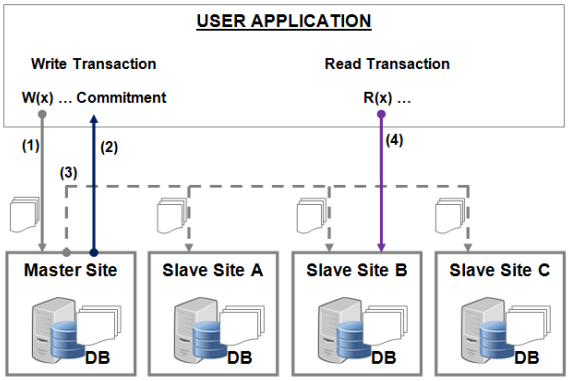


Two different Write transactions are simultaneously performed on two different local replicas of the same item of data;
Write transactions are transmitted to the other replicas independently;
At the time of commitment Updates become available.
Eager Distributed Replication Protocol

2.7 Lazy centralized

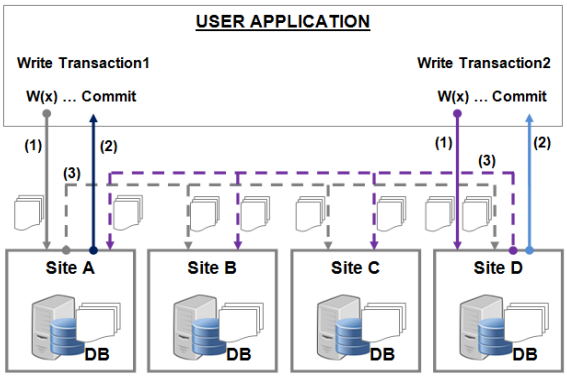
This protocol provides algorithms which are alike eager centralized replication. But in this procedure updates or modifications are first performed to a Master replica and then transmitted to the slaves. The greatest dissimilarity is that the modifications or updates can't be dispatched through the update transaction, but forwarded by a separate refresh transaction to Slaves, asynchronously after the transaction commits. Thus, it is possible that a read transaction from any Slave, anytime, reads an out-of-date copy as the updates are pushed to Slaves one lapse long after the Master's update.

First case: Single Master with Limited Transparency
Let W(x) be a write transaction and R(x) be a read transaction where x is the replicated data item. The Fig. 5, below, illustrates the sequence of execution steps for Single Master with partial sharpness. Here the write transactions are executed and deferred precisely on the main site (exactly as for Single Master). The second transaction, which we qualify as a refresh transaction, shares the updates to the slaves after validation of the first transaction. As soon as there is one master copy for all the data elements, the execution command is done according to the timestamp attached to each refreshing transaction at the site, according to the order of the commitment of the transaction. modification or actual update. Thus, in the timestamp order, Slaves would smear refresh transactions.



The modification is performed on the local replica.;
Updates become available as soon as transaction validation is successful;
A refresh transaction propagates updates to other replicas;
A read transaction is routed to a local copy of the slave.

2.8 Lazy Single Master Replication Protocol
When databases are partially replicated, a desired primary copy with a limited replication transparency approach makes sense if update transactions access only data items whose master sites are identical, since update are fully executed by a master. The same problem exists in the case of lazy primary copy, limited replication approach. The problem in both cases is how to design the distributed database so that meaningful transactions that can recognize the difference between an "update" and a "read only" transaction can be executed.
Second case: Single Master or Primary Copy with Full Replication Transparency
This protocol is an alternative that provides complete transparency by allowing the submission of read and write transactions to some site and then despatching them to the appropriate Master. This is delicate and involves two problems: the first is that, unless cautious, the global history in a serial order may not be definite; the second problem is that a transaction may not see its own updates.
So far, these two difficulties have found a partial solution: [10], proposed an algorithm respecting a chronological sequence of transactions executions. This algorithm is presented in the same way as that of centralized Eager, a primary copy with complete replication transparency case, but with the difference which is such that it makes it possible to retrace sequentially the history of serial transactions. Thus, a transaction does not start until the commitment of another, so lazily. Although this algorithm manages the first problem, but the second according to which a transaction does not see its own scripts remains unresolved. To solve this problem, it has been advocated to keep a list of all the modifications made by a transaction and to consult this list when executing a reading. Nonetheless, as soon as only the master knows the updates, he deviated more to keep the list and all transactions (reading as well as writing) must be executed on the master.
Lazy distributed
In the control of lazy distributed replicas, updates come from wherever, they are first run on the local replica, and then propagated to other replicas later. In this way, the read and write transactions are executed on the local copy and the commitment of the transaction is locally before the refresh transaction propagate updates to other sites.
Let W(x) be a write transaction where x is a duplicated data item at sites A, B, C and D. The Fig. 6, here below shows how two transactions modify or update two different copies (at sites A and D) and after commit the refresh forward updates to all sites.



Two modifications or updates are performed on two local copies;
Commitment of the transaction makes the modifications available;
The modifications or updates are self-reliantly transmitted to the other replicas.
Lazy Distributed Replication Protocol Actions
Once implemented, these protocol guarantees availability of updates on all sites, although propagated instantly. Nevertheless, it leaves the door open for analysis.
Direction of updates propagation
Both of these broadcasting updates approaches pointed out previously, use the synchronization process that has been indicated as being responsible for maintaining consistency between replicas or copies of the data. So we can distinguish [14], [16], [25], [28]:

2.9 Asymmetrical replication
It is a copy management technique based on a primary or master site only authorized to update and responsible for distributing the updates to others so-called secondary or slave copies. The primary site performs the checks and guarantees the correct scheduling of the updates. Note that the choice of an asymmetrical technique is orthogonal to the choice of a synchronous or asynchronous updates diffusion mode and one can thus distinguish the asymmetrical synchronous and asymmetrical asynchronous replication. This can run under One-way with Unidirectional Data Synchronization procedure.

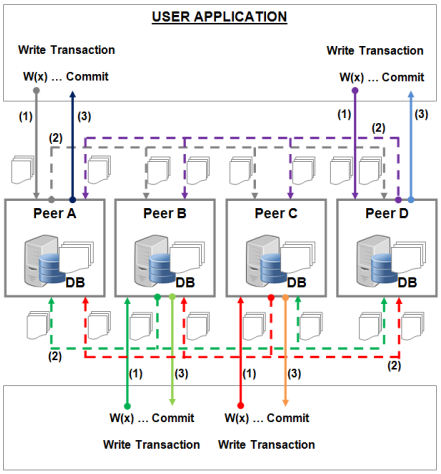
2.10 Symmetrical replication
In contrast to asymmetrical replication, no copy is preferred; so it allows simultaneous updates of all copies by different transactions. This is a copy management technique where each copy can be updated at any time and distributes updates to other copies. Note also that the choice of a symmetrical technique is orthogonal to the choice of synchronous or asynchronous diffusion mode of the updates and thus it can be distinguished symmetrical synchronous and symmetrical asynchronous replication. This can run under Two-ways with Bidirectional Data Synchronization procedure.
Deficiencies collected And proposed solutions
This section will present problems collected after the course of the analyze realized here above about database replication strategies. The replication problem has retained our attention; mostly while synchronization, which is the process of propagating modifications or updates to sites that hold the replicas of the fragment or replicas of the whole relation, in the case of full replication, is running.
Defficiecies and expected solutions
Assuming that the Database is full replicated, existing protocols present following problems:
Several updates carried by refreshing transactions, from different sites can reach a destination site at the same time but they cannot be performed on the same time [10], [31]. Proposed solution: an effective serialization algorithm.
These updates can be performed on different sites, simultaneously on different copies of the same data item, if they reach the destination like that then reliability or consistency will be lost and there will be the risk of redundancy [10], [28]. Proposed solution: an effective algorithm to reconcile updates.
During Eager or synchronous replication which is essentially Read-One/Write-All based, if some copies were unavailable at transaction time, they no longer be updated [10], [30]. But normally they should get these updates when they become available. Proposed solution: an effective approach taking in account Read-One/Write-All Available and Write-All Unavailable as soon as possible.
These protocols suffer from many problems which the introduction of the modern P2P systems should overcome. Some innovations of the modern P2P are: sites autonomy in the sense that each machine in its roles is identical to another, play same roles, and can leave and join the Network anytime [10], [24]. Proposed solution: an effective approach for synchronization over a decentralized P2P architecture, that can update several copies of the databases in real-time or near real-time.
Moreover, this approach should overcome one aspect of Distributed Databases homogeneity in the sense that it should allow replication between different DBMSs. So this synchronisation algorithm should be implemented independently of DBMSs as

a Mediator or used by designers when they need interaction between Databases managed by different DBMSs [10].

2.11 Proposed protocols

Assuming that the Database is full replicated, the proposed models of the Decentralized Peer-to-Peer Architecture are presented based on Eager replication and Lazy replication as follows:

Eager replication: Let W(x) be a write transaction where x is a replicated data item at Peers A, B, C and D. The Fig. 7, here below depicts how transactions update different copies at all Peers and before commit the refresh forward updates to all Peers.



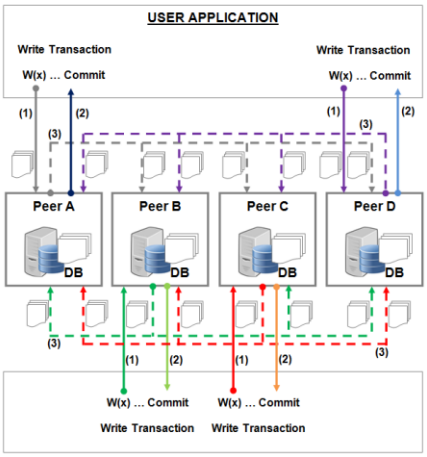
Updates are applied on all replicas;

The updates are dependently propagated to the other available replicas;

Transaction commit makes the updates permanent.

Eager Decentralized Peer-to-Peer Replication Protocol

Lazy replication: Let W(x) be a write transaction where x is a replicated data item at Peers A, B, C and D. The Fig.8, here below depicts how transactions update different copies at all Peers and after commit the refresh forward updates to all Peers.



Modifications are reflected to all replicas;

The commitment of a transaction makes the modifications stable;

The modifications are independently transmitted to the other data copies or replicas.

Lazy Decentralized Peer-to-Peer Replication Protocol Actions

3. CONCLUSION

In this paper it has been conducted an analysis of database replication protocols. As fragments or whole relations have in certain cases to exchange data among them, the Data replication, which is the unique strategy to manage this procedure, has been studied and its famous synchronization algorithm.

The main problem here is to maintain consistency among fragments or whole relations on different Sites. However, in this literature, we have indicated some issues which can violent consistency, such that: no serialization of update transactions, no

reconciliation of updates, no update of unavailable replicas in Eager or synchronous replication, no sites autonomy, no independent effective synchronization algorithm which can play the role of Mediator between different DBMSs.
Despite the correctness of all protocols studied earlier, since these problems just indicated here above are not solved then consistency can be broken anytime in replicated databases. Moreover, these protocols are unsuitable for P2P networks since they are either centralized or do not take into account the network limitations. On the other hand, existing P2P replication solutions do not satisfy all such requirements simultaneously. Thus this has been our motivation to propose "an effective approach for synchronization of distributed databases over a decentralized peer-to-peer architecture".
As future work we will write a complete, DBMS independent and effective algorithm in which it will be presented step by step scenarios to synchronize tables in an environment of replicated databases. It will be implemented as prototype in a Graphical Interface User, as a Mediator of DBMSs, to attempt to reach the aims of modern Peer-to-Peer in Distributed Database Systems.

4. REFERENCE

[1] S. Alireza, P. Saeid, and H. N. Ahmad, "Consistency of data replication protocols in database systems: a review", International Journal on Information Theory (IJIT), Vol. 3, pp. 19-32, October 2014.

[2] D. S. Hiremath and Dr. S. B. Kishor, "Distributed Database Problem areas and Approaches", Journal of Computer Engineering : National Conference on Recent Trends in Computer Science and Information Technology, pp. 2278-8727, 2016.

[3] Microsoft, SQL Server Replication, from Microsoft Documentation: https://docs.microsoft.com/en-us/sql/relational-databases/replication/sql-server-replication, Retrieved April 2018.

[4] MySQL, MySQL Server Replication, from MySQL 5.7 Reference Manual: https://dev.mysql.com/doc/refman/5.7/en/replication.html, Retrieved April 2018.

[5] Oracle, Data Replication and Integration, from Oracle Database 11g: http://www.oracle.com/technetwork/database/features/data-integration/index.html, Retrieved April 2018.

[6] C. Truică and A. Boicea, "Asynchronous Replication in Microsoft SQL Server, PostgreSQL and MySQL". International conference on cyber science and engineering, 2013.

[7] M. K. Sah, V. Kumar and A. Tiwari, "Security and Concurrency Control in Distributed Database System", International Journal of scientific research and management, (IJSRM), Vol. 2, pp. 2321-3418, 2014.

[8] M. Kaur and H. Kaur, "Concurrency Control in Distributed Database System", International Journal of Advanced Research in Computer Science and Software Engineering, Vol. 3, pp. 2277 128X, July 2013.

[9] S. Idowu and S. Maitanmi, "Transactions-Distributed Database Systems: Issues and Challenges", International Journal of Advances in Computer Science and Communication Engineering (IJACSCE), Vol. 2, pp. 2347-6788, March 2014.

[10] M. T. Özsu and P. Valduriez, "Principles of Distributed Database Systems (3rd ed.)". New York, USA: © Springer Science+Business Media, LLC, 2011.

[11] A. Srivastava, U. Shankar and S. Kumar, "Transaction Management in Homogenous Distributed Real-Time Replicated Database Systems", International Journal of Advanced Research in Computer Science and Software Engineering, Vol. 2, No. 6, pp. 2277 128X, 2012.

[12] Kaur M., & Kaur H., "Concurrency Control in Distributed Database System", International Journal of Advanced Research in Computer Science and Software Engineering, Vol. 3, No .7, pp. 2277 128X, July 2013.

[13] H. Anees and M. N. A. Khan, "Discovering Database Replication Techniques in RDBMS", International Journal of Database Theory and Application, Vol.7, pp. 93-102, 2014.

[14] G. Pucciani, F. Donno, A. Domenici and H. Stockinger, "Consistency of Replicated Datasets in Grid Computing", CERN, European Organization for Nuclear Research, CH1211Geneva 23, Switzerland, 2012.

[15] Nicoleta and Magdalena, "The Replication Technology in E-learning Systems", Procedia - Social and Behavioral Sciences, Publisher: Elsivier, Vol. 28, pp. 231 – 235, 2011.

[16] R. Gudakesa, M. Sukarsa and G. Sasmita, "Two-ways database synchronization in homogeneous DBMS using audit log approach", Journal of Theoretical and Applied Information Technology, Vol. 65, pp. 854-859, July 2014.

[17] S. Ceri and G. Pelagatti, Distributed Databases, Principles and systems. New York, USA: © McGraw-Hill, Inc. 2008.

[18] Singh and S. Singh, Distributed Database Systems : Principles, Algorithms and Systems, New-Delhi, INDIA : Khanna Book Publishing, CO. (P) LTD, 2015.

[19] N. Malhotra and A. Chaudhary, "Implementation of Database Synchronization Technique between Client and Server", International Journal Of Engineering And Computer Science, Vol. 3, pp. 7070-7073, July 2014.

[20] M. Kekgathetse and K. Letsholo, "A survey on database synchronization algorithms for mobile device, Journal of Theoretical and Applied Information Technology, Vol. 86, pp. 1817-3195, April 2016.

[21] M. Mali, Database Management System, Computer Science and Engineering-Information Technology, Mumbai University, India : Tech-Max Publications, Pune, September 2015.

[22] P. Kalyanakumar and A. Sangeetha, "A synchronization algorithm for mobile databases using SAMD", International Research Journal of Engineering and Technology (IRJET), Vol. 2, pp. 2395 -0056, May 2015.

[23] Nicoleta, and Costinela D., "Synchronous partial replication case study: implementing elearning platform in an academic environment", Procedia - Social and Behavioral Sciences, Publisher: Elsivier, Vol. 46, pp. 1522 – 1526, 2012.

[24] Q. Vu, M. Lupu and C. Ooi, "Peer-to-Peer Computing - Principles and Applications", Publisher: Springer, January 2010.

[25] M. Shodiq et al., "Implementation of Data Synchronization with Data Marker using Web Service Data", International Conference on Computer Science and Computational Intelligence (ICCSCI 2015), Procedia – Computer Science, Publisher: Elsivier, Vol. 59, pp. 366 – 372, 2015.

[26] M. Choi,. et al., "A Database Synchronization Algorithm for Mobile Devices", IEEE Transactions on Consumer Electronics, Vol. 56, No. 2, May 2010

[27] V. Balakumar and I.Sakthidevi, "An Efficient Database Synchronization Algorithm for Mobile Devices Based on Secured Message Digest", IEEE International Conference on Computing, Electronics and Electrical Technologies [ICCEET], 2012.

[28] F. Mohammad and S. “Udai, Data Synchronization in Distributed Client-Server Applications”, 2nd IEEE International Conference on Engineering and Technology (ICETECH), 17th & 18th Coimbatore, TN, India, March 2016.
[29] M. Wiesmann and al., “Understanding Replication in Databases and Distributed Systems”, IEEE Xplore, 2010.
[30] Grace and M. Hoda, “Trust in Real-Time Distributed Database Systems”, IEEE 8th International Conference on Information Technology (ICIT), 2017.
[31] P. Fernando and S. Nicolas, “Byzantine fault-tolerant deferred update replication”, The Brazilian Computer Society, Publisher: Springer, February 2012.
[32] T. Chen et al., “Performance of Database Synchronization Algorithms via Satellite”, IEEE 5th Advenced Satelite Multimedia Systems Conference and the 11th Signal Processing for Space Communications Workshop, 2010.