IJCSI International Journal of Computer Science Issues, Volume 14, Issue 5, September 2017
ISSN (Print): 1694-0814 | ISSN (Online): 1694-0784
www.IJCSI.org https://doi.org/10.20943/01201705.1320

Secure Network using Secure Hash Algorithm-3 with Locking Protocol in Distributed Database System

Prerna Gupta[1], Amit Verma[2] and B.K. Verma[3]

[1] Computer Science & Engineering, Chandigarh Engineering College Landran, Mohali, India

[2] Computer Science & Engineering, Chandigarh Engineering College Landran, Mohali, India

[3] Computer Science & Engineering, Chandigarh Engineering College Landran, Mohali, India

Abstract

The two phase locking protocol provides controlled access to a data item and prevents data inconsistencies in the distributed database system. The hashing algorithm is used to calculate hash value for the requested data item and the generated hash value is used as a medium to secure the communication between server and client connected through a network. This paper presents a methodology which is used to secure network of a distributed database system while processing requests for data items from various users at the same time. The concurrency control issues are resolved using two phase locking protocol and the security issue is resolved using secure hash algorithm-3, 512 variant. By performing extensive experiments, the performance of the proposed method is epitomized. Performance parameters like throughput, accuracy are calculated to compare the proposed methodology with the hash based optimistic concurrency control algorithm.

Keywords: *Database (DB), Distributed Databases (DDB), Distributed Database Management System (DDBMS), Distributed Database System (DDS), Concurrency Control, Hashing, Secure Hash Algorithm-3 (SHA-3), Transaction Manager (TM), Scheduler (SC), Data Manager (DM).*

1. Introduction

A database (DB) is a repository of related information that is useful for some individual or an organization. It integrates the information and provides controlled access to the information in it. A database management system is the system software that is basically a collection of programs to create, store, modify, access and delete data from a database. A distributed database is a collection of logically interrelated data and it is physically distributed over a computerized network. A distributed database management system (DDBMS) consists of a database that is divided into number of fragments which are stored at different computer sites of the network [13].

There are two types of DDBMS: Homogeneous and Heterogeneous. In homogeneous DDBMS, all the computer sites have the same database product installed and the underlying data model is also same. In heterogeneous DDBMS, some sites have different database product installed and the underlying data model is also different.

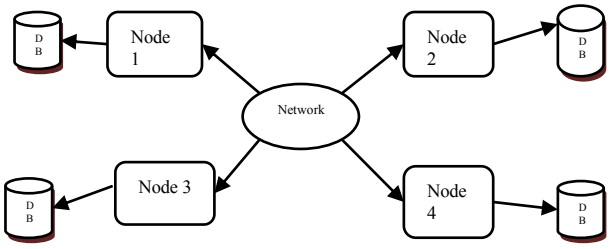

Fig. 1 Distributed Database System.

A multiple client and multiple server DDBMS architecture is used in the proposed methodology. In this, data are stored in a distributed manner at multiple servers and it provides transparency of data access at the server side. For securing the network between clients and servers, a hashing technique is used. Hashing algorithms are cryptographic in nature but they cannot be reversed like other encryption techniques. Hashing is the process of transforming message string into a fixed size value using a mathematical function known as hash or digest. The algorithm breaks the message into fixed size blocks of same size. Hashing function is applied to a block to

produce the first hash code. The main aim of using hashing technique is to maintain data integrity [15]. A hash value is calculated before and after sending data item to the client. Then these values are compared to check if data item is modified during the communication between client and server or not [11].

Concurrency control is the activity of coordinating concurrent accesses to a database by multiple users in a distributed database management system. The main difficulty in this is to prevent updates of one user from interfering with retrievals and updates of another user. Concurrency control is required to guarantee consistency and serializability of data in a database. There are various concurrency control algorithms like timestamp ordering protocol, wound-wait and wait-die, validation based protocol, locking protocol, etc. Two-phase locking protocol which is a type of locking protocol is used in the proposed methodology [11].

2. Architecture of Distributed Database System

A distributed database system (DDS) is an integration of distributed database (DDB) and distributed database management system (DDBMS). A DDBMS consists of two types of sites: query sites and data sites. A query site is utilized to query information and the data sites are associated with the local database. Data is stored at data sites. Each node is assumed to have either a single processor or multiple processors. The processors are connected by a computer network [8]. A DDB is composed of set of databases that are stored at several distinct nodes which are connected through a communication network like LAN. It is managed by a DDBMS.

At each node in the DDS, there is a local DDBMS and a local transaction processing system (TPS). In distributed TPS, there are four main components: a transaction manager (TM), a scheduler (SC), a data manager (DM) and a data repository (database) as shown in fig 2. The TM is responsible for supervising interactions between users and the DDBMS. The scheduler acts as a lock manager and synchronizes transaction operations to avoid inconsistencies. The DM is responsible for managing the database [9].

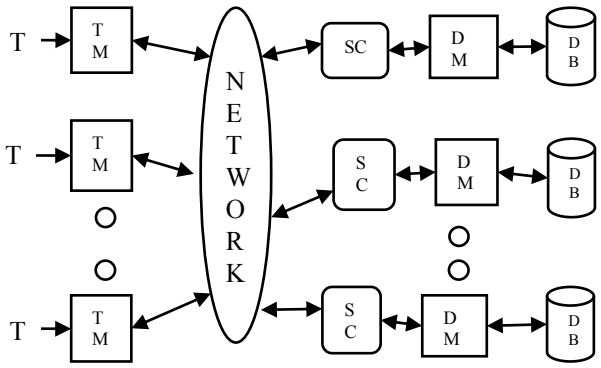

Fig. 2 Transaction Processing Model.

When a user or an application interacts with the DDBMS by executing transactions on a client, which are usually online queries or application programs, TM records that transaction (T). The TM contacts SC. For a read operation, the SC contacts only one of the nodes where data item has been stored [3]. In case of no conflicts, request is sent directly to the DM. The DM provides the access to the database and returns the value of the data item. For a write operation, each node with the replica of the data item is contacted by the SC. In case of no conflict, the data item is provided by the DM [10].

3. Literature Survey

Abbas Qasim et al. 2016 [18] has compared various concurrency control techniques like locking, wound-wait and wait-die and basic timestamp ordering protocol to preserve the ACID property of transactions in distributed database management systems.

Akshay M. Gupta, Yogesh R. Gore, 2016 [1] has reviewed the security issues, data security aspects of client and server. Few concurrency control techniques have been reviewed like timestamp ordering, distributed optimistic protocol and wait-die and wound-wait algorithm.

Sonal Kanungo, Morena Rustom. D, 2015 [22] has discussed about various concurrency control techniques like two-phase locking, timestamp based protocols, validation based protocol and multi version schemes. Optimistic, pessimistic and multi version techniques have been compared.

Dharavath Ramesh, Harshit Gupta et al., 2015 [19] presented a methodology which was based on calculation of a hash value using MD5 hashing algorithm and a locking protocol. It was an optimistic concurrency control algorithm.

Anand Mhatre, Rajashree Shedge, 2014 [2] has compared some distributed concurrency control techniques like distributed speculative locking, validation queue approach and stamp based approach. Parameters like waiting time, validation, performance and delay has been used to compare the techniques.

4. Problem Formulation

The security breaches in a DDS are typically categorized as: Unauthorized data observation, incorrect data modification and data availability [1]. The problem of maintaining data integrity during communication between clients and servers can occur anytime in the DDS. The data could be altered by an intruder in between the communication between server and client. This could hamper the whole transaction process.

A hash based optimistic concurrency control algorithm was introduced which helped to resolve this issue [4]. A hash was calculated using MD5 hashing algorithm and the hash was sent along with the data item. The hash was calculated again for the data and compared with received hash. If the values did not match, data was considered to be altered in between the communication [19].

But the MD5 hashing algorithm is already broken and more prone to hash collisions. Hash collision means that the same hash can be generated for two or more different strings. To solve this issue, a more secure hashing algorithm has been used in the proposed approach. SHA-3 hashing algorithm is more hash collision resistant. It is considered impossible to get same data by reversing the hash digest created for that data [5].

5. Protocols Used

In the proposed methodology, two main protocols have been used: Secure hash algorithm-3, Two-phase locking protocol.

5.1 Secure Hash Algorithm-3

It is prominently known as SHA-3. It is a 5X5 state matrix of 64 bit words. It is based on sponge construction. Data is absorbed in the sponge and the final output is squeezed out. It works in three phases:

1) Initialization Phase: The state matrix, say A, is initialized with all 0's.
2) Absorbing Phase: Each r-bit wide block of message is XOR-ed with the current matrix state and 24 rounds of compression functions are applied on the block.
3) Squeezing Phase: The state matrix is truncated to the desired length of the output.

The size and part of the state matrix that can be read and written is the called as rate "r" and the remaining size and part is known as capacity "c" [6]. The security level is half the capacity and "r+c" determines the width of the SHA-3 functional variation used and its maximum value can be 1600. "r" and "c" are selected depending on the desired output value. SHA-3 has 6 variants which are as follows:

1) SHA-3 224
2) SHA-3 256
3) SHA-3 384
4) SHA-3 512
5) SHAKE 128
6) SHAKE 256

In the proposed methodology, SHA-3 512 variant has been used to calculate the hash of the message string. There is no confinement on the length of the message that can be processed using SHA-3. It has 256 security bits which makes it more secure than other hashing algorithms [7].

5.2 Two-Phase Locking Protocol

A lock is a variable that is associated with a data item which determines if that item could be read or written by the lock requesting user. Two-phase locking protocol is a locking algorithm that guarantees serializability [18].

Table 1: Lock Compatibility Table of 2PL Protocol

Lock Requested by Transaction	***Lock Held by Transaction***	
	Read	**Write**
Read	YES	NO
Write	NO	NO

Table 1 shows compatibility of locks which can be acquired by various transactions at the same time [17]. There are two types of locks that can be acquired by the user on a data item. A shared lock can be acquired on one or more transactions at the same time before reading an item. An exclusive lock can be acquired on an item before

writing and updating the current item by one user at a time [21].

There are two phases locking protocol: expanding and shrinking phase.

1) Expanding Phase: In this phase, all locks are acquired and no locks are released.
2) Shrinking Phase: In this phase, all locks are released and no locks are acquired.

In the proposed methodology, strict 2PL protocol has been used.

6. Proposed Methodology

6.1 Flowchart and Psuedocode of the proposed methodology

The pseudocode of the proposed methodology is as follows:
Input: Database with requested data with concurrent transactions.
Output: Ensures successful modification of the required data
While transaction
If user sends lock_acquire on X to TM
TM contacts SC.
SC checks compatibility with previous locks on X.
If no_conflict
Provide lock_acquire(X).
Else
Wait
If lock_acquire equals to Read
SC contacts DM
Read_lock applied on X.
Else
SC contacts all DMs
Write_lock is applied on all replicas of X.
Calculate hash H1.
Send X, H1.
Client receives X, H1.
Calculates H2 for X
If compare (H1, H2)
Do operation
Calculates H3 for X'
User requests lock_release (X)
Send X', H3
Else
Request lock_acquire on X.
Server side receives X', H2.
Calculate H4.
If compare (H3, H4)
Update X'
Commit
Unlock X'
Else
Abort.
Restore previous consistent state.
End.

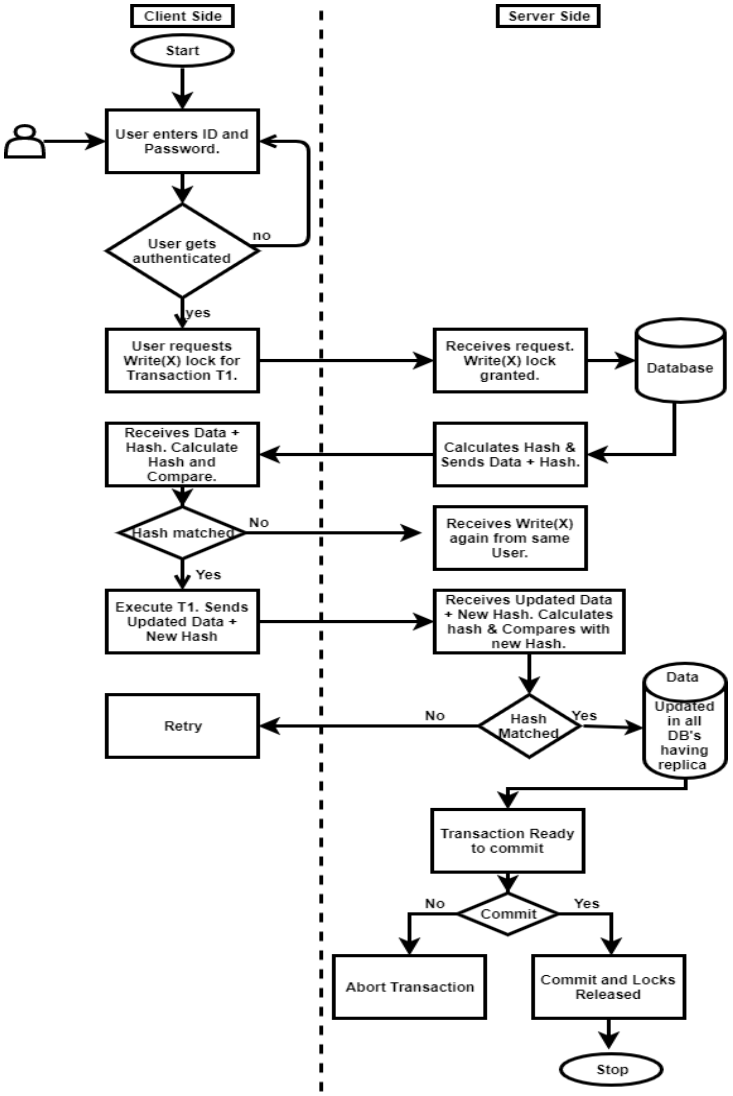

Fig 3. Flowchart of the proposed methodology

6.2 Detailed Steps

The methodology is described as follows:

1.The initial step is to load various processing libraries in the memory.

2.A network is created which consists of various nodes having clients and servers.
3.Users must provide correct Id, password when establishing a connection to prevent unauthorized access to a database. After being authenticated, users start sending requests for write or read locks on a data item (X, A, B) which will be used for their transaction (T1).
4.The request is sent through their clients (Ci…Cn) to the Transaction Manager. After logging the details of T1, the Transaction Manager contacts the Scheduler.
5.The locks are provided by SC after checking the compatibility of the requested locks with locks previously acquired on X, A, B.
6.Suppose, n no. of users are requesting lock on X. For a read operation, the SC contacts one of the local node (S1) and remote nodes (Si…Sn) which also have X. The DM of a node, say Sj, checks its database and provides X. A read lock is applied on X. n no. of users can be granted read lock on X simultaneously.
7.For a write operation, the SC contacts S1 as well as one of the other remote nodes and write lock is applied on all the replicas of X residing on the remote nodes. One of the DMs provides X to one user at a time.
8.The hash value (H1) is calculated for X using SHA-3 hashing algorithm and sent along with X to the client side (C1).
9.The user is provided with X along with H1.
10.The hash value (H2) is calculated again at C1 and compared with H1 using same SHA-3 Algorithm.
11.If H1equals H2, X is used for the operation and if H1 is not equal to H2, request is again.
12.The user which obtains the lock on all the data items(X, A, B) performs its transaction (T1).
13.In case of a write lock, the updated data (X') and the new hash value (H3) is sent to Sj. The hash (H4) is calculated again and compared with H3.
14.If H3 is not equal to H4, it means that X' was modified during communication and if H3 is equal to H4, X is updated with X' and all the replicas of locked X residing over other remote nodes are also updated.
15.If the execution of all the operations of T1 is completed, a release lock request is sent to the nodes which granted X, A, B to C1. T1 is committed and locks are released so that the locks on X, A, B can be acquired by other users for their transactions [16].

7. Results and Discussions

7.1 Simulation Environment

To compare the performance of various methodologies, we modeled a distributed database system. The experiment was conducted on a laptop equipped with 2.13GHz, i3 Intel processor, 3Gbyte RAM and 32-bit operating system. As for the simulation tool, MATLAB 2013a has been used. MATLAB stands for MATRIX LABORATORY. It is a high appearance language for technical computing. It consists of a calculation and programming environment. It is an interactive system. It has debug tools, complex data-structures.

The simulated distributed database system consists of a set of nodes. Each node has a pool of servers, all having identical capabilities and serving one global queue of transactions [17]. The system is assumed to be reliable and scalable in terms of capacity [23].

The read and write operation for first two transactions is entered manually. The number of incoming transactions is manually entered and then the operations for those transactions are generated randomly and performed on a set of data. The number of incoming transactions varies from 1 to 5000. The distributed database contains 1000 data items.

7.2 Performance Results

The performance of SHA-3 based protocol and MD5 based protocol is evaluated by analyzing various performance parameters like throughput, accuracy, security, correctness, number of committed transactions, etc. The cost of performing concurrency control operations is negligible. For each number of transactions, 10 experiments were carried out. The average of the results of the experiments is taken and represented in the graphs. All use cases were taken into consideration to get a final result. For eg. If 10 transactions requested lock on a same data item simultaneously, the transaction to achieve the write lock first was executed. Other 9 transaction were denied for any type of lock on the same data item. If one transaction acquired read lock on a requested data item first, other transactions requesting

read lock on the same data item, obtained the read lock. Other transactions requesting write lock were denied for the lock on the same data item. The throughput, accuracy, no. of committed transactions, calculation time were calculated based on 10 experiments each for all the four cases possible.

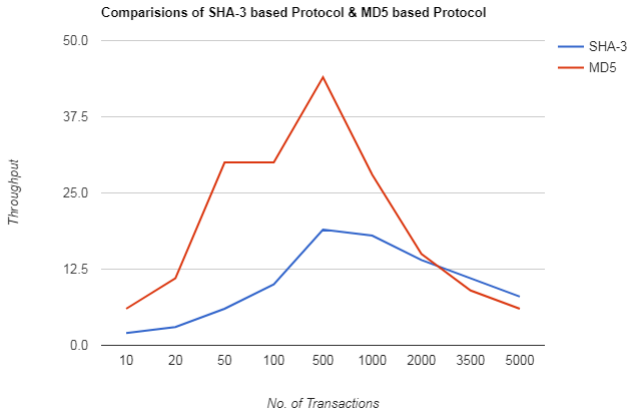

Fig. 4 Throughput versus no. of transactions.

Fig 4 shows how throughput for SHA-3 based protocol and MD5 based protocol vary with number of transactions executed at a time. Throughput is the number of transactions executed per second [24]. It can be observed that the performance of SHA-3 based protocol is significantly higher than MD5 based protocol. For shorter duration transactions, throughput of MD5 was better. But with the increasing numbers of transactions, throughput is higher for SHA-3 based protocol.

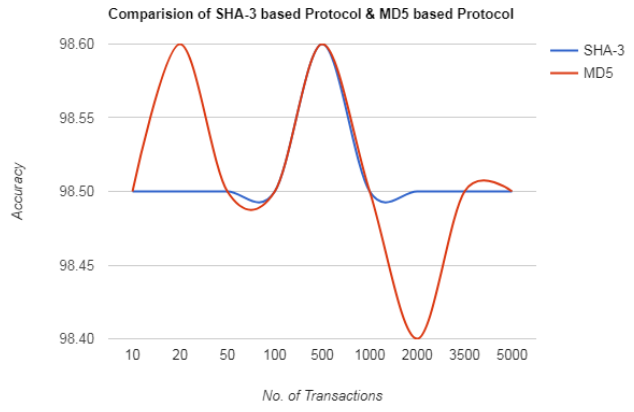

Fig. 5 Accuracy versus no. of transactions.

Fig 5 shows how accuracy performance for SHA-3 based protocol and MD5 based protocol vary with the number of transaction executed at a time. It can be observed that the accuracy for SHA-3 based protocol is slightly higher for MD5 based protocol for longer duration transactions.

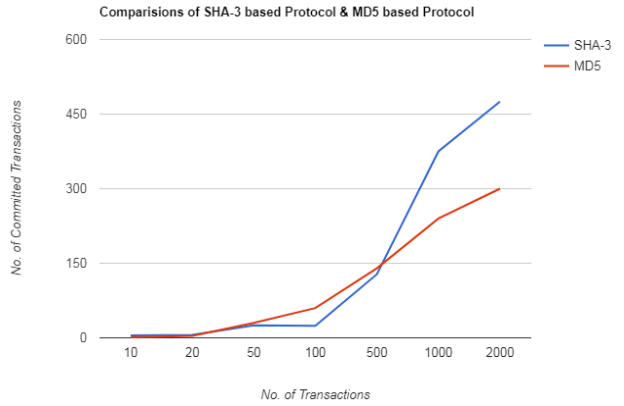

Fig. 6 No. of committed transactions versus total no. of transactions.

Fig 6 shows how the number of committed transactions for SHA-3 based protocol and MD5 based protocol vary with the number of transaction executed at a time. It can be observed that number of committed transactions for SHA-3 based protocol is significantly higher for MD5 based protocol for longer duration transactions.

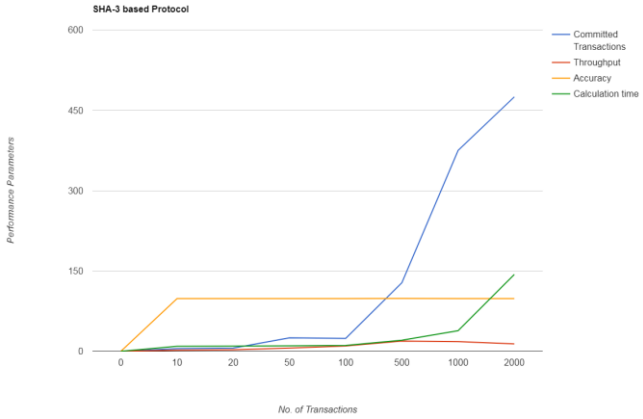

Fig 7 Consolidated Graph

Fig 7 shows the performance of various parameters of the proposed methodology graphically. As the calculation time increases, no. of committed transactions also increases considerably. The throughput is slightly higher when the no. of transactions to request for a data item

increases. The accuracy becomes constant after a certain point of time (as no. transactions increases).

Table 2: Comparison of SHA-3 Based Protocol and MD5 Based Protocol

Parameter	*SHA-Based Protocol*	*MD5 Hash Based Protocol*
Throughput	High	Medium
Correctness	Serializable	Serializable
Accuracy	High	High
Data Currency	High	Low
No. of committed transactions	High	Medium
Memory Usage	Slightly Low	Medium
Security	High	Low
Flexibility of Hashing Algorithm	High	Very Low
Hash Collisions	Very Low	High
Speed	High	Medium
Calculation Time	Medium	High

Table 2 shows various performance parameters which are used to compare SHA-3 based protocol and MD5 based protocol. **Throughput** is significantly higher for SHA-3 based protocol as shown in figure. SHA-3 based protocol does not suffer from **correctness** issues, as two-phase locking used in the protocol is serializable. The **memory usage** of MD5 based protocol is slightly more than that of SHA-3 based protocol as state bits are also stored in case of MD5 based protocol.

The hashing algorithms used in the two protocols i.e. SHA-3 and MD5 provide **security** but MD5 is prone more to **hash collisions** as compared to SHA-3. The security bits in SHA-3, 512 variant are 256 bits whereas MD5 has 64 bits as security bits. SHA-3 is based on sponge construction and output could be squeezed to various lengths depending on the variant of SHA-3 being used. Only 24 rounds of functions are applied on the input, which leads to increase in the **speed** of the SHA-3 based protocol, especially for longer duration transactions. More speed means less **calculation time**. SHA-3 is more **flexible** as it is compatible with XOR functions.

Data currency refers to how up-to-date an object is for a transaction [24]. Two-phase locking provides high data currency to read-only transactions by satisfying serializability criteria. The two-phase locking is widely used in commercial DDBMS to synchronize data access [14]. Optimistic concurrency control protocols are best when there are Read Only transactions but if conflicts start occurring, a lot of rollbacks are generated [20]. If there are several interferences, many transactions that execute to completion will have their results discarded and must be restarted later [22].

8. Conclusions

In conclusion, the novelty of the proposed methodology is shown in taking hash value of the stored data. This ensures that data integrity of the data item is maintained during communication in the network. The two-phase locking is used to prevent concurrency control problems. To ensure that the data item is securely transferred over the network, SHA-3 algorithm is used. SHA-3 algorithm is more hash collision resistant. The performance is improved for longer transactions as shown in the figures. The SHA-3 based protocol is compared with MD5 using various parameters like throughput, accuracy, no. of committed transactions, etc.
In future, work will be done to improve the overhead which occurs due to calculation of hash value in the existing as well as proposed methodology.

Acknowledgments

The authors would like to express their gratitude and heartiest thanks to Department of Computer Science and Engineering, Landran for providing outstanding support.

References

[1] A. M. Gupta, Y. R. Gore, “Concurrency Control and Security Issue in Distributed Database System”, International Journal of Engineering Development and Research, 2016, vol. 4, pp. 177-181.
[2] A. Mhatre, R. Shedge, “Comparative Study of Concurrency Control Techniques in Distributed Databases”, in 4th IEEE International Conference on Communication Systems and network Technologies, 2014, pp. 378-382.
[3] B. K. Verma, “Design Issues for a Data warehousing from Enterprise Resource Planning (ERP) Systems”, in 4th IEEE ICCSIT, 2011.
[4] B. Thuraisingham, H. H. Rubinovitiz, “Multilevel Security Issues in Distributed Database Systems-III”, Computers and Security, 1992, vol.11, no. 7, pp. 661-674.
[5] B.K. Verma, “Context Aware Social Popularity Based Recommender System,” International Journal of Computer Applications, 2014, vol. 92.
[6] B.K. Verma, “Social Popularity Based SVD++ Recommender Systems,” International Journal of Computer Applications, 2014, vol. 87.
[7] C. H. Romine, “SHA-3 Standard: Permutation-Based Hash Extendable Output Functions”, Federal Information Processing Standards Publication, 2015, pp. 1-29.

[8] G. Bertoni, J. Daemen, M. Peeters, G. V. Assche, "The Keccak SHA-3 Submission Version 3", 2011, pp. 1-14.
[9] J. Conan, W. Constant, S. Pierre, "A Comparative study of some concurrency control algorithms for cluster-based communication networks", Computers and Electrical Engineering, Elsevier, 2005.
[10] J. Sheetlani, V.K. Gupta, "Concurrency Issues of Distributed Advance Transaction Process", International Science Congress Association, 2012, pp.426-429.
[11] K. Solaiman, G. Morgan, "Later Validation/Earlier Write: Concurrency Control for Resource Constrained Systems with Real-Time Properties", in 30th IEEE Symposium on Reliable Distributed Systems Workshops, 2011, pp.9-12.
[12] L. Nguyen, "An Advanced Approach of Local Counter Synchronization to Timestamp Ordering Algorithm in Distributed Concurrency Control", Open Access Library Journal, 2015.
[13] M. Kaur, H. Kaur, "Concurrency Control in Distributed Database Systems", International Journal of Advanced Research in Computer Science and Software Engineering, 2013, vol.3, no. 7, pp. 1443-1447.
[14]M. Goyal, T. Ragunanthan, P. K. Reddy, "Extending Speculation-Based Protocols for Processing Read-Only Transactions in Distributed Database Systems", in 12th IEEE International Conference on high Performance Computing and Communications, 2010, pp. 527-532.
[15] P. Gupta, S. Kumar, "A Comparative Analysis of SHA and MD5 Algorithm", International Journal of Computer Science and Information Technologies, 2014, vol. 5.
[16] P. Gupta, B.K. Verma, "SHA-3 & Locking Protocol for Distributed Database Systems", Press, 2017.
[17] P. Krishna Reddy, Masaru Kitsuregawa, "Speculative Locking Protocols to Improve Performance for Distributed Database Systems", IEEE transactions on Knowledge and Data Engineering, 2004, vol.16, no. 2.
[18] Q. Abbas, H. Shafiq, I. Ahmad, S. Tharanidharan, "Concurrency Control in Distributed Database System," in IEEE International Conference on Computer Communication and Informatics, 2016.
[19] R. Dharavath, H. Gupta, K. Singh, C. Kumar, "Hash Based Incremental Optimistic Concurrency Control Algorithm in Distributed Databases", Advances in Intelligent Systems and Computing, Springer, 2015, pp. 139-149.
[20] R. A. Obaidah, M. A. Hiba, A. A. Nedhal, "An Optimistic Approach in Distributed Database Concurrency Control", in 5th International Conference on Computer Science and Information Technology, 2013, pp. 71-75.
[21] S. Kaspi, S. Venkatraman, "Performance Analysis of Concurrency Control Mechanisms for OLTP Databases", International Journal of Information and Education Technology, 2014, vol. 4, no. 4.
[22] S. Kanungo, D. R. Morena, "Analysis And Comparison Of Concurrency Control Techniques", International Journal of Advanced Research in Computer and Communication Engineering, 2015, vol. 4, pp.245-251.
[23] T. Ragunanthan, P. K. Reddy, "Improving the Performance of Read-Only Transactions through Speculation", Springer-Verlag Berlin Heidelberg, 2007, pp.203-221.
[24] T. Ragunanthan, P. K. Reddy, "Exploiting Semantics And Speculation for Improving the Performance of Read-Only Transactions", in International Conference on Management of Data COMAD, 2008.