Kafrelsheikh Journal of Information Sciences
ISSN (Online): 2535-1478, ISSN (Print): 2537-0677
Volume 3, Issue 1, 2022, PP. .

A Review on Concurrency Control Techniques in Database Management Systems

Mahmoud Y. Shams[a], Ahmed S. Abolaban[b], Amr A. Abohany[b]

[a]Faculty of Artificial Intelligence, Kafrelsheikh University, Kafrelsheikh, 33516, Egypt.
Faculty of Computers and Artificial Intelligence, Helwan University, Helwan, 11795, Egypt
[c]Faculty of Computers and Information, Kafrelsheikh University, Kafrelsheikh, Egypt, 33516, Egypt.

[*]Corresponding Author: Mahmoud Y. Shams, E-mail: mahmoud.yasin@ai.kfs.edu.eg

Abstract: Conflicts, deadlock and rolled-back transactions are being considered as the most recent challenges related to executing the transaction concurrently on different environments of Database Management Systems (DBMS). More precisely, in distributed database systems, to handle and avoid these challenges, there are different techniques and protocols are utilized. In this paper, we highlight some of these techniques which includes Two-Phase Commit (2PC) protocol and Three-Phase Commit (3PC) protocol) as well as and Deadlock-Free Cell lock (DFCL) algorithm. Moreover, the paper surveys all these protocols and demonstrate the pros and cons of each techniques. Afterwards, we proposed the solution of some important problems related to concurrency control techniques in DBMS

Keywords: Distributed database; Concurrency control; 2PC; 3PC; Deadlock; 2phase locking.

1. Introduction

There are various database management systems, including centralized and distributed database management systems. In Distributed database systems (DDBS), the data is distributed and replicated over multiple nodes. In DDBMS there are many types of fragmentation for duplicating data, these types are horizontal, vertical and hyper fragmentation; in contrast to the centralized data base system (CDBS), just one copy of the data is stored in centralized location or node [3,6]. In the database management systems Concurrency control is one of the most critical issues; execution of multiple transactions at the same time is called concurrency [2,5]. Managing the concurrent transaction badly leads to some problems; the rule of concurrency control is to ensure that the consistency and integrity constraints will not be violated. [3] The deadlock problem and conflict problem are some of the transactions which executed concurrently, Simultaneous transaction execution, if not properly handled, will lead to a conflict. The system will roll back the conflicted transaction to give the others to be completed. The infinite waiting of transactions for data lock to be executed is a problem called deadlock in which each transaction is waiting the other one to release its locked data item to use it [2]. The good management for execution of the concurrent transactions could lead to some advantages and enhancing the performance of the database management system these advantages like increasing the amount of throughput and reduction of waiting-time for the transactions [4, 9, 10].

The amount of Throughput and response time for transactions in the database management system are two performance factors that the concurrency control algorithms depend on. Transaction blocking, transaction restarts, local processing and the site communication are the four factors which impact the DBMS performance [3, 14].

In the environment of DDBMS the 2PC algorithm is very useful to manage the concurrent transactions [14]. The two-phase-commit is One of the most popular protocols used in the commitment and concurrency control In DDBMS [7]. For transactions consistently termination, it is necessary to adapt an atomic commitment protocol in the system. The Two phases-commit protocol considered as one of the atomic

protocols which needed to terminate global transactions consistently in the system [8]. In 2PC protocol, the nodes which have more queries considered as a primary site and the others which have fewer queries considered as secondary sites; voting phase and commit phase are two phases in 2PC [1].

3PC was established to solve the blocking issue with the two-phase commit protocol [19]. Compared to 2PC, 3PC has an additional phase, called the pre-commit phase, this process makes the protocol non-blocking, but it comes at the cost of higher message transmission costs, and since 3PC solves this problem, 2PC has its own set of benefits [18].

Deadlock is a significant problem that must be resolved when several transactions are being processed at the same time. When two or more transactions need resources owned by other transactions to complete their execution on a regular basis, deadlock occurs [2].

2PL protocol is locking-based technique, which considered as a pessimistic concurrency control technique used that executes transactions in two phases: growing and shrinking [4, 12].

2 Current State of Art.

A Backup coordinator was added to a new framework for 2PC in [14] (2012), which considerably reduces transaction blocking. The backup coordinator, however, may be prevented in the worst-case scenario. The first of the two proposed protocols in [15] (2012) is a different strategy from the pessimistic concurrency control. The second algorithm is selective contention analysis (SCA), which enables systems that implement the very light locking (VLL) algorithm to achieve a higher amount of throughput under a higher workload. The very light locking (VLL) algorithm is designed for main-memory database systems and eliminates the overhead associated with lock manager traditional operations. Adding a backup phase to the 2PC algorithm stages is part of the Backup Commit (BC) protocol, which was introduced in [13] (2016).The concept of this protocol based on attaching one backup site coordinator site. Once all nodes have reacted in the first phase of the backup procedure, the coordinator just sends his decision to the backup site. Blocking happens if the coordination site is unavailable; in this case, the subordinator sites adopt the termination procedures of the backup site. The non-blocking quality may be attained by using BC protocol in the vast majority of coordinator site failures. TicToc is a brand-new optimistic concurrency algorithm introduced in [16] (2016) that does away with concurrency and scalability issues. TicToc assigns “write” and “read” timestamps to data items to determine availed commit timestamp for each transaction; it eliminates the need for central timestamp allocation and commits transactions that would otherwise be discarded by traditional T/O schemes

In [17] (2017) an approach for solving conflicts in the multi-version concurrency control [MVCC], this technique. In the form of dependency this the algorithm summarizes the transaction programs, after that When our mechanism detects a conflict between transactions, it easily locates the conflict in the system and partially re-executes the conflicting transactions. This method increases transaction processing throughput by increasing the reuse of computations performed in the initial execution round.

We can suggest that for a 2PL improvement deadlock free cell locking (DFCL) can be applied to accomplish which concurrent transaction. Moreover the DFCL boosts the enrolled database by improving committed transfers, response time, throughput, concurrency, and concurrency related to the structured query language of the DBMS.

3 A Literature Review On Some Currency Techniques

In papers [1] and [2] different concurrency control techniques are proposed, which used to deal with the problems with concurrent transactions in DBMSs. 2PC, 3PC and the DFCL are the techniques that I will discuss. They have been explained as follows:

3.1 Two phases-commit protocol(2PC).

The 2PC is an algorithm which is classified as a distributed algorithm which can be used in distributed database systems and computer networks. The idea behind this strategy is that a master node serves as the coordinator, while the other nodes, which are slave nodes, serve as subordinates. The 2PC protocol contains two stages, which are as follows: All nodes vote yes or no on whether the coordinator node should serve as the commitment node during the first phase, known as the preparation phase. This phase includes the following processes:

- The coordinator sends a message to all the subordinators asking them to cast a vote on whether to commit the transaction or cancel it.
- Subordinator generates Vote-commit and sends it to the coordinator if they can commit the transaction; otherwise, they send vote-abort to the coordinator.

The transactions will be completed concurrently during the second phase, known as the decision-making phase, upon receiving "commits" or "aborts" signals from the coordinator, depending on the related criteria. The following will take place during the second process:

- The coordinator sends out a "commit" message if all the subordinators can complete the transaction.
- The coordinator sends out a "abort" message if at least one subordinate node is unable to complete the necessary transaction.
- If all participants received the vote-commit message, the coordinator will send a global "commit" message to all participating nodes.
- In the event that at least one vote-abort is received, the coordinator will send a worldwide "abort" message to everyone who voted for a commitment.

As shown in Figure 1 and Figure 2 and Figure 3 the two phases commit processes are illustrated.

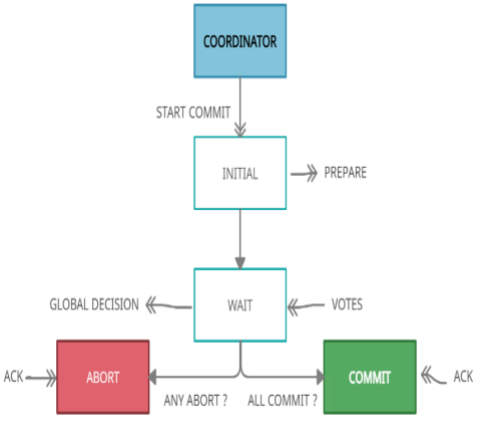

Figure 1. Coordinator processes in 2PC

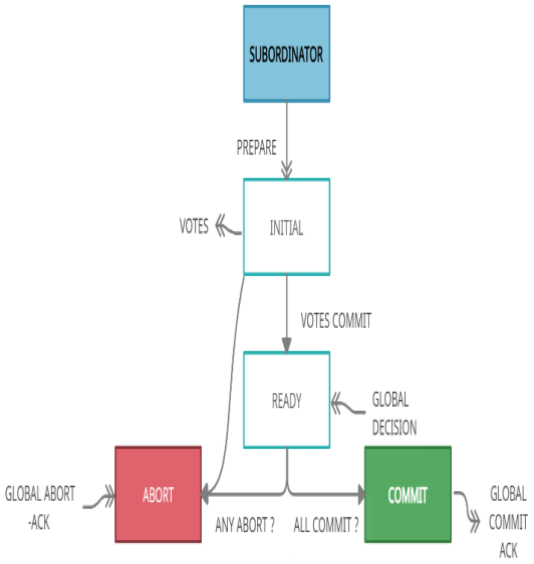

Figure 2. Subordinator processes in 2PC

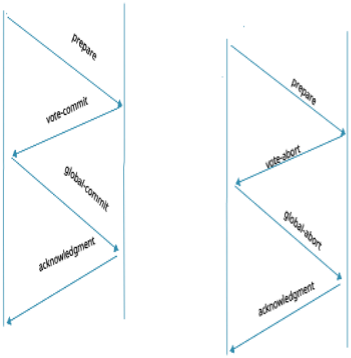

Figure 3. Two phase commit processes

The two-phase commitment approach is actually costly since it necessitates the simultaneous writing of essential information to secured storage and the exchange of several messages between metadata servers. The system performance is affected because 2PC employs locking mechanisms to safeguard the resource, which necessitates serialising several operations on the same directory. Two–phase commit technique can handle network failures and guarantee atomicity, the 2PC protocol suffers from the following:

- In case of failure of the coordinator participant site will be blocked.
- Compared to a simple optimistic protocol, the latency is increased due to communication overhead writing logs forcedly.

- Database availability is challenged by the blocking nature of the 2PC protocol, making its implementation through distributed database management systems unsuitable.
- The 2PC approach has been shown to be blocking in the case of numerous failures.

3.2 Three phases-commit protocol (3PC).

Due to the blocking nature of the 2PC between the READY and COMMIT states, the 3PC protocol introduces a new PRE-COMMIT state; the pre-commit state makes the 3PC non-blocking and ensures that no transaction is executed directly between committable states and the non-committable states in the event of node failure. The three phases of 3PC are as follows:

3.2.1 Voting phase

- The other phases commit and decision are prepared
- The node where the transaction starts become the coordinator, and it then asks the other peers to vote on whether to commit or abort the transaction.
- Cohorts vote on whether or not to commit transactions, and the coordinator judges whether or not to commit the transactions depending on the results of voting. Otherwise, even though one of the cohorts is opposed to the transaction, it decides to abort.

3.2.2 Commit phase preparation:

- The coordinator informs all cohorts of their decision.
- All cohorts receive a message “enter into ready to commit stage”, If the decision is to committing the transaction

3.2.3 The decision phase

- If the coordinator decides to commit the transaction, it will send global-commit to all cohorts and wait for them to acknowledge the transaction. It continues to commit the transaction after receiving their acknowledgment.
- If the coordinator decides to abort the transaction, global-abort will be sent to all sites and the transaction will be aborted. After receiving the acknowledgements, the final decision is taken.
- If every participant submits a vote-commit, the 3PC does not act right away; instead, the coordinator sends out a message instructing everyone to be ready to commit, at which point everyone enters the pre-commit stage and notifies the coordinator. After obtaining acknowledgment from all parties, the coordinator transmits "commit," and the participants commit the transaction.

As shown in Figure 4 and Figure 5 the three phases commit processes are illustrated.

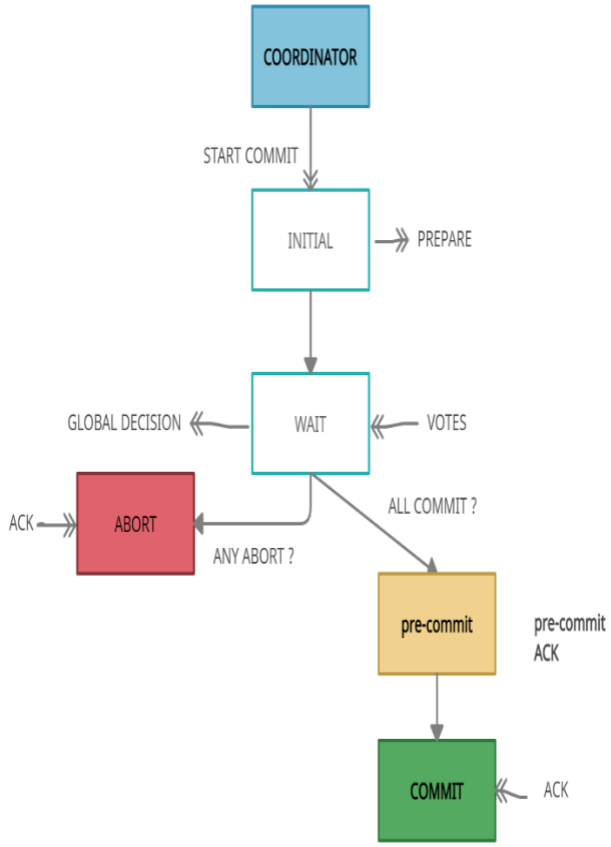

Figure 4. Coordinator processes in 3PC

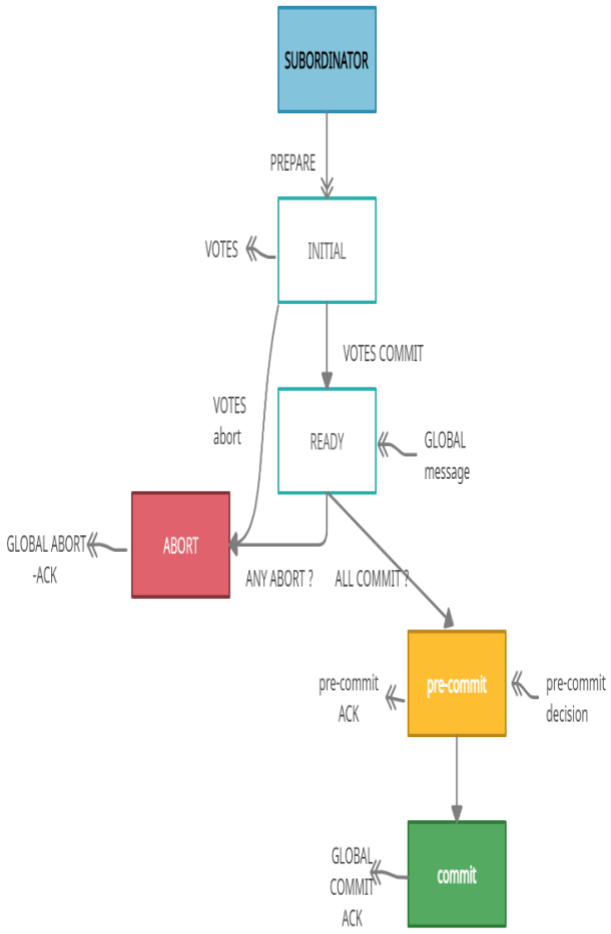

Figure 5. Subordinator processes in 3PC

Comments and discussion on "2PC and 3PC" algorithms. When we look at the first one we can identify that the 2PC has two main problems, blocking and state inconsistency, the first problem blocking state problems, this problem is proven for the 2PC technique which occurred when the coordinator goes to failure state when the slaves nodes Is in an uncertain state, the second problem is with the 2PC is inconsistency which happened when the "commit" and "abort" states are both present in the global state vector, on the other hand; The 3PC protocol is useful only when there are failures in multiple nodes. There are some different between the two protocols for examples:

2PC has the Medium average of latency but the 3Pc has High latency, and about the communication overhead, the complexity and cost average the 2PC is less than the 3PC, the 2Pc surpassed the 3PC in the performance.

DFCL is an enhanced two-phase locking technique that minimizes the conflicts and tries to achieve deadlock-free locking this technique is called the "deadlock-free cell lock". The approach seeks to lock data at the least restrictive level possible. By requiring the pending transaction to move into the commit phase or the rollback, it also seeks to address the deadlock problems caused by various locking mechanisms. It boosts the volume of committed transactions, which enhances database performance. Protocol DFCL based on adding a low degree of locking to the system. The following are some DFCL concepts:Depends on that locking degree of the system is low.

- It enables concurrent transactions to use the same data simultaneously reducing the conflicts to the minimum number of conflicts.
- The technique makes each concurrent transaction to conduct its operations on the cells it wants, and make the other transactions have the ability to use the rest of the data.
- The lock of the small parts is divided into different locks with different transactions instead of one, and the record lock is divided into multiple pieces of locks that can be spread out over many transactions as a result of dividing the database into small parts and partitioning its record in too many cells.
- The pending transaction with a number of performed statements is given priority execution in the DFCL trigger.

The idea behind executing read and write operations in DFCL is as follows:

- The write transaction employs the exclusive lock (write lock mode) to prevent any other similar read or write transactions from accessing the same cell. In DFCL, read transactions never reject each other's.
- In the DFCL, a cell lock should only be obtained by one transaction and kept in place until that transaction has finished working on the cell.
- Read transactions lock the cell they need in share lock mode (read locks), allowing other transactions to simultaneously access the cell for reading or writing operations.The DFCL processes for execution of (insert, update, delete and read) operations are shown as following in

Figure6, Figure 7, Figure 8, Figure 9 respectively:

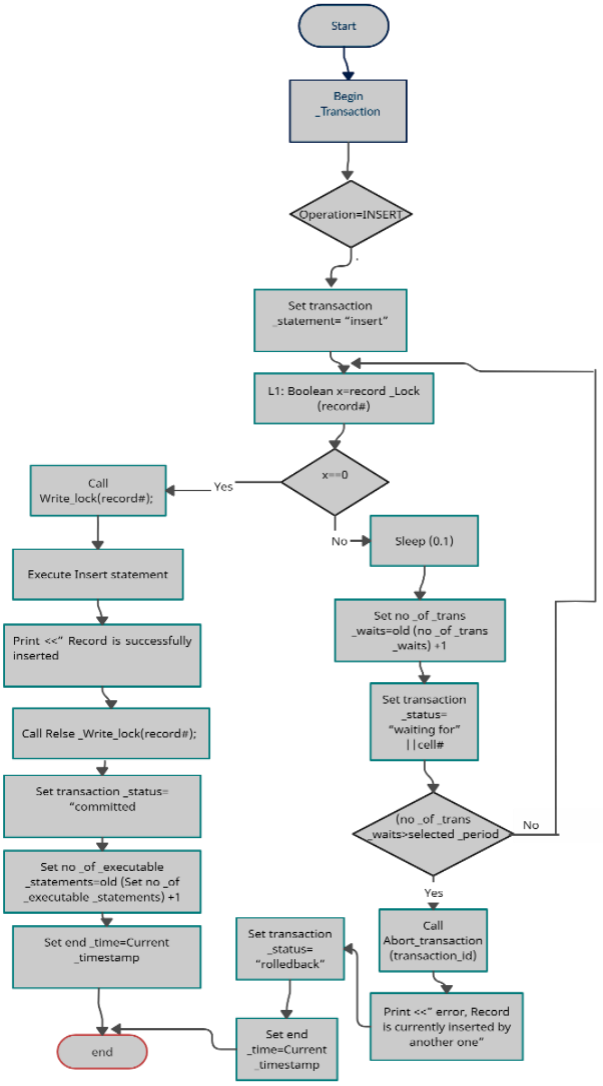

Figure 6. Inserts Process in DFCL

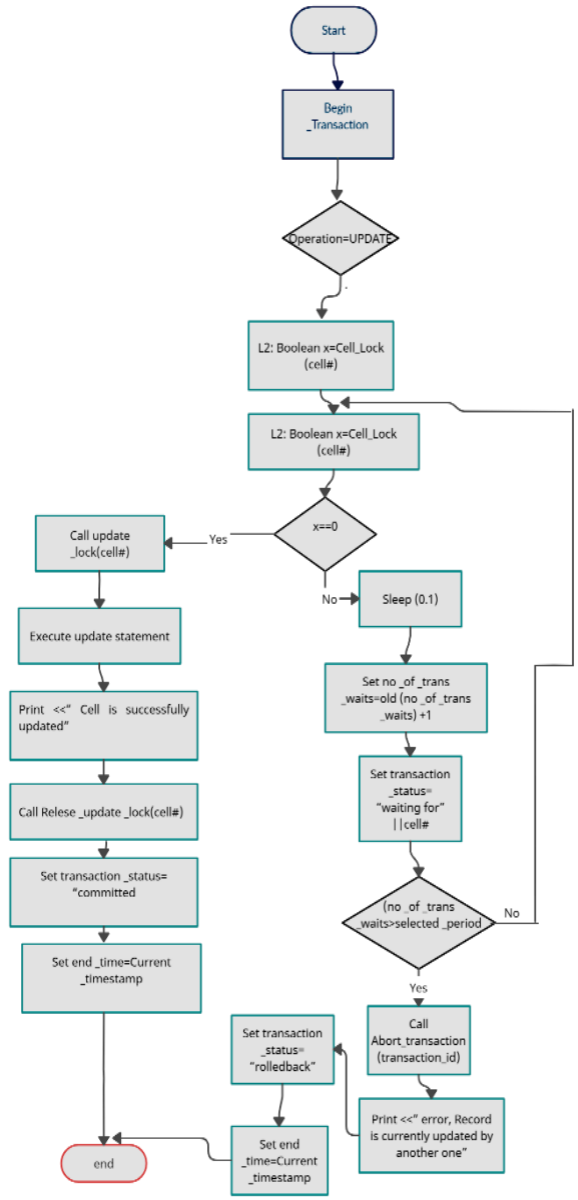

Figure 7. Update Process in DFCL

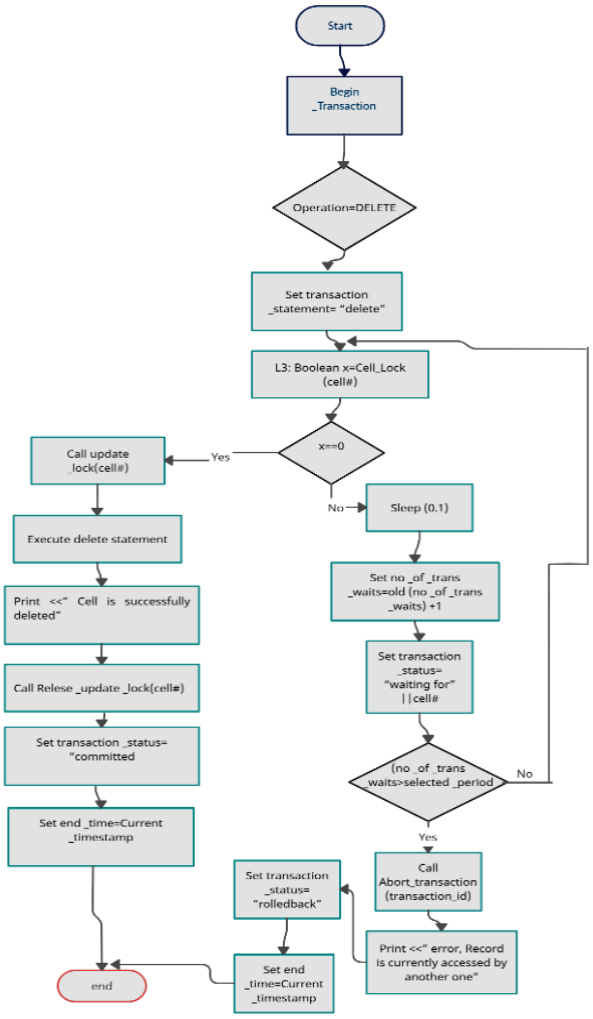

Figure 8. Delete Process in DFCL

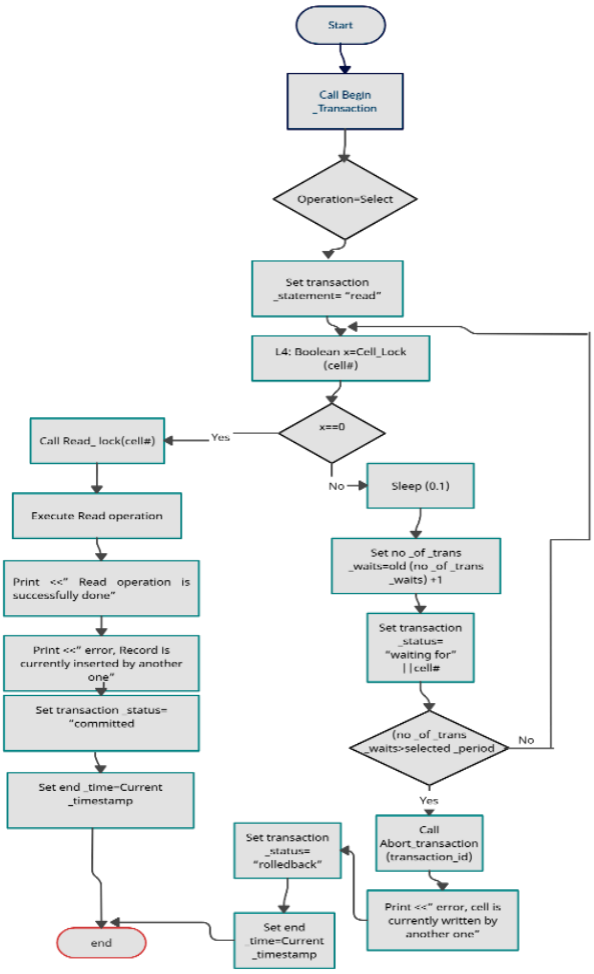

Figure 9. Read Process in DFCL

4 The experimental test of the "DFCL" algorithm

They adapt the APACHE MySQL database management system and create some tables then they use SQL statements to insert some data into multiple records. In each run in this test, there are the following transaction parameters. These parameters includes the number of both read and write transations as investigated in Table 1.The concurrency of control algorithm evaluation metrices are demonstrated in Table 2. By which they evaluate the technique according to the following metrics.

Table 1. The transaction in each run for both read and write transactions.

No of transactions in each run	No of "Read" transactions	No of "Write" transactions
1500	300 (20%)	1200 (80%)

Table 2. The evaluation metrics of the concurrence control algorithms.

	Concurrency control algorithms' evaluation metrics
1	Committed transactions number
2	No. rolled-back transactions
3	No. waiting transactions
4	Throughput = No. executed transactions / No. execution.
5	The Concurrency = (Throughput/latency)
6	Increased throughput and concurrency improve database performance
7	Reducing the waiting time reduces the complexity of the system

The results of the DFCL method are compared to those of other concurrency control techniques in Table 2. According to the average number of (committed, rolled-back, waiting transactions, and deadlock) as indicated in Table 3,

4.1 Comments and discussion on "DFCL" algorithm.

As shown in the table, the experimental test results of the proposed algorithm DFCL compared with other algorithms of concurrency control we can determine that the total number of committed transactions is higher when cell locking is used with the DFCL algorithm than when other concurrency control algorithms are used. DFCL has a lower conflict rate, which decreases the need for transaction abortions while also decreasing the total number of rollbacks.

Table 3. The comparative between different types of concurrency control DBMS.

	Transactions in each run	Number of Runs	Average Number of Committed Transactions	Average Number of the Waiting Transactions	Deadlock free	Execution Time (s)	Throughput Transactions
DFCL	1500	10	1234	0	Yes	3	411
2PL	1500	10	290	800	No	3	100
Timestamp based Concurrency	1500	10	432	0	Yes	3	144
Partition Locking	1500	10	1009	150	No	3	336
MVCC	1500	10	999	0	Yes	3	333
OCC	1500	10	499	0	No	3	166

DFCL has the largest number of transactions completed in the shortest amount of time which means that the number of Throughput is the largest one comparing with the other concurrency control algorithms are shown in the Table 3.

The suggested approach has the lowest average number of rolled back transactions and the lowest average number of waiting transactions, both of which contribute to the system's improved performance by preventing deadlocks and enhancing responsiveness.

One of the drawbacks of the DFLC algorithm is that a lot of storage space is required to store each transaction and each individual cell with its state. We can conclude that the DFCL technique increases space complexity and reduces time complexity.

To solve this problem, the forthcoming works attempts and focus on managing concurrency algorithms using high overhead level of certainty by determining th the timestamp, multi-version, and/or positive performance measures of the applied database.

5 Conclusions

In this study, we discussed concurrency control in the context of distributed database management systems, as well as several approaches to dealing with issues like deadlock and conflicts that arise while running transactions simultaneously. The methods have covered include the two-phase commit protocol, the three-phase commit protocol, and the deadlock free cell lock. I have covered each method's operation as well as some of its benefits and drawbacks. Finally, we can state that there are several sorts of concurrency control protocols, each of which has a particular work style and In certain ways, each variety is superior than the others.

References

[1] R. Nyasuguta Arika and W. Cheruiyot, "A survey on efficient concurrency control algorithm in distributed database systems," *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, pp. 1176–1185, 2019.

[2] M. Mohamed, M. Badawy, and A. EL-Sayed, "An improved algorithm for database concurrency control," *Int. J. Inf. Technol.*, vol. 11, no. 1, pp. 21–30, 2019

[3] P. B. Gbaranwi and P. O. Asagba, "Distributed transactions and distributed concurrency control," International Journal of Computer Science and Mobile Computing, vol. 10, no. 1, pp. 61–68, 2021.

[4] S. Pandey and U. Shanker, "RACE: A concurrency control protocol for time-constrained transactions," *Arab. J. Sci. Eng.*, vol. 45, no. 12, pp. 10131–10146, 2020.

[5] S. Kanungo, R. D. Morena, "Issues with Concurrency Control Techniques," *International Journal of Engineering and Applied Computer Science*, vol. 02, no. 08, pp. 259–263, 2017.

[6] G. Alkhatib and R. S. Labban, "Transaction management in distributed database systems: The case of Oracle's two-phase commit," *J. Inf. Syst. Educ.*, vol. 13, no. 2, pp. 95–104, 2002.

[7] S. Gupta and M. Sadoghi, "EasyCommit: A Non-blocking Two-phase Commit Protocol," 2018, pp. 157–168.

[8] M. L. Liu, D. Agrawal, and A. El Abbadi, "The performance of two-phase commit protocols in the presence of site failures," *Distributed and Parallel Databases*, vol. 6, no. 2, pp. 157–182, 1998.

[9] K. Ramamritham, S. H. Son, and L. C. DiPippo, "Real-time databases and data services," *Real-Time* Syst., vol. 28, no. 2/3, pp. 179–215, 2004.

[10] B. Kao and H. Garcia-Molina, "An overview of real-time database systems," Real Time Comput, vol. 127, pp. 261–282, 1993

[11] Qin, B.; Liu, Y.; Yang, J.: A commit strategy for distributed realtime transaction. J. Comput. Sci. Technol. 18(5), 626–631 (2003)

[12] Qin, B.; Liu, Y.; Yang, J.: A commit strategy for distributed realtime transaction. J. Comput. Sci. Technol. 18(5), 626–631 (2003)

[13] P. Krishna and K. Masaru, Reducing the Blocking in Two-phase Commit Protocol Employing Backup Sites. Institute of Industrial Science, The University of Tokyo, 2016.

[14] V. Manikandan, R. Ravichandran, R. Suresh, and S. F, "An Efficient Non-Blocking Two Phase Commit Protocol for Distributed Transactions," International Journal of Modern Engineering Research, vol. 2, no. ue. 3, pp. 788– 791, 2015.

[15] K. Ren, A. Thomson, and D. J. Abadi, "Lightweight locking for main memory database systems," Proceedings VLDB Endowment, vol. 6, no. 2, pp. 145–156, 2012.

[16] X. Yu, A. Pavlo, D. Sanchez, and S. Devadas, "Tictoc: Time traveling optimistic concurrency control," 2016, pp. 1629–1642.

[17] M. Dashti, S. Basil John, A. Shaikhha, and C. Koch, "Transaction repair for multi-version concurrency control," in Proceedings of the 2017

ACM International Conference on Management of Data, 2017.

[18] K. Tabassum, F. Taranum, and A. Damodaram, "A simulation of performance of commit protocols in distributed environment," in Advances in Parallel Distributed Computing, Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 665–681.

[19] P. Singh, P. Yadav, A. Shukla, and S. Lohia, "An extended three phase commit protocol for concurrency control in distributed systems," Int. J. Comput. Appl., vol. 21, no. 10, pp. 35–39, 2011.