International Journal for Multidisciplinary Research (IJFMR)
E-ISSN: 2582-2160 ● Website: www.ijfmr.com ● Email: editor@ijfmr.com

Exploring Concurrency Control Techniques in Multi-User Database Systems: A Comprehensive Review

Rupali Banerjee

Department of Computer Science and Engineering, Future Institute of Engineering and Management, Sonarpur, Kol-150, West Benga

ABSTRACT

Multi-user database systems require concurrency control techniques in order to maintain data consistency and integrity while handling concurrent transactions. Four important concurrency control techniques are examined in this paper: timestamp-based, validation-based, lock-based (including the popular Two-Phase Locking (2PL) protocols). Lock-based protocols use locks to control access to shared resources; on the other hand, two-phase logic (2PL) guarantees transaction serialization. Timestamp-based protocols provide a consistent execution order by scheduling and ordering transactions according to distinct timestamps.Optimistic concurrency control, or validation-based protocols, validate transactions after they are executed in order to reduce conflicts without first obtaining locks. By means of an extensive investigation of various protocols, the principles, methods, benefits, and drawbacks are clarified, assisting in the well-informed choice and application of concurrency control techniques that are customized to meet particular database system needs.

Introduction:

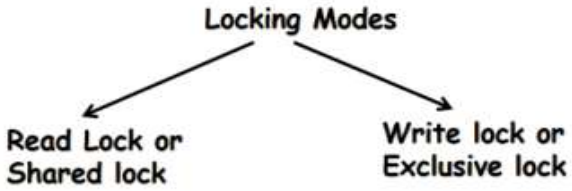
A key component of database management systems (DBMS) is concurrency control, which is essential to guaranteeing the concurrent execution of transactions while maintaining data consistency and integrity. Lock-based, time stamp-based, and validation-based protocols—which control access to shared data in DBMS environments—are crucial to this effort. These protocols control access to data objects by using write locks (exclusive locks) and read locks (shared locks), as shown in the lock compatibility matrix. Strict Two-Phase Locking (Strict-2PL), Conservative 2PL, and Distributed 2PL for DDBMS are some of the varieties of lock-based methods that play crucial roles in ensuring transaction serializability. Two-Phase Locking (2PL) protocols, which consist of growing and shrinking phases, are also important.Transaction timestamps are used by time stamp-based protocols, such as Conservative Timestamp Ordering and the Basic Time Ordering (TO) protocol, to effectively sequence transactions and resolve conflicts. Simultaneously, protocols that rely on validation, such as Sagas and Optimistic Replication, take an optimistic stance, permitting transactions to move forward without acquiring a lock. Resolving conflicts arises during the validation stage. This introduction provides an in-depth analysis of concurrency control mechanisms in database management systems (DBMS), elucidating fundamental concepts and recent developments in the field. It draws on seminal works by Agrawal and El Abbadi [1], Kung and Robinson [2], Papadimitriou and Yannakakis [3], Lynch et al. [4], Saito and Shapiro [5], and Garcia-Molina and Salem [6].

Concurrency Control Protocols
Lock -Based Protocols
Two phase locking protocols(2PL)
Time stamp -Based Protocols
Validation -Based Protocols

1. Locking Based Concurrency Protocol

- To attain uniformity ,the most crucial concept is isolation, and the simplest way to achieve isolation is through locking, which involves first obtaining a lock on a data item, performing a desired action on it, and then unlocking it.
- Locking mechanisms are a commonly employed approach for achieving synchronization.
- A data variable that is linked to a data item is called a lock.
- A lock ensures that a data item is only used for the current transaction.
- The transaction must lock the data item before it may be accessed.

To provide better concurrency along with isolation we use different modes of locks



1. Read Lock or Shared lock(S):

Assigned with lock-S(Q).when a transaction requests to read rather than update a data item.With the shared lock, the data item can be read in between transactions.Another name for it is a read-only lock.The same lock on the same data item may be obtained concurrently by any other transaction (referred to as shared).

2. Write or Exclusive Lock (X):

Assigned with lock-X(Q).Compatibility verification is done before the read lock is upgraded to the write lock.(Write and read both).A data object with the exclusive lock can be written to and read from.Lock-X instruction is used to request X-lock.Neither Write nor Exclusive mode lock can be gained by any other transaction.

Lock Compatibility Matrix –

	S	X
S	✓	✗
X	✗	✗

A transaction may be granted a lock on an item if the requested lock is compatible with locks already held on the item by other transactions.

Any number of transactions can hold shared locks on an item, but if any transaction holds an exclusive(X) on the item no other transaction may hold any lock on the item.

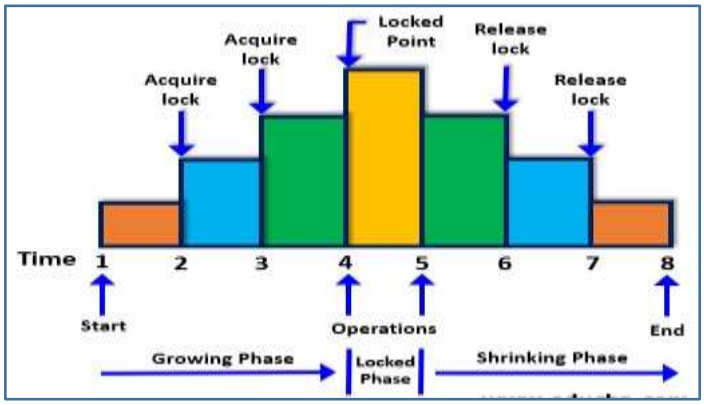
If a lock cannot be granted, the requesting transaction is made to wait till all incompatible locks held by other transactions have been released. Then the lock is granted.

3. Two phase Locking (2PL) Protocols

- Two-phase locking protocol which is also known as a 2PL
- Two Phase Locking (2PL) is a concurrency control locking protocol
- In this type of locking protocol,the transaction should acquire a lock after it releases one of its locks.
- This locking protocol divides the execution phase of a transaction into three different parts
  1. In the first phase ,when the transction begins to execute ,it requres for the locks it needs
  2. The second part is where the transaction obtains all the locks when a transaction release its first lock,the third phase starts
  3. In the third phase,the transaction cannot demand any new locks.Instead it only release the acquired locks.

There are two phases in this approach:

- **Growing Phase**
- **Shrinking Phase**

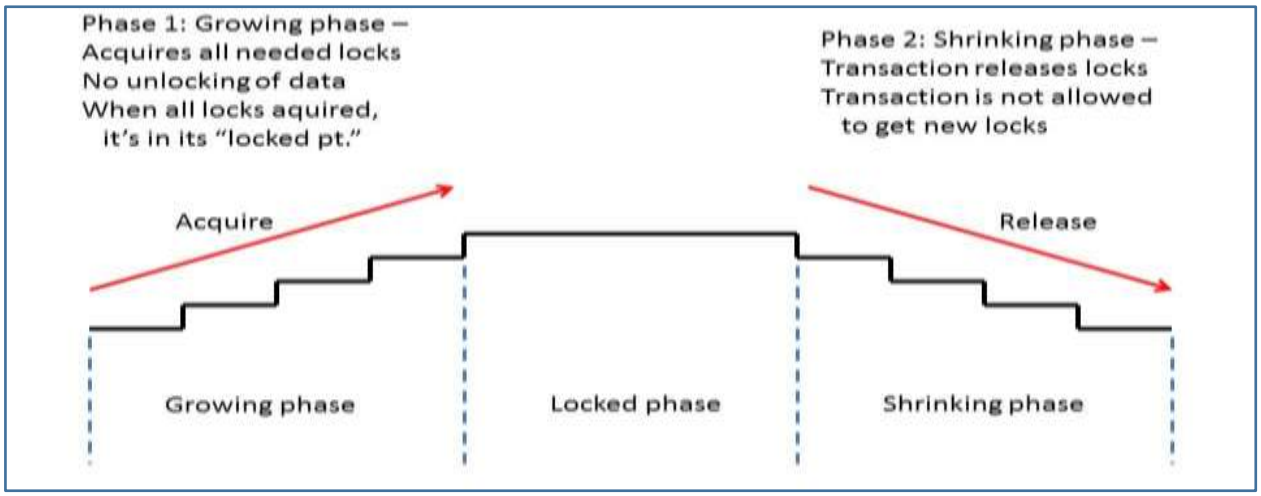


Growing Phase

New locks on items can be acquired.In this phase transaction may obtain locks but may not release any locks is known as growing phase.It is also called expanding phase.

Shrinking Phase

Existing locks,but no new lock can be acquired.In this phase transaction may release locks but not obtain any new lock



Example

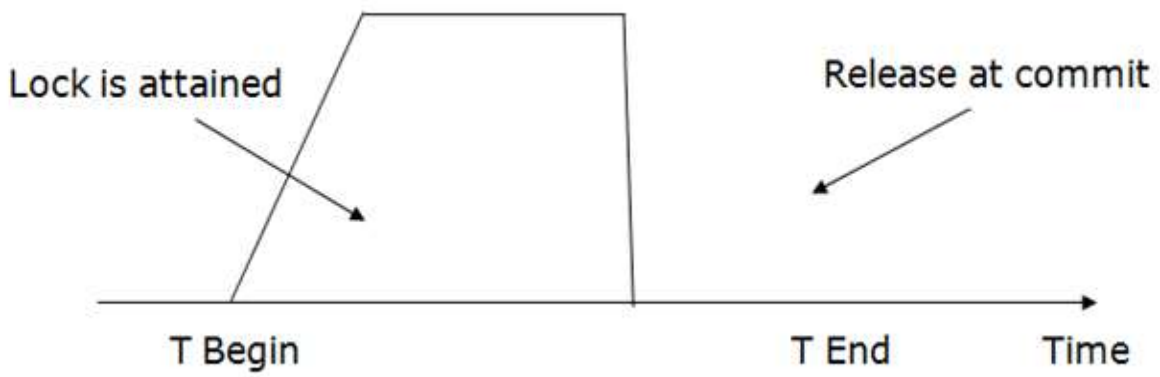
	T1	T2
0	LOCK-S(A)	
1		LOCK-S(A)
2	LOCK-X(B)	
3	———	———
4	UNLOCK(A)	
5		LOCK-X(C)
6	UNLOCK(B)	
7		UNLOCK(A)
8		UNLOCK(C)
9	———	———

Transaction T1:
Growing phase: from step 0-2
Shrinking phase: from step 4-6
Lock point: at 3

Transaction T2:
Growing phase: from step 1-5
Shrinking phase: from step 7-8
Lock point: at 6

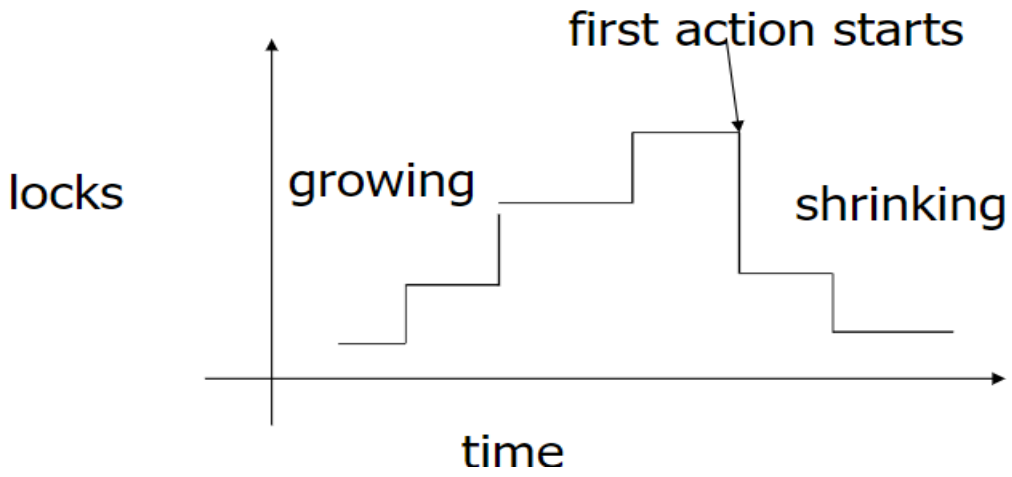
Strict Two-phase locking (Strict-2PL)

- The first phase of Strict-2PL is similar to 2PL. In the first phase, after acquiring all the locks, the transaction continues to execute normally.
- The only difference between 2PL and strict 2PL is that Strict-2PL does not release a lock after using it.
- Strict-2PL waits until the whole transaction to commit, and then it releases all the locks at a time.
- Strict-2PL protocol does not have shrinking phase of lock release.



Conservative 2PL

Lock all items it needs then transaction starts execution If any locks can not be obtained, then do not lock anything Difficult but deadlock free



MULTI VETSION TWO PHASE LOCKING

	Read	Write	Certify
Read	Yes	Yes	No
Write	Yes	No	No
Certify	No	No	No

The idea behind multiversion 2PL is to allow other transactions T to read an item X while a single transaction T holds a write lock on X.

This is accomplished by allowing two versions for each item X;

The second version X is created when a transaction T acquires a write lock on the item. Other transactions can continue to read the committed version of X while T holds the write lock.

Transaction T can write the value of X as needed, without affecting the value of the committed version X. However, once T is ready to commit, it must obtain a certify lock on all items that it currently holds write locks on before it can commit.

The certify lock is not compatible with read locks, so the transaction may have to delay its commit until all its write-locked items are released by any reading transactions in order to obtain the certify locks

Once the certify locks—which are exclusive locks—are acquired, the committed version X of the data item is set to the value of version X ,

(2PL) for DDBMS

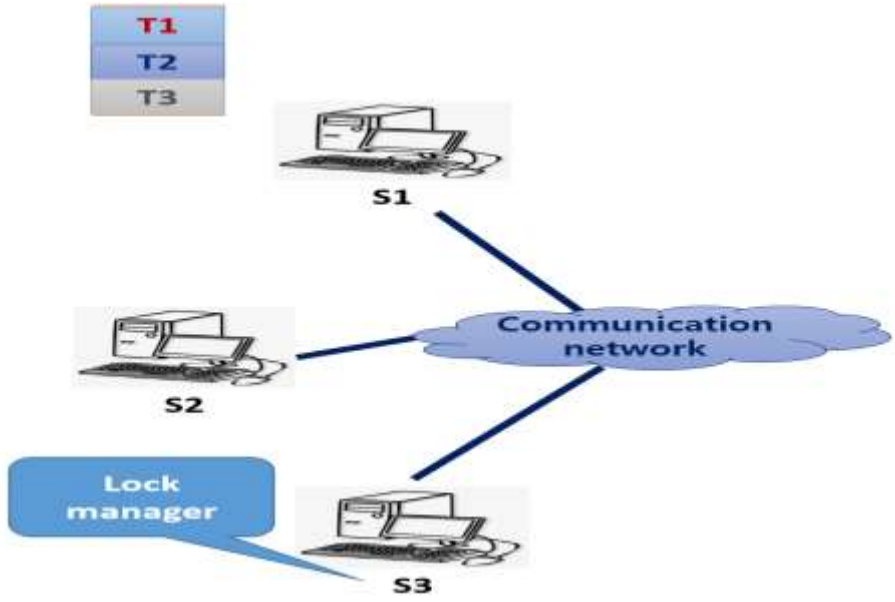
Types of 2PL

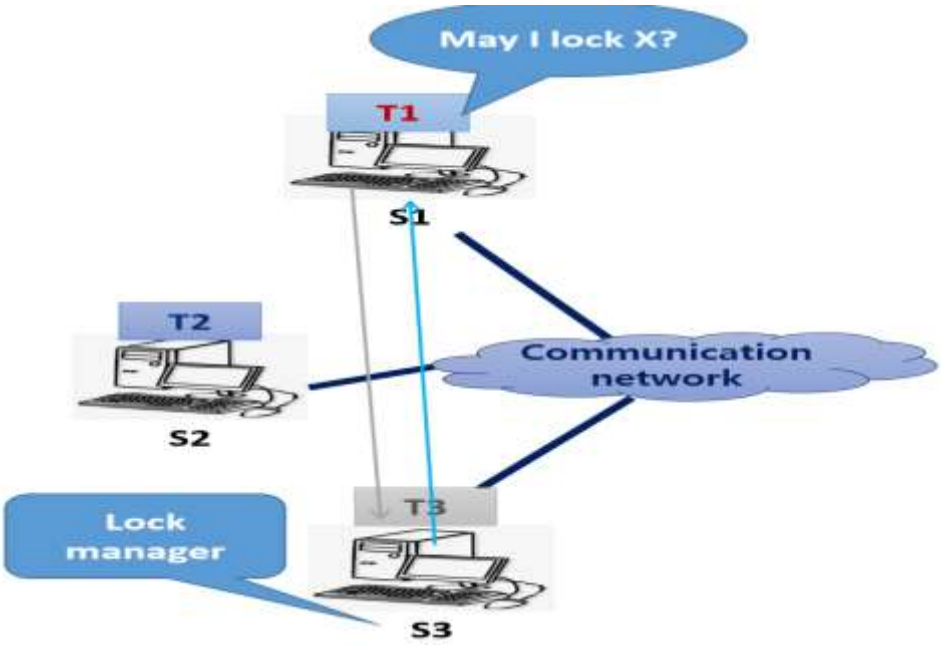
1. Centralized 2PL
2. Primary copy 2PL

3. Distributed 2PL

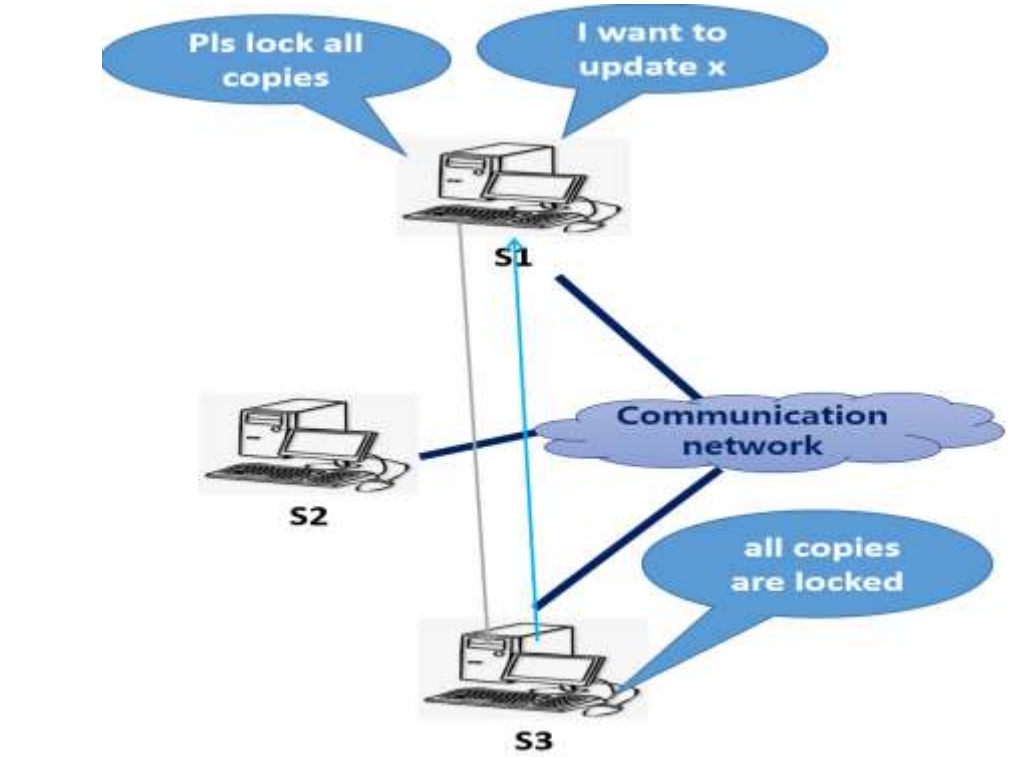
Centralized 2PL/Single lock manager

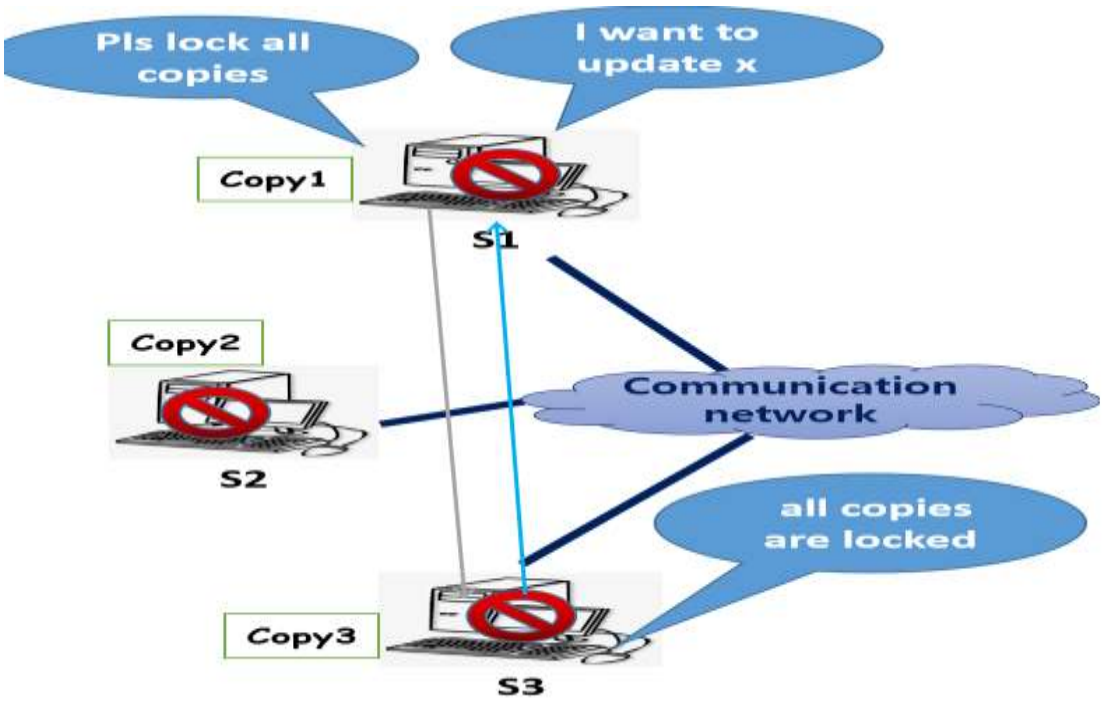
- System maintains a single lock manager that resides in a single chosen site,say S3
- Transaction Coordinator at site S1 divides the transactions
- Transaction Coordinator at site S1 acts as global transaction manager or transaction coordinator
- DM(Data Manager) component at site S1 sends the subtransactions to the appropriate sites
- When a transaction needs to lock a data item,it sends a lock request to S3 & lock manager determines whether the lock can be granted immediately
- If yes, lock manager sends a message to the site which initiated the request.
- If no, request is delayed until it can be granted ,at which time a message is sent to the initiating site.





- If the transaction involves an update of the data item that is replicated,the coordinator must ensure that all copies of the data item are updated
- The coordinator requests write locks on all copies before updating each copy & releasing the locks
- The coordinator can elect to use any copy of the item for reads ,generally the copy at its site,if one exists



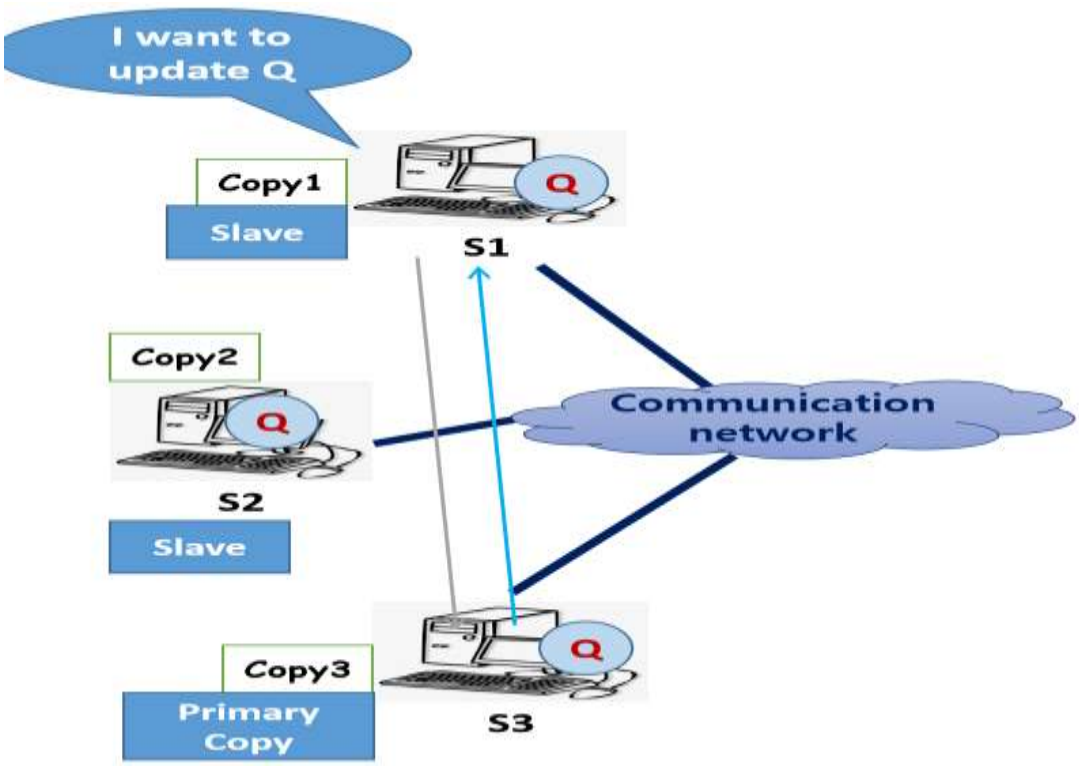


Advantages:
Simple implementation
Simple deadlock hanling

Disadvantages :
Bottleneck:Lock manager site becomes a bottoleneck
Vulnerability:system is vulnerable to lock manager site failure.

Primary Copy 2PL

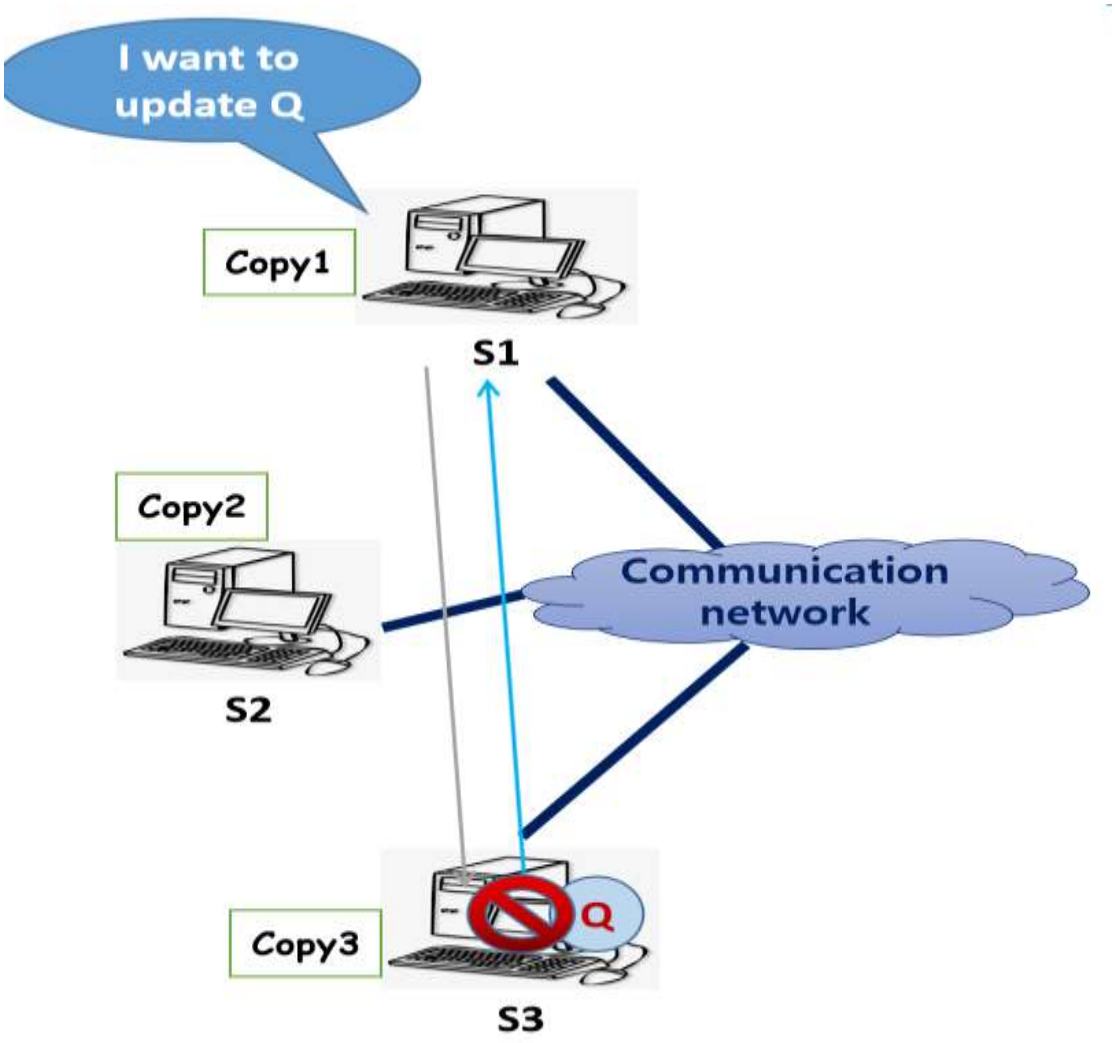
- Choose one replica of data item to be the primary copy
- Site containg the replica is called the primary site for that data item
- Other copies are called Slave copies
- Different data item can have different primary site
- When a transaction needs to lock a data item Q(update),it requests a lock at the primary site of Q
- The response to the request is delayed until it can be granted
- Implicitly gets lock on all replicas of the data item
- Once the primary copy has been updated the change can be propageted to the slave copies
- The propagation should be carried out ASAP to prevent other transactions from reading out-of -date values



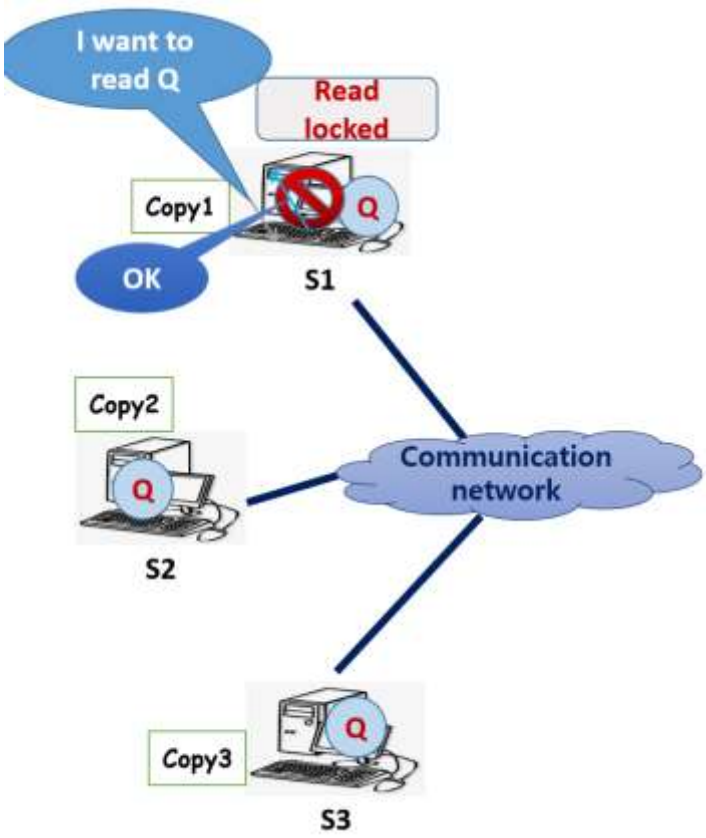
Disadvantages:
If the Primary site of Q fails ,Q is inaccessible even though other sites containing a replica may be accessible.

Distributed lock manager(2PL)

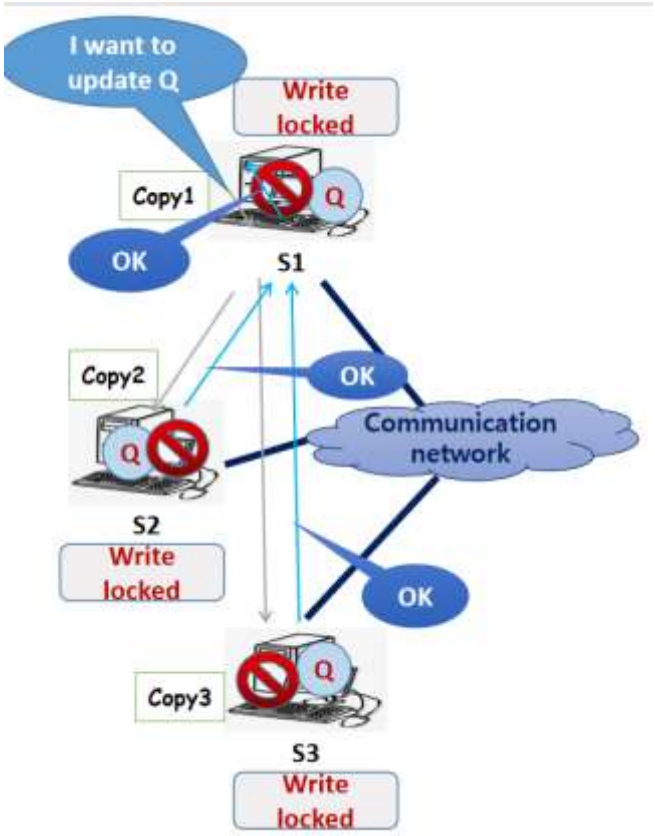
- Lock managers are assigned at each site
- Lock managers control acess to local data items
- When a transaction wishes to lock a data item Q ,which is not replicated & resides at site S3 ,a message is sent to the lock manager at site S3,requesting a lock
- if Q is locked in an incompatible mode ,the request is delayed until it can be granted
- Once the lock request is granted the lock manager sends a message back to the initiator indicating that it has granted the lock request



- When a transaction wishes to lock a data item Q,which is replicated, Read-One-Write -All is implemented
- Any copy can be used for reading a data item



All copies must be write locked before an item can be updated



Advantages:

Work is distributed & can be robust in failure

Disadvantages :
Deadlock detection is more complicated due to multiple lock managers

4. Time stamp-based Protocols

Time stamp=Assigning time to a trasaction

or

Assigning logical counter to any new transaction

- Basic idea of Time stamping is to decide the order between the transactions before that enters into the system.So that in case of conflict during execution ,we can resolve the conflict using ordering
- Time stamp-based protocols uses a timestamp to serialize the execution of concurrent transactions.
- The reason we call timestamp not stamp because for stamping we use the value of system clock(as it will always be unique or never repeat itself)
- This protocols ensure that every conflicting read & write operations are executed in time stamp order
- Every transaction is timestamp with its start time

Two ideas of time stamping

Time stamp with transaction:

With each transactionTi ,we associated a time stamp denoted by T.S(Ti).It is the value of the system clock when a transaction enters into the system.So if a new transaction Tj enters after Ti then T.S(Ti) < T.S(Tj) always unique will remain fixed through the execution

Time stamp with data item:

For each data item Q protocol maintains two time stamps

W-time stamp(Q):is the latest time stamp of any transaction that executed Write(Q) successfully

R-time stamp(Q):is the latest time stamp of any transaction that executed Read(Q)successfully

T1(5)	T2(10)	T3(12)
R(Q)		
	R(Q)	
		R(Q)

T1(5)	T2(10)	T3(12)
W(Q)		

		W(Q)
	W(Q)	

Basic TO protocol

The Basic TO protocol in the following two cases:

1. Transaction T issues a read(A) operation:

If (write_TS(A)>TS(T))
  abort T and Rollback;
else{
  read(A);
  read_TS(A)=TS(T);
}

T1(5)	**T2(10)**
	W(A)
R(A)	

Roll back

T1(5)	**T2(10)**
W(A)	
	R(A)

2. Transaction T issues a write (A)operation,

If (read_TS(A)>TS(T) OR write_TS(A)>TS(T) abort T and Rollback;
else{
  write(A);
  write_TS(A)=TS(T);
}

T1(5)	**T2(10)**
	R(A)

W(A)	

Roll back

T1(5)	**T2(10)**
	W(A)
W(A)	

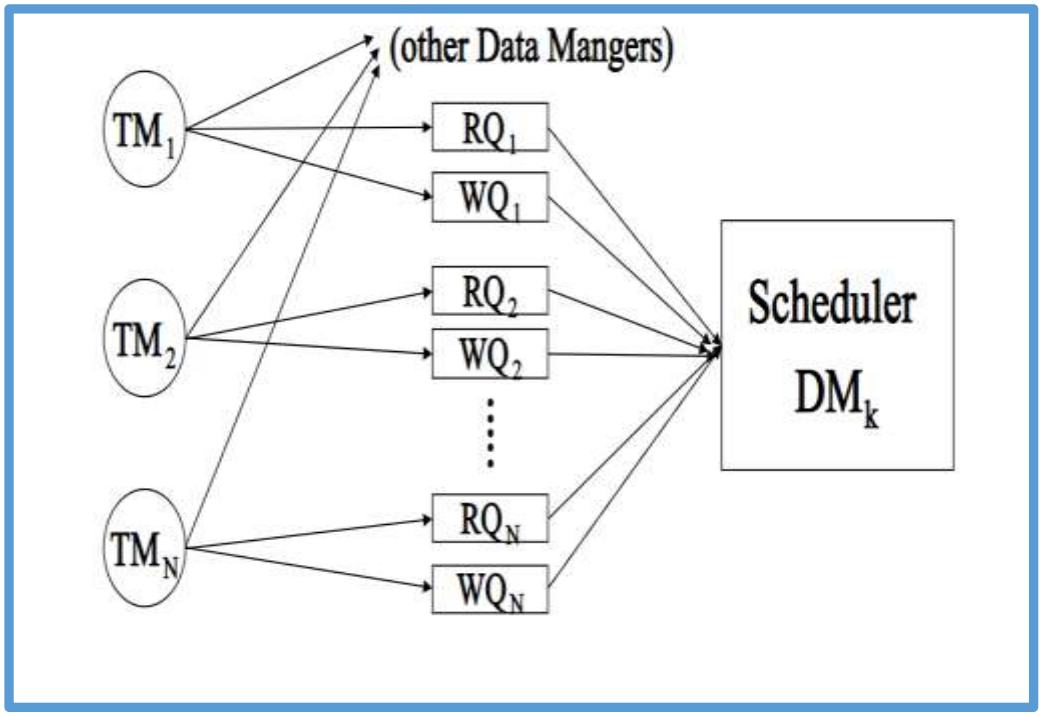
Roll back

T1(5)	**T2(10)**
R(A)	
	W(A)

T1(5)	**T2(10)**
W(A)	
	W(A)

Conservative Timestamp Ordering

Each Data Manager maintains:
a read queue (RQi)
a write queue (WQi)
for each Transaction Manger, TMi
Let: TS(Qi) denote the timestamp of the first operation in Qi
Let: TS(Qi) denote the timestamp of the first operation in Qi
read <object,TS>
if (non-empty(WQi) and TS(WQi) > TS for i = 1 ….N)
then execute the read operation
else add the read operation to RQi
write<object,val,TS>
if (non-empty (RQi) and non-empty (WQi) and
TS(RQi) > TS or TS(WQi) > TS for i = 1….N)
then execute the write operation
else add the write operation to WQi



Unique timestamp generation is difficult in DDBMS than centralized DBMS

The principle idea behind the timestamp based concurrency control technique is that a unique timestamp is assigned to each transaction to determine the serialization order of tansaction in a distributed system.

The serialization order is determined depending on the timestamp values of transactions prior to the execution of transactions.

A timestamp value is a simple identifier that is used to define each transaction uniquely & to permit ordering.

In a centralized DBMS, generation of unique timestamp values for a transaction can be handled in a simpler way while in a DDBMS time stamp values must be derived from a totally ordered domain

There are two primarily methods for generating unique timestamp value in a distributed environment;centralized & distributed

In the centralized approach,a single site is responsible for assigning unique timestamp values to all transactions that are generated at different sites of the distributed system.The central site can use a logical counter or its own system clock for assigning timestamp values.The centralized approach is very simple.

In the **distributed approach,**each site generates unique local time stamp values by logical counter or its own system clock.This approach is difficult comparing to the centralized one

5. Validation based Protocols

Validation Based Protocol is also called Optimistic Concurrency Control Technique.

In the validation based protocol, the transaction Ti is executed in the following three phases:

Read phase : In this phase, the transaction Ti reads the value of various data items and stores them in temporary local variables of Ti.i.e it does not update actual database.

Validation Phase: Transaction Ti perform validation test whether it can copy value of its local variable to database.Checking is performed to make sure that there is no violation of serializability when the tra-

saction updates are applied to the database.

Write Phase: If the validation of the transaction Ti is validated, then the temporary results are written to the database i.e system applies actual update to the database .

To perform validation test,we need to know when the various phases of transaction Ti took place.

We shall therefore associate three different timestamp with transaction Ti.

Here each phase has the following different timestamps:

1. Start(Ti): It is the time when Ti started its execution.

2. Validation(Ti): It is the time when Ti just finished its read phase and begin its validation phase.

4. **Finish(Ti):** the time when Ti end it's all writing operations in the database under write-phase.

We determine the serializability order by timestamp ordering technique,using the value of timestamp Validation(Ti).

Thus value TS(Ti)=Validation(Ti)

VALIDATION TEST:

The validation test for transaction Tj requires that for all transaction Ti with TS(Ti)<TS(Tj) one of the following condition hold.

Finish(Ti)<Starts(Tj), since Ti finished before Tj started hence serializability order is indeed maintained.

Finish(Ti)<Validation(Tj).This ensures actual write by Ti and Tj will not overlap.

Validation(Ti)<Validation(Tj).It ensures that Ti has completed read phase before Tj completed read phase.

Example:

VALIDATION TEST:

1. Finish(Ti)<Starts(Tj)
2. Finish(Ti)<Validation(Tj)
3. Validation(Ti)<Validation(Tj)

Ti	**Tj**
Read(B)//15	
	Read(B)//15
	B=B*5//75
	Read(A)//10
	A=A+10//20
Read(A)//10	
<Validate>	
display(A+B)//25	**<Validate>**
	Write(A)//20
	Write(B)//75

Advantages:

1. **Avoid Cascading-rollbacks:** This validation based scheme avoid cascading rollbacks since the final write operations to the database are performed only after the transaction passes the validation phase. If the transaction fails then no updation operation is performed in the database. So no dirty read will happen hence possibilities cascading-rollback would be null.
2. **Avoid deadlock:** Since a strict time-stamping based technique is used to maintain the specific order of transactions. Hence deadlock isn't possible in this scheme.

Disadvantages:

1. **Starvation:** There might be a possibility of starvation for long-term transactions, due to a sequence of conflicting short-term transactions that cause the repeated sequence of restarts of the long-term transactions so on and so forth.

Conclusion

To sum up, concurrency control is essential to guaranteeing data consistency and integrity in database management systems (DBMS). Strong mechanisms for controlling access to shared data are offered by lock-based protocols, including Two-Phase Locking (2PL), which use read (shared) and write (exclusive) locks. Certain 2PL variants—such as Strict Two-Phase Locking and Conservative 2PL—offer specialized methods for handling transactions, while Distributed 2PL tackles the particular difficulties associated with distributed databases. Transaction timestamps are used by time stamp-based protocols, such as Conservative Timestamp Ordering and Basic Time Ordering, to effectively sequence transactions and resolve conflicts. Optimistic Replication and Sagas are two examples of validation-based protocols that provide an optimistic approach to concurrency control by enabling transactions to continue without locks and resolving conflicts during the validation stage.

Overall, the cooperative nature of these concurrency control techniques guarantees consistent and seamless transaction execution in multi-user database management systems (DBMSs), supporting dependable and effective data administration.

References

1. Agrawal, R., & El Abbadi, A. (1987). Efficient Concurrency Control for Broadcast Environments. ACM SIGMOD Record, 16(3), 212-223.
2. Kung, H. T., & Robinson, J. T. (1981). On Optimistic Methods for Concurrency Control. ACM Transactions on Database Systems (TODS), 6(2), 213-226.
3. Papadimitriou, C. H., & Yannakakis, M. (1985). On the Complexity of Database Consistency Problems. Journal of the ACM (JACM), 32(3), 562-586.
4. Lynch, N. A., Merritt, M. J., & Weihl, W. E. (1992). Timing-Based Algorithms for Distributed Systems. Distributed Computing, 6(1), 51-61.
5. Saito, Y., & Shapiro, M. (2005). Optimistic Replication. ACM Computing Surveys (CSUR), 37(1), 42-81.
6. Garcia-Molina, H., & Salem, K. (1987). Sagas. Proceedings of the ACM SIGMOD International Conference on Management of Data, 249-259.