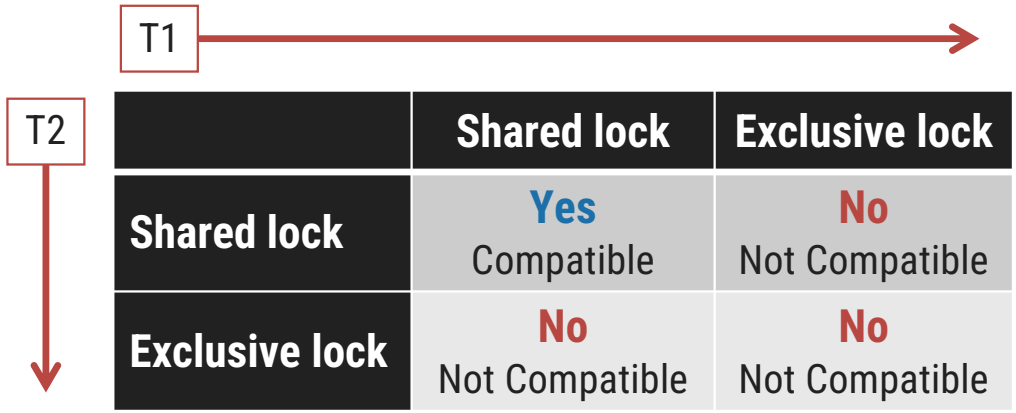
Lock based protocol

- Data items can be locked in two modes :
  - **Shared (S) mode**: When we take this lock **we can just read the item but cannot write**.
  - **Exclusive (X) mode**: When we take this lock **we can read as well as write the item**.
- Lock-compatibility matrix



	Shared lock	Exclusive lock
Shared lock	**Yes** Compatible	**No** Not Compatible
Exclusive lock	**No** Not Compatible	**No** Not Compatible

- 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**.
- If a lock cannot be granted, the requesting transaction is made to wait till all incompatible locks held by other transactions have been released. The lock is then granted.
- **Any number of transactions can hold shared locks** on an item, but **if any transaction holds an exclusive on the item no other transaction can hold any lock** on the item.

Lock based protocol

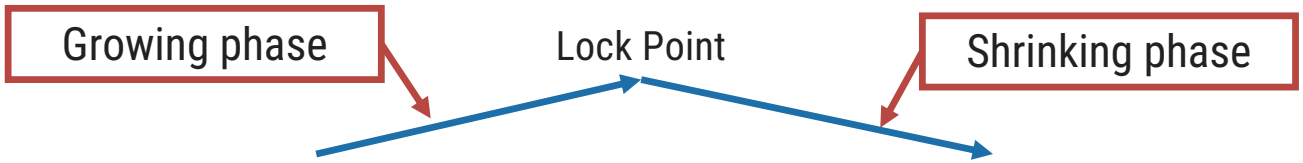
Non-Serial Schedule (S1)	
T1	**T2**
Lock-X (A)	
Read (A)	
Write (A)	
Unlock (A)	
	Lock-S (B)
	Read (B)
	Unlock (B)
Lock-X (B)	
Read (B)	
Write (B)	
Unlock(B)	
	Lock-S (A)
	Read (A)
	Unlock (A)

Two phase locking (2PL) protocol (Basic)

- This protocol works in two phases,

1. Growing Phase
   - In this phase a **transaction obtains locks**, but **can not release any lock**.
   - When a transaction takes the final lock is called lock point.
2. Shrinking Phase
   - In this phase a **transaction can release locks**, but **can not obtain any lock**.
   - The **transaction enters the shrinking phase as soon as it releases the first lock** after crossing the Lock Point.

- Transactions can perform **read/write operations** both **in growing and shrinking phase**.



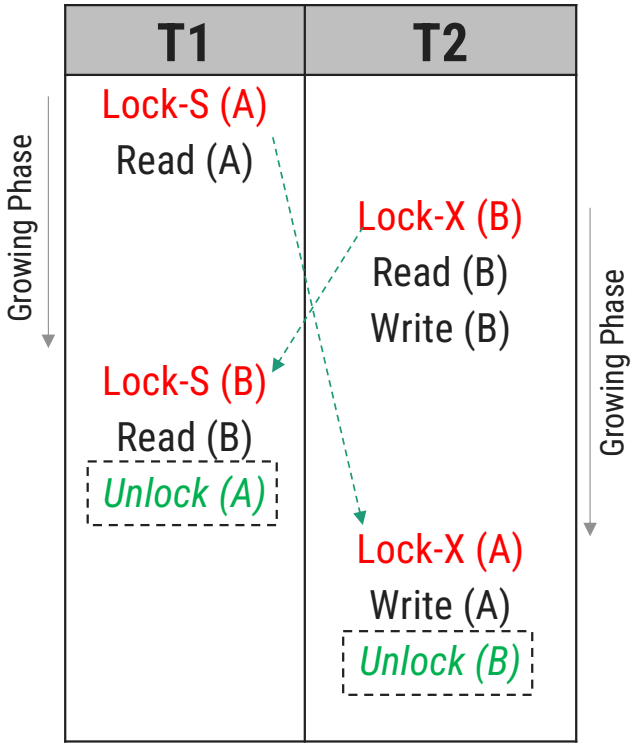
- Two-phase locking protocol ensures **conflict serializability** but does **not** ensure freedom from deadlock.
- Cascading rollback may occur under two-phase locking.

Two Phase Locking (2PL) protocol – Example

Non-Serial Schedule (S1)	
T1	**T2**
Lock-X (A)	
Read (A)	
Write (A)	
Lock-X (B)	
Unlock (A)	
	Lock-S (A)
	Read (A)
Read (B)	
Write (B)	
Unlock (B)	
	Lock-S (B)
	Read (B)

Non-Serial Schedule (S2)	
T1	**T2**
Lock-S (A)	
Read (A)	
	Lock-X (B)
	Read (B)
	Unlock (B)
Lock-X (C)	
Write (C)	
Unlock (A)	
Unlock (C)	
	Not allowed
	Lock-X (C)
	Write (C)

Two Phase Locking (2PL) protocol – Issues



Deadlock

T1	T2	T3	T4
	Lock-S (A)		
Lock-X (A)	.	Lock-S (A)	
Waiting	.	.	Lock-S (A)
	.	.	.
	.	.	.
	Unlock (A)	.	.
Lock-X (A)		.	.
Waiting		.	.
		Unlock (A)	.
Lock-X (A)			.
Waiting			.
			Unlock (A)

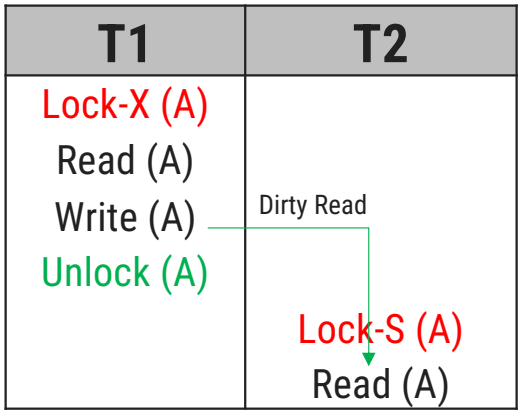
Starvation

T_5	T_6	T_7
lock-X(A)		
read(A)		
lock-S(B)		
read(B)		
write(A)		
unlock(A)		
	lock-X(A)	
	read(A)	
	write(A)	
	unlock(A)	
		lock-S(A)
		read(A)

Possibility of Irrecoverability

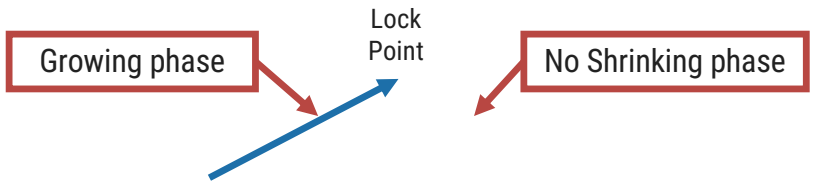
Conservative/Static 2PL protocol

- There is **no growing phase**. First, Transactions will **acquire all the required LOCKS** and then directly will start from lock point.
- If all locks are not available then transaction must release the lock acquired so far and wait.
- Shrinking phase works as usual and transactions can unlock any data item at any time.
- It **requires prior knowledge** of all the required data items to hold the lock on them before starting the execution.
- Independent from Deadlock because all the locks are acquired beforehand.
- Possibility of irrecoverable schedule due to dirty read problem



Rigorous 2PL Protocol

- It is an improvement over 2PL protocol where we try to ensure recoverability and cascade-lessness.
- In this protocol, a **transaction is not allowed to release any lock (either shared or exclusive) until it commits**.
- This means that **until the transaction commits, other transaction can not acquire even a shared lock on a data item on which the uncommitted transaction has a shared lock**.
- Ensures recoverability, C.S., and V.S.
- Suffer from deadlock and inefficiency.



Strict 2PL protocol

- It is an improvement over Rigorous 2PL.
- In this protocol, a **transaction may release all the shared locks after the Lock Point has been reached**, but **it cannot release any of the exclusive locks until the transaction commits or aborts**.
- It **ensures that if data is being modified by one transaction, then other transaction cannot read it until first transaction commits**.
- This protocol **solves dirty read problem** and ensures recoverability.
- All the properties are same as that of Rigorous 2PL but it provides better concurrency.