Two Phase Locking

A transaction is said to follow the Two-Phase Locking protocol if Locking and Unlocking can be done in two phases.

1. **Growing Phase:** New locks on data items may be acquired but none can be released.
2. **Shrinking Phase:** Existing locks may be released but no new locks can be acquired.

Note – If lock conversion is allowed, then upgrading of lock(from S(a) to X(a)) is allowed in the Growing Phase, and downgrading of lock (from X(a) to S(a)) must be done in shrinking phase.
Let's see a transaction implementing 2-PL.

	T_1	T_2
1	lock-S(A)	
2		lock-S(A)
3	lock-X(B)	
4	…….	……
5	Unlock(A)	
6		Lock-X(C)
7	Unlock(B)	
8		Unlock(A)
9		Unlock(C)
10	…….	……

This is just a skeleton transaction that shows how unlocking and locking work with 2-PL.
Note for:
Transaction T_1:
- The growing Phase is from steps 1-3.
- The shrinking Phase is from steps 5-7.
- Lock Point at 3

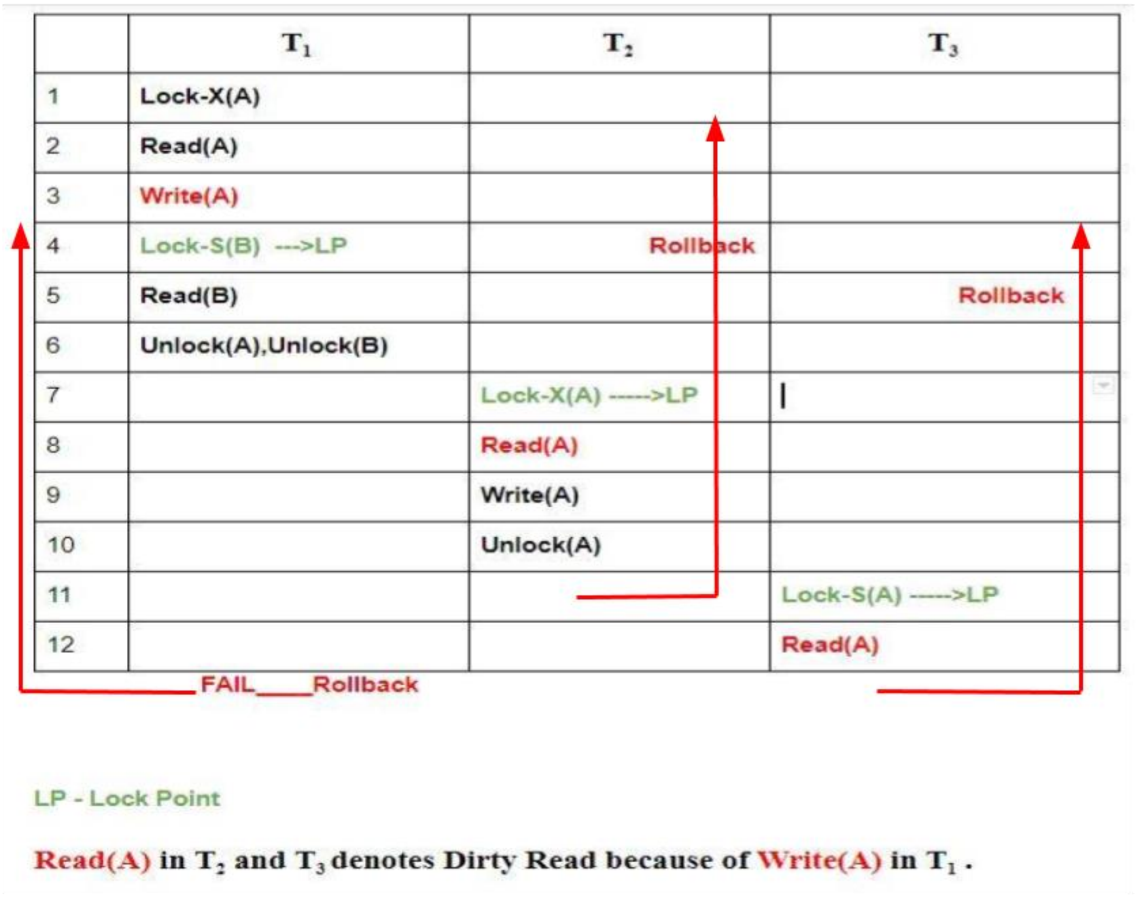
Transaction T_2:
- The growing Phase is from steps 2-6.
- The shrinking Phase is from steps 8-9.
- Lock Point at 6

LOCK POINT:
The Point at which the growing phase ends, i.e., when a transaction takes the final lock it needs to carry on its work. Now look at the schedule, you'll surely understand.

I have said that 2-PL ensures serializability, but there are still some drawbacks of 2-PL. Let's glance at the drawbacks:

- Cascading Rollback is possible under 2-PL.
- Deadlocks and Starvation are possible.

Cascading Rollbacks in 2-PL –
Let's see the following Schedule:



	T_1	T_2	T_3
1	Lock-X(A)		
2	Read(A)		
3	Write(A)		
4	Lock-S(B) --->LP		
5	Read(B)		
6	Unlock(A),Unlock(B)		
7		Lock-X(A) ----->LP	
8		Read(A)	
9		Write(A)	
10		Unlock(A)	
11			Lock-S(A) ----->LP
12			Read(A)

Rollback
Rollback
FAIL____Rollback

LP - Lock Point

Read(A) in T_2 and T_3 denotes Dirty Read because of Write(A) in T_1 .

Take a moment to analyze the schedule. Yes, you're correct, because of Dirty Read in T_2 and T_3 in lines 8 and 12 respectively, when T_1 failed we have to roll back others also.
Hence, **Cascading Rollbacks are possible in 2-PL.** I have taken skeleton schedules as examples because it's easy to understand when it's kept simple. When explained with real-time transaction problems with many variables, it becomes very complex.

Deadlock in 2-PL –
Consider this simple example, it will be easy to understand. Say we have two transactions T_1 and T_2.
Schedule: Lock-X_1(A) Lock-X_2(B) Lock-X_1(B) Lock-X_2(A)
Drawing the precedence graph, you may detect the loop. So Deadlock is also possible in 2-PL.

Two-phase locking may also limit the amount of concurrency that occurs in a schedule because a Transaction may not be able to release an item after it has used it. This may be because of the protocols and other restrictions we may put on the schedule to ensure serializability, deadlock freedom, and other factors. This is the price we have to pay to ensure serializability and other factors, hence it can be considered as a bargain between concurrency and maintaining the ACID properties.

The above-mentioned type of 2-PL is called **Basic 2PL**. To sum it up it ensures Conflict Serializability but ***does not*** prevent Cascading Rollback and Deadlock. Further, we will study three other types of 2PL, Strict 2PL, Conservative 2PL, and Rigorous 2PL.

Problem with Two-Phase Locking:
1. It does not insure recoverability which can be solved by strict two-phase locking and rigorous two-phase locking.
2. It does not ensure a cascadeless schedule which can be solved by strict two-phase locking and rigorous two-phase locking.
3. It may suffer from deadlock which can be solved by conservative two-phase locking.