Two-phase locking protocol

→ one protocol that ensures serializability is the two-phase locking protocol.

→ This protocol requires that each transaction issue lock and unlock requests in two phases.

(1) Growing Phase:- A transaction may obtain locks, but may not release any lock. lock-S(A) } lock-X(A) }
It should not use unlock(A).

(2) Shrinking Phase:- A transaction may release locks, but may not obtain any new locks. unlock(A).
It can't use the lock(A).

Eg:-

T_1	T_2
lock-X(B) → GP	
read(B)	
B := B-50	
write(B)	
	lock-S(A) → GP } growing phase
	read(A)
	lock-S(B) → GP }
	read(B)
	display(A+B)
	unlock(A) } shrinking phase.
	unlock(B) }
lock-X(A) → GP.	
read(A)	
A := A+50	
write(A)	
unlock(B) → SP } shrinking phase.	
unlock(A) → SP }	

Both T_1 & T_2 are two-phase locking protocol.

Schedule 1

T_1	T_2
lock-X(B) → GP read(B) B := B-50 write(B) unlock(B) → SP	lock-S(A) → GP read(A) unlock(A) → SP lock-S(B) → GP read(B) unlock(B) → SP display(A+B)
lock-X(A) → GP read(A) A := A+50 write(A) unlock(A)	

not a growing phase because lock followed by unlock.

not a GP

not a GP

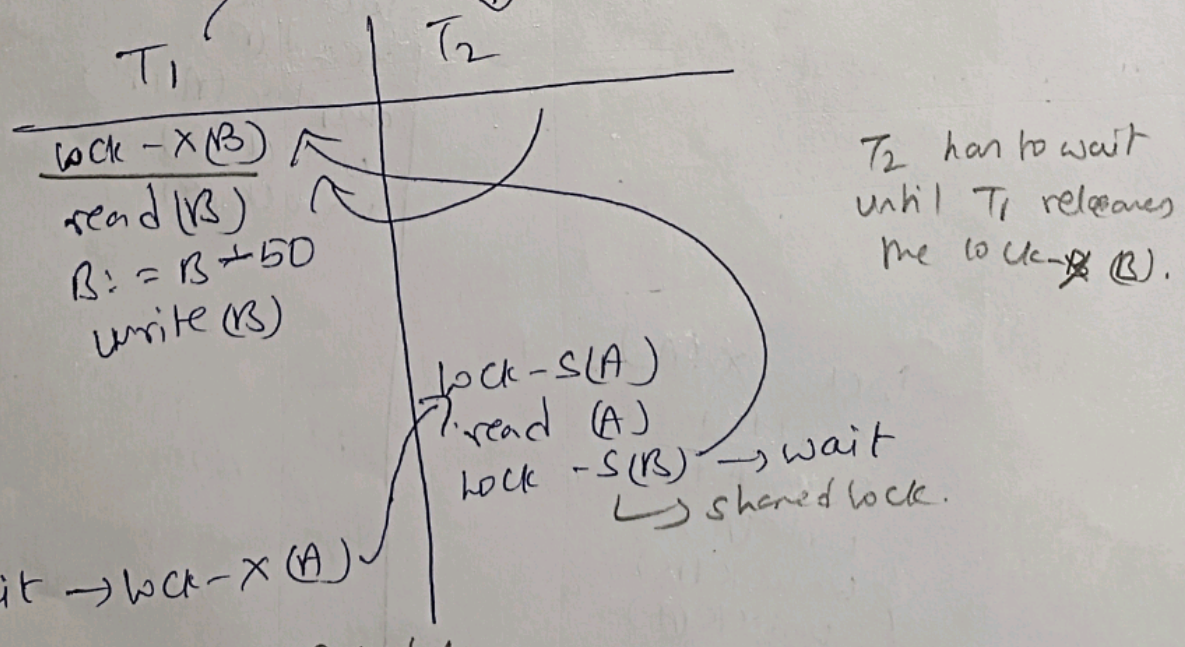
T_2 is not a 2-phase locking protocol.

T_1 is not in 2-phase

schedule-2

Two-phase locking does not ensure freedom from deadlock.

cycle waiting so deadlock occurs.



T_2 has to wait until T_1 releases the lock-X(B).

Schedule-3

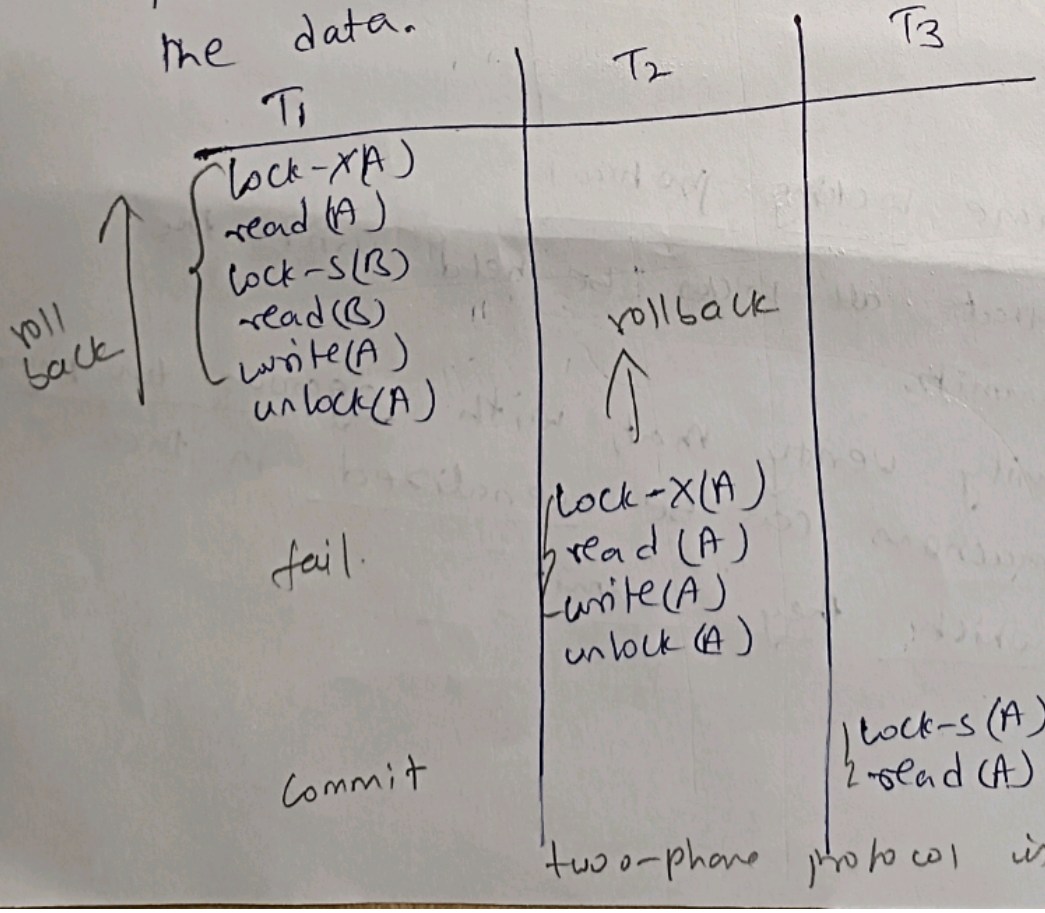
Strict - two - phase locking protocol

→ cascading rollback may occur under two-phase locking.

→ cascading rollbacks can be avoided by a modification of two-phase locking called the strict two-phase locking protocol.

→ This protocol requires not only that locking be two phase, but also that all exclusive-mode locks taken by a transaction be held until that transaction commits.

→ This requirement ensures that any data written by an uncommitted transaction are locked in exclusive mode until the transaction commits, preventing any other transaction from reading the data.



two-phase protocol is converted to

T_1	T_2	T_3
lock-X(A) read(A) lock-S(B) read(B) write(A)		
fail rollback		
Committed unlock(A)	lock-X(A) → T_2 will start after T_1 enter into commit & unlock(A). read(A) write(A) commit unlock(A)	
		lock-S(A) read(A)

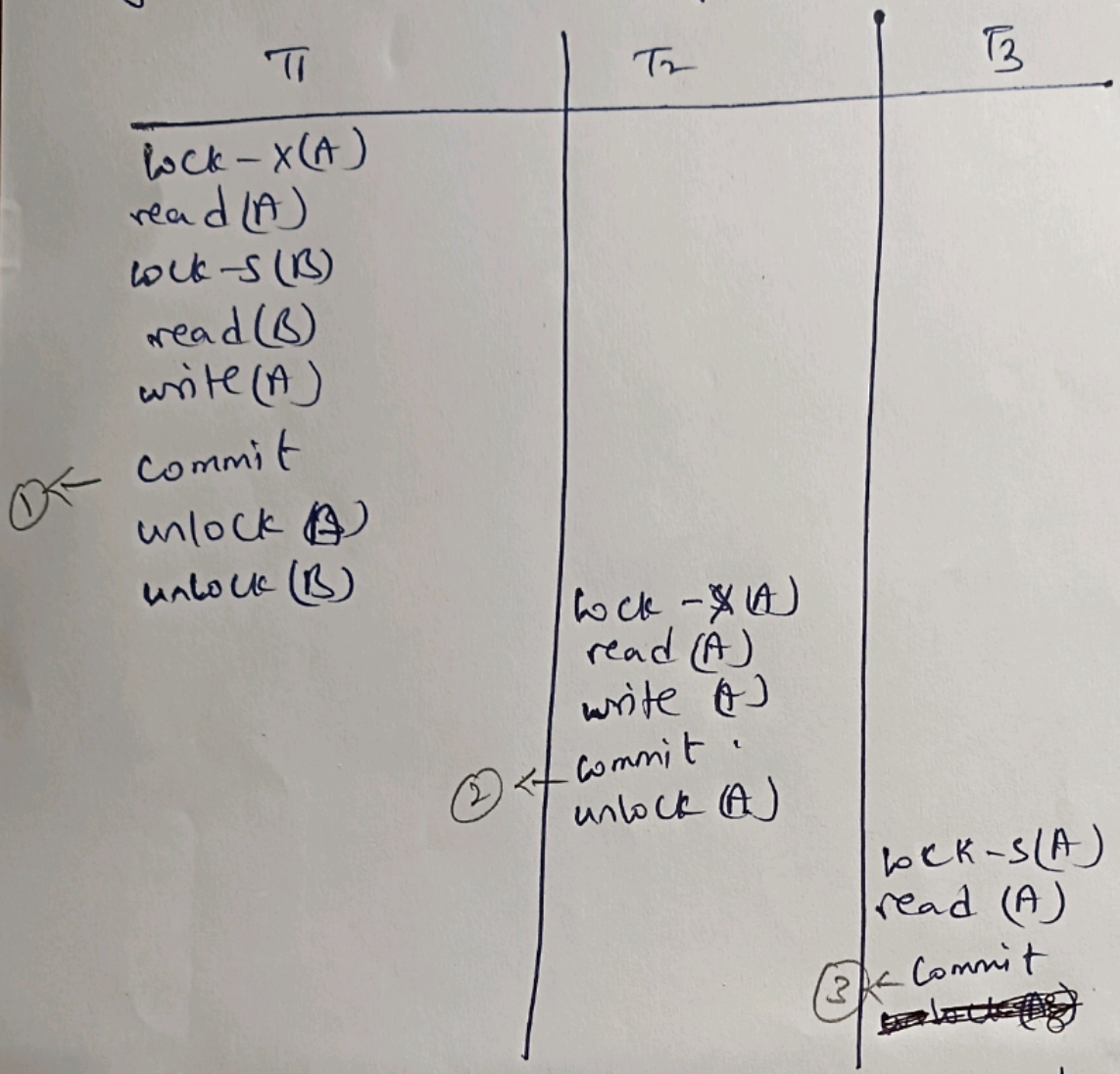
strict-two-phase locking protocol.

Rigorous two-phase locking protocol

→ It requires that all locks be held until the transaction commits.

→ we can easily verify that, with rigorous two-phase locking transactions can be serialized in the order in which they commit.

→ Most database system implement either strict or rigorous two-phase locking. (4)



$\langle T_1, T_2, T_3 \rangle$ are serialized in the order in which they committed.