Problem with 2PL

- Unnecessary Early Lock
- Cascading rollback
- Deadlock

T1	T2
LOCK-X(A)	
R(A)	
W(A)	
LOCK-X(B)	
	LOCK-S(A)(WAIT BS OF EARLY LOCK OF A BY T1

T1	T2
LOCK-X(B)	
R(B)	
W(B)	

	LOCK-S(A)
	R(A)

	LOCK-X(B)
LOCK-X(A)	

Cascading rollback

T1	T2	T3
XL(A)		
R(A)		
W(A)		
U(A)		
	XL(A)	
	R(A)	
	W(A)	
	U(A)	
		XL(A)
		R(A)
		W(A)
		U(A)

Conservative 2PL

Prevents deadlock by locking all desired data items before transaction begins execution.

No Growing Phase

All locks are granted ->U(A)-> U(B)

Strict 2PL

Basic: Transaction locks data items incrementally. This may cause deadlock which is dealt with.

A more stricter version of Basic algorithm :Strict

Unlocking is performed after a transaction terminates (commits or aborts and rolled-back).

This is the most commonly used two-phase locking algorithm.

RL(A)-WX(B)-RL(c)-U(A)-U(C) –

Dead lock is allowed- recovery is easy

Rigorous 2PL

- a transaction T does not release any of its locks until after it commits or abort

EXAMPLE

B2PL

LOCK-S(A)
R(A)
LOCK-X(B)
R(A)
R(B)
B=A+B
UNLOCK(A)
W(A)
UNLOCK(B)

B2PL,
S2PL

LOCK-S(A)
R(A)
LOCK-X(B)
UNLOCK(A)
R(A)
W(B)
COMMIT
UNLOCK(B)

LOCK-S(A)
R(A)
UNLOCK-X(A)
LOCK-X(B)
R(B)
W(B)
COMMIT
UNLOCK(B)
COMMIT