

# Correction #1

- Incorrect: In discussion, I stated that the schedule is distinct from the actual order in which actions are executed. This is not true.
- Correct: Actions that need to wait are incorporated into the schedule - that is, the schedule will show that the transaction is suspended.
- See following slides for more details.

# Correction #1

T1	T2
Lock_X(B)	
Lock_X(F)	
Read(B)	Lock_S(F)
B = B*10	Read(F)
Write(B)	Lock_S(B)
F = B*100	
Write(F)	
Unlock(F)	
Unlock(B)	
	Read(B)
	Print(F+B)
	Unlock(B)
	Unlock(F)

- The Read(F) and Lock\_S(B) (highlighted in red) will not occur until after T1 unlocks F.

# Correction #1

T1	T2
Lock_X(B)	
Lock_X(F)	
Read(B)	Lock_S(F)
$B = B * 10$	
Write(B)	
$F = B * 100$	
Write(F)	
Unlock(F)	
Unlock(B)	
↓	Read(F)
	Lock_S(B)
	Read(B)
	Print(F+B)
	Unlock(B)
	Unlock(F)

If we put Unlock(B) here it would be 2PL →

- This is what the previous schedule *should* look like. Every action waits as it should.
- As before, this schedule is in Strict 2PL and will not result in deadlocks.
- Incorrect: I had previously stated that this schedule would be serializable but not conflict/view serializable.
- It is in fact both conflict and view serializable, which can be verified with the usual methods.

# Correction #1

T1	T2
Lock_X(B)	
Read(B)	Lock_S(F)
$B = B * 10$	Read(F)
Write(B)	Lock_S(B)
Lock_X(F)	Unlock(F)
$F = B * 100$	
Write(F)	
Unlock(F)	
Unlock(B)	
	Read(B)
	Print(F+B)
	Unlock(B)

- Revisiting an earlier example, this schedule results in a deadlock, and consequently everything afterwards (highlighted in red) does not make sense if we honor the 2PL protocol.

# Correction #2




- In discussion I made a distinction between serializability and view serializability.
- Incorrect: The example I gave for this was that the database does not know a transaction will need to wait. This is not true as the schedule accounts for the waiting.
- Correct: For the purposes of this class, serializability and view serializability are the same. A more nuanced definition is that a schedule *could* be serializable but not view serializable, but this is a technicality and not practically relevant (i.e. are the non-read/write actions interchangeable? this requires human interpretation).

- For reference, here are the original slides (much of it is incorrect).

The schedule is incorrect, but the checks are correct.

# Worksheet - 3

T1	T2
Lock_X(B)	
Read(B)	Lock_S(F)
$B = B * 10$	Read(F)
Write(B)	Lock_S(B)
Lock_X(F)	Unlock(F)
$F = B * 100$	
Write(F)	
Unlock(F)	
Unlock(B)	
	Read(B)
	Print(F+B)
	Unlock(B)

- 2PL? 
- Strict 2PL? 
- No deadlock? 

The schedule is incorrect, but the checks are correct.

# Worksheet - 3

T1	T2
Lock_X(B)	
Read(B)	Lock_S(F)
$B = B * 10$	Read(F)
Write(B)	Lock_S(B)
Lock_X(F)	
$F = B * 100$	
Write(F)	
Unlock(F)	
Unlock(B)	
	Read(B)
	Print(F+B)
	Unlock(B)
	Unlock(F)

- 2PL? ✓
- Strict 2PL? ✓
- No deadlock? ✗



The schedule is incorrect, but the checks are correct.

# Worksheet - 3

T1	T2
Lock_X(B)	
Lock_X(F)	
Read(B)	Lock_S(F)
$B = B * 10$	Read(F)
Write(B)	Lock_S(B)
$F = B * 100$	
Write(F)	
Unlock(F)	
Unlock(B)	
	Read(B)
	Print(F+B)
	Unlock(B)
	Unlock(F)

- 2PL? ✓
- Strict 2PL? ✓
- No deadlock? ✓

None of this is correct - disregard.

# Worksheet - 3

T1	T2
Lock_X(B)	
Lock_X(F)	
Read(B)	Lock_S(F)
$B = B * 10$	Read(F)
Write(B)	Lock_S(B)
$F = B * 100$	
Write(F)	
Unlock(F)	
Unlock(B)	
	Read(B)
	Print(F+B)
	Unlock(B)
	Unlock(F)

- Conflict serializable?



- View serializable?



- Serializable?

