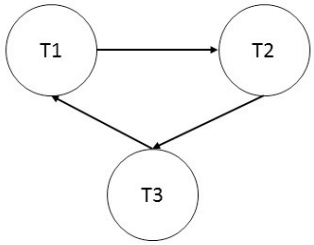
Question 1

Schedule 1

Precedence Graph

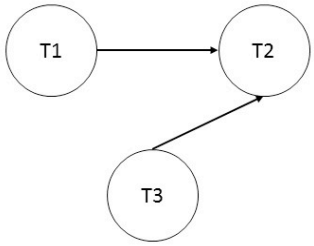


The presence of a cycle shows that the schedule is not conflict-serializable, since there are circular data dependencies among the transactions.

As the schedule is not conflict-serializable, there is no 2PL schedule that could have generated it.

Schedule 2

Precedence Graph



The absence of a cycle shows that the schedule is conflict-serializable. The schedule is conflict-equivalent, e.g., to the serial schedule [T1,T3,T2].

Since the schedule is conflict-serializable, we can now check whether a strict 2PL scheduler could have generated it. This turns out to be the case, as illustrated by the following locking sequence (S shared, X exclusive):

Transaction	Action	Resource
T1	Acquire S lock	X
T3	Acquire X lock	Z
T3	Release X lock (on commit)	Z
T2	Acquire S lock (succeeds, since lock released above)	Z

T1	Acquire X lock	Y
T1	Release S, X locks (on commit)	X, Y
T2	Acquire X lock (succeeds, since lock released above)	X
T2	Acquire X lock (succeeds, since lock released above)	Y
T2	Release X locks (on commit)	X, Y

Question 2

Scenario 1

T3 is rolled back. The issue is that the write set of T2 intersects the read set of T3 (object 4). Since T2 completes before T3 begins with its write phase, there is no overlap in the write phases and thus write-write conflicts are immaterial. However, there is potential overlap between T2's write phase and T3's read phase, and we must check for write-read conflicts. Since the whole of T1 precedes T3's read phase, no conflicts are possible between these two transactions.

Scenario 2

T3 is rolled back. Similarly to the argument above, we need to check for write-read conflicts between T1 and T3, since T1 completes before T3 begins with its write phase. The write set of T1 intersects the read set of T3 (object 3), leading to the abort decision. For T2 and T3, there could be potential overlap between their write phases in addition to the write phase of T2 and the read phase of T3, since T2 completes its read phase before T3 does. Thus, we need to check for both WR and WW conflicts. No conflict exists, however, since the write set of T2 intersects neither the read nor the write sets of T3.

Scenario 3

T3 commits. By the argument above, we must check for intersection of the write sets of T1 and T2 with the read set of T3. Both intersections are empty, and thus there are no WR conflicts.