Exercises

16.1 Show that the two-phase locking protocol ensures conflict serializability, and that transactions can be serialized according to their lock points.
Answer:

16.2 Consider the following two transactions:

```
T31: read(A);
     read(B);
     if A = 0 then B := B + 1;
     write(B).

T32: read(B);
     read(A);
     if B = 0 then A := A + 1;
     write(A).
```

Add lock and unlock instructions to transactions T_{31} and T_{32}, so that they observe the two-phase locking protocol. Can the execution of these transactions result in a deadlock?
Answer:

a. Lock and unlock instructions:

```
T31: lock-S(A)
     read(A)
     lock-X(B)
     read(B)
     if A = 0
     then B := B + 1
     write(B)
     unlock(A)
     unlock(B)

T32: lock-S(B)
     read(B)
     lock-X(A)
     read(A)
     if B = 0
     then A := A + 1
     write(A)
     unlock(B)
     unlock(A)
```

b. Execution of these transactions can result in deadlock. For example, consider the following partial schedule:

T_{31}	T_{32}
lock-S(A)	
	lock-S(B)
	read(B)
read(A)	
lock-X(B)	
	lock-X(A)

The transactions are now deadlocked.

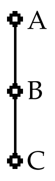
16.3 What benefit does strict two-phase locking provide? What disadvantages result?
Answer: Because it produces only cascadeless schedules, recovery is very easy. But the set of schedules obtainable is a subset of those obtainable from plain two phase locking, thus concurrency is reduced.

16.4 What benefit does rigorous two-phase locking provide? How does it compare with other forms of two-phase locking?
Answer: Rigorous two-phase locking has the advantages of strict 2PL. In addition it has the property that for two conflicting transactions, their commit order is their serializability order. In some systems users might expect this behavior.

16.5 Most implementations of database systems use strict two-phase locking. Suggest three reasons for the popularity of this protocol.
Answer: It is relatively simple to implement, imposes low rollback overhead because of cascadeless schedules, and usually allows an acceptable level of concurrency.

16.6 Consider a database organized in the form of a rooted tree. Suppose that we insert a dummy vertex between each pair of vertices. Show that, if we follow the tree protocol on the new tree, we get better concurrency than if we follow the tree protocol on the original tree.
Answer: The proof is in Buckley and Silberschatz, "Concurrency Control in Graph Protocols by Using Edge Locks," Proc. ACM SIGACT-SIGMOD Symposium on the Principles of Database Systems, 1984.

16.7 Show by example that there are schedules possible under the tree protocol that are not possible under the two-phase locking protocol, and vice versa.
Answer: Consider the tree-structured database graph given below.



Schedule possible under tree protocol but not under 2PL:

T_1	T_2
lock(A)	
lock(B)	
unlock(A)	
	lock(A)
lock(C)	
unlock(B)	
	lock(B)
	unlock(A)
	unlock(B)
unlock(C)	

Schedule possible under 2PL but not under tree protocol:

T_1	T_2
lock(A)	
	lock(B)
lock(C)	
	unlock(B)
unlock(A)	
unlock(C)	

16.8 Consider the following extension to the tree-locking protocol, which allows both shared and exclusive locks:

- A transaction can be either a read-only transaction, in which case it can request only shared locks, or an update transaction, in which case it can request only exclusive locks.
- Each transaction must follow the rules of the tree protocol. Read-only transactions may lock any data item first, whereas update transactions must lock the root first.

Show that the protocol ensures serializability and deadlock freedom.
Answer: The proof is in Kedem and Silberschatz, "Locking Protocols: From Exclusive to Shared Locks," JACM Vol. 30, 4, 1983.

16.9 Consider the following graph-based locking protocol, which allows only exclusive lock modes, and which operates on data graphs that are in the form of a rooted directed acyclic graph.

- A transaction can lock any vertex first.
- To lock any other vertex, the transaction must be holding a lock on the majority of the parents of that vertex.

Show that the protocol ensures serializability and deadlock freedom.
Answer: The proof is in Kedem and Silberschatz, "Controlling Concurrency Using Locking Protocols," Proc. Annual IEEE Symposium on Foundations of Computer Science, 1979.

16.10 Consider the following graph-based locking protocol that allows only exclusive lock modes, and that operates on data graphs that are in the form of a rooted directed acyclic graph.