12.7

证：假设两阶段封锁协议不能保证冲突可串行化，则知优先图中存在循环设为 $T_0 \to T_1 \to \cdots \to T_{n-1} \to T_0$

设 t_i 为最后获得锁时间，则对

$T_i \to T_j$，必有 $t_i < t_j$，则有 $t_0 < t_1 < \cdots t_{n-1}$

又由 $T_0 \to T_1 \to \cdots \to T_{n-1} \to T_0$

则 $t_0 < t_{n-1}$

∴ 矛盾，

∴ 两阶段封锁协议必可得到冲突可串行化

又∵ 事务封锁顺序也是拓扑顺序（否则必循环）

∴ 必可根据封锁点串行化

12.8

会引起死锁，由于 T_{34} 与 T_{35} 首先处理的对象不同，若两事务分别获得 A 与 B 的锁，就会引起死锁.

12.16

A: Atomicity, 原子性, 事务的所有操作在数据库中要么全部正确反映, 要么全部不反映, 维持一致性.

C: consistency 一致性, 隔离执行事务时保持数据库一致性

I: Isolation 隔离性, 确保事务正常执行而不被来自并发执行的数据库语句干扰

D: Durability 持久性, 事务成功完全后, 即使系统出故障, 其对数据库影响也会持续存在

12.19

a. A=B=0, 无论是 $T_{13} \to T_{14}$ 还是 $T_{14} \to T_{13}$, 其最后的结果都是一样的, 即串行执行后的结果必为0, 1. 满足一致性需求
即每一个串行执行都保持数据库一致性

b.

T_{13}	T_{14}
read(A)	
read(B)	
	read(B)
	read(A)
if A=0 then B=B+1	
write(B)	
	if B=0 then A=A+1
	write(A)

C、不存在.

事务若想对某个值执行write，必须先执行read、要想得串行化结果，需在其中一个事务执行最后一条 write 再执行另一个，即串行.

∴不存在

13、 a.

T_3	T_4	T_6	$R\text{-}TS(Q)$	$W\text{-}TS(Q)$
Read(Q)			1	0
	write(Q)		1	2
write(Q)			1	2
		write(Q)	1	3
Read(Q)			4	3
write(Q)			4	4

T_3 rollback

根据时间戳协议，T3需 roll back.

b.

T_3	T_4	T_6	$R\text{-}TS(Q)$	$W\text{-}TS(Q)$
			0	0
Read(Q)			1	0
	write(Q)		1	2
write(Q)			1	2
		write(Q)	1	3

忽略

① T_3 中的 write(Q) 操作已过时，按时间戳排序协议需 roll back，并赋予其新的时间戳重新启动.

② 而按 Thomas 写规则，系统将直接忽略