16337341(朱志儒)数据库系统作业 6

14.6

由于优先图中不包含环，所以相应的调度是冲突可串行化的，串行化顺序可由拓扑排序得到，即T_1, T_2, T_3, T_4, T_5。

14.15

a.

两个事务的的串行执行分为两种情况，如下：

第一种情况：

	A	B
初始值	0	0
执行T_{13}后	0	1
执行T_{14}后	0	1

第二种情况：

	A	B
初始值	0	0
执行T_{14}后	1	0
执行T_{13}后	1	0

从上表可以看到这两个事务的每一个串行执行都保持数据库的一致性($A = 0 \vee B = 0$)。

b.

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

c.

不存在可串行化调度的T_{13}和T_{14}的并发执行。假设T_{14}的 read(B);首先执行，在T_{14}事务 write(A);执行前的任意位置调度T_{13}事务的 read(A);执行，那么T_{13}中读取的 A=0，同时T_{14}中读取的 B=0，因此，T_{13}和T_{14}均执行完毕后，A=1，B=1，此时数据库不能保持一致性。所以，T_{13}事务的 read(A);必须在T_{14}事务 write(A);完成后执行。故不存在可串行化调度的T_{13}和T_{14}的并发执行。

14.17

可恢复调度：对于每对事物T_i和T_j，如果T_j读取了之前由T_i所写的数据项，则T_i先于T_j提交。

要求调度的可恢复性的原因：一个事务的失败可能导致数据库不能保持一致性。

允许出现不可恢复调度的情况：对于一个长时间的事务，即使尚未提交，也可能需要尽快使更新可见，这时就需要不可恢复调度。

14.19

已提交读隔离性是指只允许读取已提交的数据，也就是说，对于每对事务T_i和T_j，如果T_j需要读取T_i所写的数据项，则T_j必须等到T_i提交后才能读取，这满足无极联调度的条件，所以已提交读隔离性保证调度是无极联的。

15.2

这两个事务会引起死锁，调度如下：

T_{34}	T_{35}
Lock-S(A);	
Lock-S(B);	
	Lock-S(B);
	Lock-S(A);
Read(A);	
Read(B);	
	Read(B);
	Read(A);
If $A = 0$ then $B := B + 1$	
	If $B = 0$ then $A := A + 1$
Upgrade(B);	
	Upgrade (A);

按照上述的调度，T_{34}等待T_{35}在 B 上共享锁的释放，T_{35}等待T_{34}在 A 上共享锁的释放，但他们双方并不会释放各自的共享锁，所以将会发生死锁。

15.21

严格两阶段封锁协议流行的三点理由：

1. 保证可串行性
2. 避免联级回滚
3. 实现较为简单

15.27

因为父节点持有 SIX 或 S 锁，所以该父节点的所有后代节点都隐式地加上了 S 锁或 IS 锁，故协议不允许当父节点持有 SIX 或 S 锁时对该节点加 S 或 IS 锁。

16.1

undo-list 中事务的记录必需由后往前进行处理，因为如果由前往后进行处理的话，将出现错误，例如，undo-list 中事务 T 将数据项 A 从 1 变为 2，再从 2 变为 3，如果由前往后进行处理，最后 A 将变为 2，与实际不符，如果由后往前进行处理，最后 A 将变为 1，与实际相符，所以 undo-list 中事务的记录必需由后往前进行处理。

redo-list 中事务的记录必需由前往后进行处理，因为如果由后往前进行处理的话，将出现错误，如上述的例子，如果由后往前进行处理，最后 A 将变为 2，与实际不符，如果由前往后进行处理，最后 A 将变为 3，与实际相符，所以 redo-list 中事务的记录必需由前往后进行处理。

16.18

对于事务T_0，日志中不存在$< T_0$ abort $>$日志记录，也不存在$< T_0$ commit $>$日志记录，所以执行undo(T_0)，即将事务T_0更新过的所有数据项的值都恢复为旧值，则 $B = 2000$。

对于事务T_1，日志中存在$< T_1$ commit $>$日志记录，所以执行 redo(T_1)，即将事务T_1更新过的所有数据项的值都设置为新值，则 $C = 600$。

对于事务T_2，日志中不存在$< T_2$ abort $>$日志记录，也不存在$< T_2$ commit $>$日志记录，所以执行undo(T_2)，即将事务T_2更新过的所有数据项的值都恢复为旧值，则 $A = 500$。

16.22

a. 某个页面不在检查点脏页表中，说明该页面已经存入磁盘中，则在检查点前的 redo 记录不需要应用于该页，但是在检查点后，该页面可能会被更新，那么在分析阶段快结束时该页面有可能在脏页表中，部分 redo 记录可能会应用于该页。

b. RecLSN 是脏页表中反映页面被加载到脏页表中时日志末端的 LSN，它大于或等于稳定存储器中该页的 PageLSN。在分析阶段，将 RedoLSN 设置为脏页表中页的 RecLSN 的最小值；在重做阶段，从 RedoLSN 开始正向扫面日志，找到一个更新日志记录，如果该更

新日志记录的 LSN 小于脏页表中的该页的 RecLSN，重做阶段就跳过该日志记录。