数据库 HW8

王嵘晟　PB1711614

1.

以两个并发事务 T_1, T_2 为例，多个并发事务时同理。当出现以下两种情况时，不可串行化：

(1) 事务 T_1 写某个数据对象 A，T_2 读或写 A;
(2) T_1 读某个数据对象 A，T_2 写 A;

设 T_1, T_2 的潜在冲突公共对象为 $A_1, A_2, ..., A_n$，其中 $X = \{A_1, ...A_j\}$ 符合 (1)，$Y = \{A_{j+1}, ..., A_n\}$ 符合 (2)。所以此时 T_1 需要用 X 锁 ①，而 T_2 用 S 锁或 X 锁 ②。假设 ① 先执行，则 T_1 获得锁，T_2 等待。由 2PL，T_1 在全部获得 X 与 Y 中的对象的锁后才会开始释放锁。如果此时对于 $A_i, 1 \leq i \leq n$，T_2 获得了 A_i 的锁，则会发生死锁。否则只有当 T_1 全部执行完了对于 X,Y 的事务才会释放锁，让等待的 T_2 开始执行。这使得 T_1, T_2 可以顺序串行执行，则 T_1, T_2 的调度是冲突可串的，因而是可串化调度。对于先执行 ② 同理可证。因此若并发事务遵循 2PL，在不发生死锁的情况下，对这些事务的并发调度一定是可串行化的调度。

2.

假设两个事物 T_1, T_2，对于一个数据对象 A，T_1 取得了 S 锁，T_2 取得了 U 锁，则 T_1 只读 A，T_2 拿到了 U 锁可以升级到 X 锁对 A 做写操作，这样有良好的并发性。而当 T_1 取得了 A 的 U 锁时，若此时 T_2 取得 A 的 S 锁，则会导致 T_1 永远没法升级到 X 锁，无法对 A 进行写操作。这在 DBMS 中是无效的，所以为了避免这种情况，DBMS 设计了不对称的锁相容规则，旨在提高并发性且可以正常对数据库进行事务操作。

3.

对于普通 2PL 会，例如:

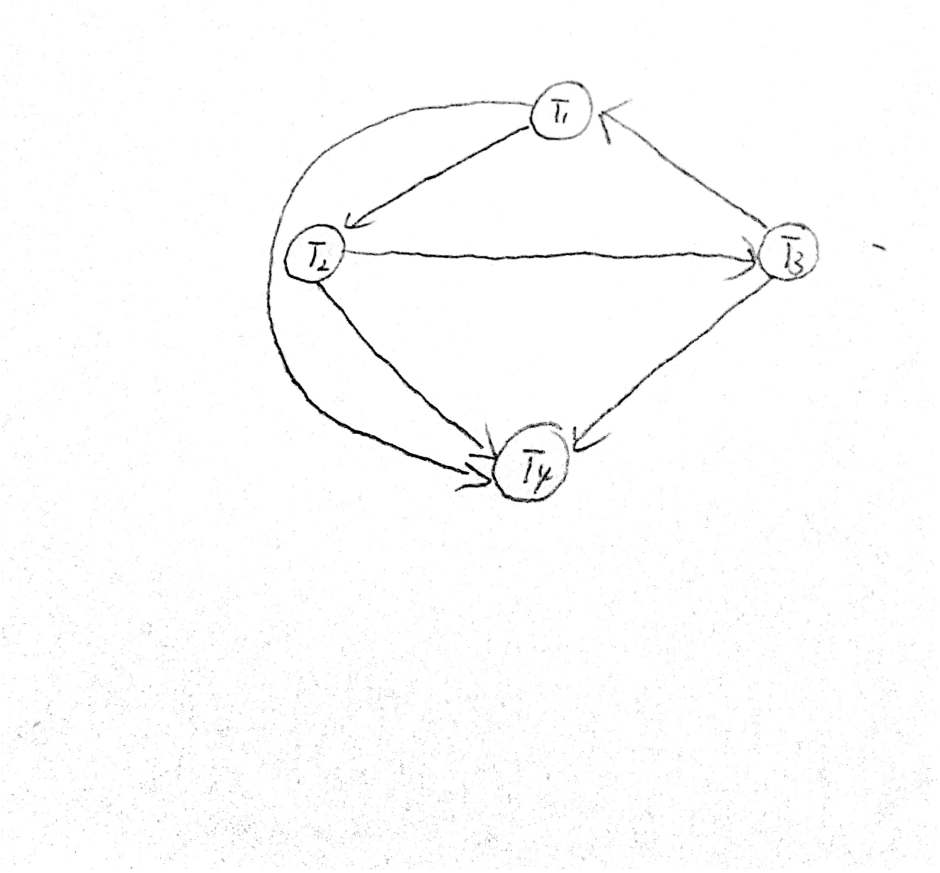
t	T_1	T_2
1	xL1(A)	
2	A=A-100	
3	xL1(B)	
4	Write(A)	
5	xL1(B)	
6	Unlock(A)	
7		sL2(A)
8	B=B+100	Read(A)
9	Write(B)	
10	Unlock(B)	
11	Rollback	

满足 2PL，但由于最后 T_1 Rollback，导致 T_2 对 A 的 Read 是脏读

但对于严格 2PL，X 锁会保持到事务结束，当 T_1 对事务 A 持有 X 锁时，T_2 不能持有 A 的任何锁，所以此时不会出现脏读，强 2PL 同理不会出现脏读。

4.

对于给定的操作串，其优先图如下:



由于优先图中有环，所以是不可串。

5.

t	T_1	T_2	T_3	T_4
1			xL3(x)	
2			write(x)	xL4(y)
3	xL1(z)			write(y)
4	write(z)		xL3(a)	u4(y)
5			write(a)	
6			xL3(y)	
7			write(y)	
8		xL2(a)	write(x)	
9		wait	write(y)	
10		wait	u3(x)	
11	xL1(a)	wait	u3(a)	
12	wait	write(a)	u3(y)	
13	wait	xL2(z)	write(y)	
14	xL1(z)	wait		
15	wait	wait		
16	wait	wait		

这时根据 2PL，T_1 等待 T_2 持有的 a 的锁，而 T_2 等待 T_1 持有的 z 的锁，出现了死锁情况，所以不能用用 2PL 调度器生成并发调度。