第六次作业答案:

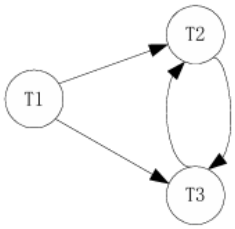
习题 8.2（1）:

8.2 考虑下面的（不完全）调度 S：$R_1(X), R_1(Y), W_1(X), R_2(Y), W_3(Y), W_1(X), R_2(Y)$

（1） 假定三个事务最终都正常提交，请画出它们的优先图。S 是否为可串行化调度？如是，请进一步给出与它等价的串行调度。

【解答】

（1） 其优先图如右图所示。因为其优先图有环，所以，调度 S 不能肯定是可串行化调度。实际上，因 $R_2(Y), W_3(Y)$不可交换，它也不是一个冲突可串行化调度。但它确实一个(视)可串行化调度，其等价的串行串行调度为:



$R_1(X), R_1(Y), W_1(X), W_1(X), R_2(Y), R_2(Y), W_3(Y)$

习题 8.4（1）~（4）

8.4 考虑以下动作序列，假设它们按此顺序依次提交 DBMS。

S1: R1(X), W2(X), W2(Y), W3(Y), W1(Y), COMMIT(T1), COMMIT(T2), COMMIT(T3)

S2: R1(X), W2(Y), W2(X), W3(Y), W1(Y), COMMIT(T1), COMMIT(T2), COMMIT(T3)

对每个序列和每个以下并发控制机制，描述相应并发控制机制将如何控制序列。假设事务 Ti 的时间戳为 i。对于封锁并发控制机制，添加相关 DB 元素的封锁和解锁请求。如果一个事务阻塞，则假设它的所有动作将进入等待队列直至等待条件满足，而 DBMS 可继续执行其它未阻塞事务的动作。

（1） 具有死锁检测特性的 strict-2PL;

（2） 保守-2PL;

（3） 基于确认的优化并发控制;

（4） 时间戳并发控制;

【解答】

（1）使用基于等待图死锁检测的strict-2PL。事务允许等待，不会因等待而被中止；除非检测到死锁，而且事务被选中为牺牲事务。

对序列S1: T1获得X的共享锁；T2等待X的排它锁；T3获得Y的排它锁；T1等待Y的排它锁；T3结束释放Y的锁；T1被唤醒，获得Y的排它锁；T1执行结束释放X,Y上的锁；T2获得X和Y的排它锁，T2执行完成、提交释放所拥有的所有锁。

对序列S2:T1获得X的共享锁；T2获得Y的排它锁，T2等待X的排它锁；T3等待Y的排它锁；T1等待Y的排它锁。这时，发生了死锁：T1 等待 T2, 而T2 又等待T1。

（2）使用保守-2PL。调度安排很简单：

对序列S1和S2都是：T1获得X和Y的锁后，开始执行，完成提交后释放X,Y上的锁；然后是T2获得所有需要锁，执行/完成/提交/释放所有的锁；最后是T3获得所有需要锁，执行/完成/提交/释放所有的锁。

（3）基于确认的优化并发控制。

调度序列中的每个事务都将执行，从DB读相关元素的值并写到私有空间；它们然后获得一备个时间戳进入确认阶段。事务T_i的时间戳值为i。

对调度序列S1/S2：因T1具有最早时间戳，它的确认不会有任何问题。但当T2准备有效确认时，因其写元素集和T1的读元素集有交集，而T1尚未完成，故T2无法通过有效确认，将被中止并在稍后重启。同样，T3也不能通过有效确认。直到T1完成之前，T2可能会被反复中止/重启多次。

而T3则要等到T2完成之后重启才能通过有效确认。

（4）多版本时间戳控制。

对序列S1:T1读X，RTS(X)=1;T2允许写X，因为TS(T2)>RTS(X)。而后，RTS(X) 和 WTS(X)被设置为2；因为TS(T3)>RTS(X)，T3也能写X；之后，将RTS(X) 和 WTS(X)被设置为3。

现在T1准备写Y，因为TS(T1)<RTS(Y),T1需要被中止并在稍后重启。

序列S2的情况类似于S1.