Problem 1

Winter 2016 #4

(a) Consider a concurrency control manager that uses strict two phase locking that schedules three transactions. For the following transactions, you can assume that all relevant locks are acquired at the beginning of the transaction (eg, T1 acquires locks on A and B at the beginning).

- *T1 : R1(A), R1(B), W1(A), W1(B), Co1*
- *T2 : R2(B), W2(B), R2(C), W2(C), Co2*
- *T3 : R3(C), W3(C), R3(A), W3(A), Co3*

Each transaction begins with its first read operation, and commits with the *Co* statement. Answer the following questions for each of the schedules below:

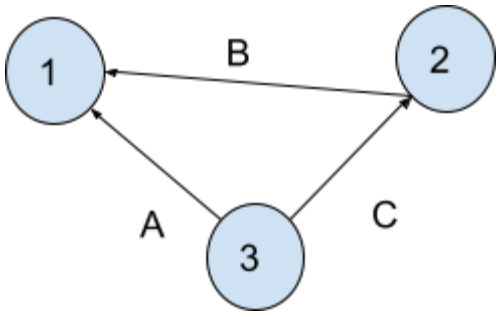
- All relevant locks are acquired at the beginning of the transaction (eg, T1 acquires locks on A and B at the beginning).
- Is the schedule conflict-serializable? If yes, indicate a serialization order.
- Is this schedule possible under a strict 2PL protocol?
- If strict 2PL does not allow this schedule because it denies a read or a write request, is the system in a deadlock at the time when the request is denied?

Note: Common Question if Conflict Serializable -/-> strict 2PL

i. Schedule 1:

R2(B), W2(B), R3(C), W3(C), R3(A), W3(A), Co3, R2(C), W2(C), Co2, R1(A), R1(B), W1(A), W1(B), Co1

α) Is this schedule conflict-serializable? If yes, indicate a serialization order.



Solution: yes: 3,2,1

β) Is it possible under strict 2PL

Solution: Yes

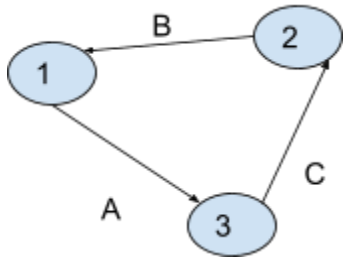
γ) Does strict 2PL lead to a deadlock?

Solution: No

ii. Schedule 2:

R2(B), W2(B), R3(C), W3(C), R1(A), R1(B), W1(A), W1(B), Co1, R2(C), W2(C), Co2, R3(A), W3(A), Co3

α) Is this schedule conflict-serializable? If yes, indicate a serialization order.



Solution: no

β) Is it possible under strict 2PL?

Solution: no. T2 holds L(B) thus R1(B) is not possible before Co2.

γ) Does strict 2PL lead to a deadlock?

Solution: yes: T1 holds L(A), T2 holds L(B), T3 holds L(C) and none can make progress.

(b) Consider the following three transactions:

- *T1 : R1(A), W1(B), Co1*
- *T2 : R2(B), W2(C), Co2*
- *T3 : R3(C), W3(D), Co3*

Give an example of a conflict-serializable schedule that has the following properties: transaction T1 commits before transaction T3 starts, and the equivalent serial order is T3, T2, T1.

Solution: R1(A), R2(B), W1(B), Co1, R3(C), W2(C), Co2, W3(D), Co3
Variations include: swap the first two reads (of A and B), and the last two writes (of C and D, together with the commit order)

(c) A read-only transaction is a transaction that only reads from the database, without writing/inserting deleting. Answer the questions below by circling the correct answer.

i. If all transactions are read-only, then every schedule is serializable.

TRUE or FALSE

ii. Only one transaction can hold a shared lock on the same item at any time.

TRUE or **FALSE**

iii. Only one transaction can hold an exclusive lock on the same item at any time.

TRUE or FALSE

Problem 2

Autumn 2016 #4
Given the following three transactions:

```
T1: R(A), W(B), I(D), R(C)
T2: R(B), R(D), W(C)
T3: R(D), R(C), R(D), W(A)
```

Assume that `R(X)` reads all tuples in table `X`, `W(X)` updates all tuples in `X`, and `I(X)` inserts one new tuple in `X`. `Co` and `Ab` mean commit and abort, respectively.

a-c) SKIPPED

d) Does there exist a schedule of the above transactions that would result in a deadlock if executed under strict 2PL with **both shared and exclusive table locks**? If so write such a schedule with lock / unlock ops, and explain why the transactions are deadlocked. Otherwise write "No". Use `L1(A)` to refer to `T1` locking table `A`, and `U1(A)` for unlocking.
Use S1(A) for grabbing shared lock, and X1(A) for exclusive lock.

```
S1(A); R1(A); X1(B); W1(B); S3(D); R3(D); S3(C); R3(C); R3(D);
X1(D); X3(A); <deadlock>
```

T1 becomes blocked on D at X1(D) (the underlined transaction) because T3 is already holding (a shared) lock on D.

The deadlock happens when T3 attempts to acquire a(n exclusive) lock on A, which is currently held by the already-blocked T1.

e) Does there exist a schedule of the above transactions that would result in a deadlock if executed under strict 2PL with **only exclusive table locks**? If so write such a schedule with lock and unlock operations and indicate why the transactions are deadlocked. Otherwise write "No". Use `L1(A)` to refer to `T1` locking table `A`, and `U1(A)` for unlocking.

```
L1(A); R1(A); L2(B); R2(B); L3(D); R3(D); L3(C); R3(C); <deadlock>
```
T1 holds lock on A and is waiting for lock on B, which is held by T2
T2 holds lock on B and is waiting for lock on D, which is held by T3
T3 holds lock on D and is waiting for lock on A, which is held by T1

To get full points, students will need to show both a schedule, explain which transaction holds which lock, and how that leads to a deadlock.

f-g) SKIPPED