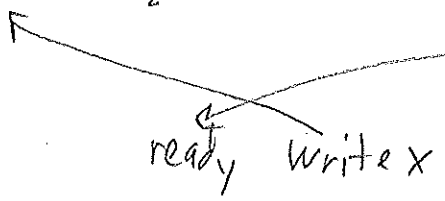
OS HW #4 Ch. 5 Exercises 5.4, 5.5, 5.8, 5.11, 5.17

1.25 .75 .5 1 1.5 = 5

5.4 (a) T_1: read x write z read v write y

T_2: read y write x

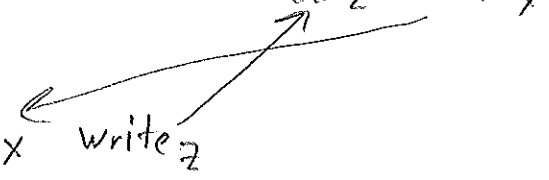


- Moving T_2 before T_1 causes "forbidden swaps: $r_1(x), w_2(x)$, etc.
- moving T_2 after T_1 cause "forbidden swaps: $r_2(y), w_1(y)$

So not serializable.

(b) T_1: read v

T_2: read z write y read x write z read v write y



Serializable if T_2 and then T_1

(c) Serializable since only does reads and read locks, so no forbidden transactions. Plus, transactions obey two-phase locking.

Serial order T_1 then T_2 or T_2 then T_1 are same

(d) T_1: write x write y

T_2: write x write z

T_3: write z write y

T_2 and T_3 are serial already, but their order must be fixed as T_2 then T_3 because of the write on z.

T_2 can move in front of T_1 due to write on x and T_2-T_3 cannot move after T_1 due to T_3's and T_1's write on y.

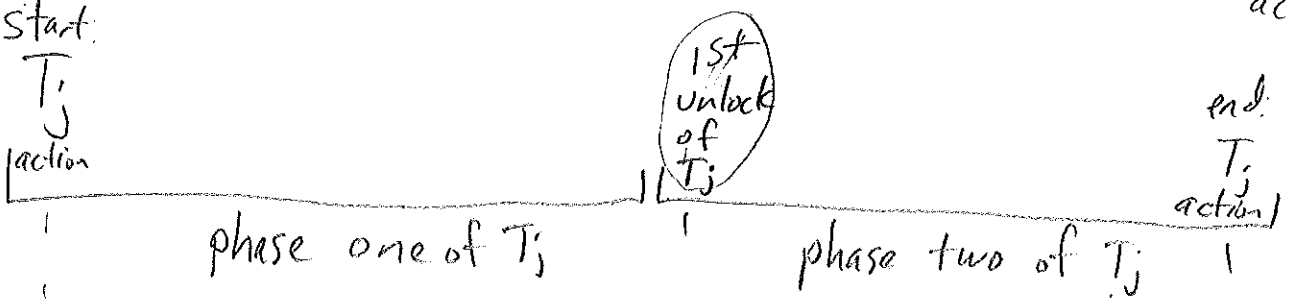
Thus, Not serializable

(e) All transactions follow two-phase locking so serializable.

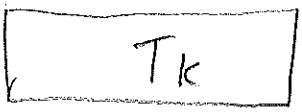
Serial Order: T_2, T_1, T_3

5.5. Histories (c) and (e) obey two-phase locking.

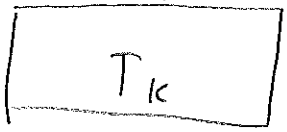
5.8 Consider T_j's phase two which starts with its 1st unlock action



Recall that T_K being pure means that no actions from T_j occur within T_K.

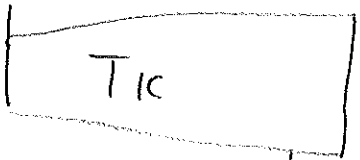


can not straddle the starting action of T_j or T_K would not be pure



cannot straddle the start of phase two of T_j or T_K would not be pure

.5



T_K cannot straddle the end of T_j or T_K would not be pure

Thus, T_K must be entirely in T_j's phase one or T_j's phase two.

5.11 Clearly an infinite number of examples could be devised by you!

Simple answer: If we process the undo log entries in reverse chronological order, then the state should be restored to when the transaction started.

5.17 T_1 writes x and y

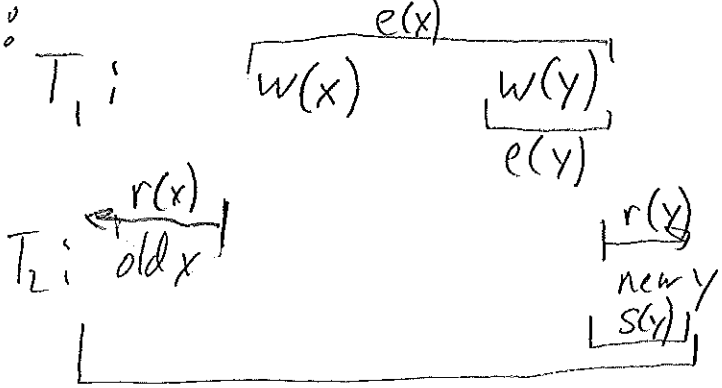
T_2 read x and y

Is it possible for T_2 to see the old value of x, but the new value of y?

<u>With two-phase locking</u> the answer is <u>No</u>.

We don't know the order T_1 writes x and y so we must show neither is possible

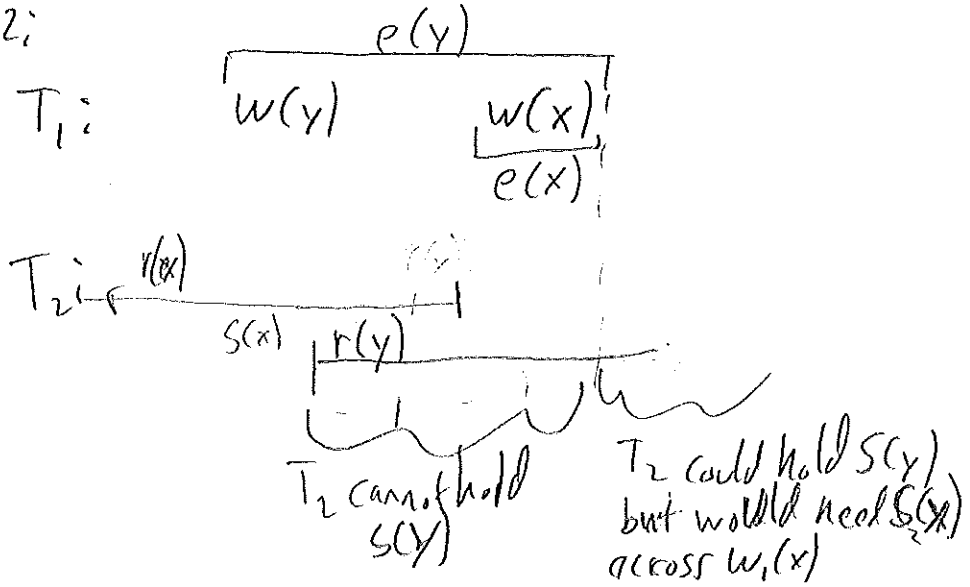
case 1:



T_1 follows two-phase locking

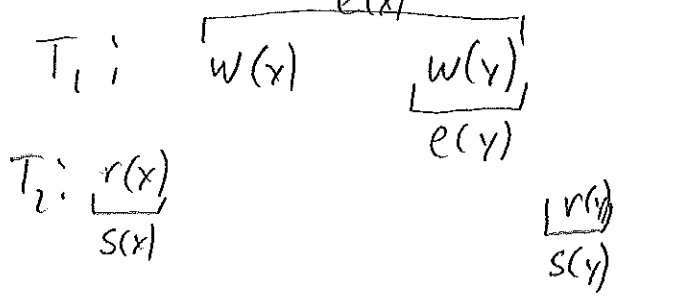
T_2 must hold S(x) or it cannot acquire S(y). Conflict on $S_2(x)$ and $w_1(x)$

case 2:

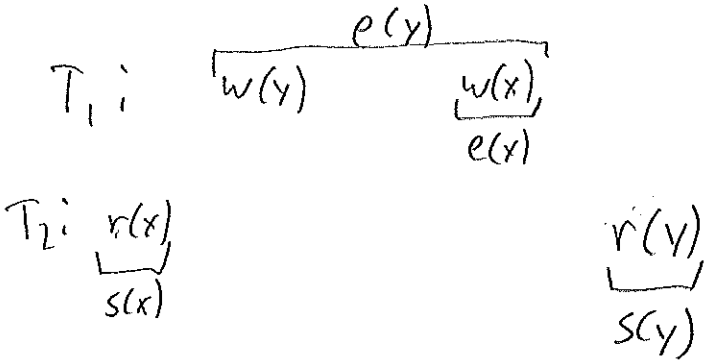


Read committed isolation – acquire shared lock before each read operation and release it immediately after read.

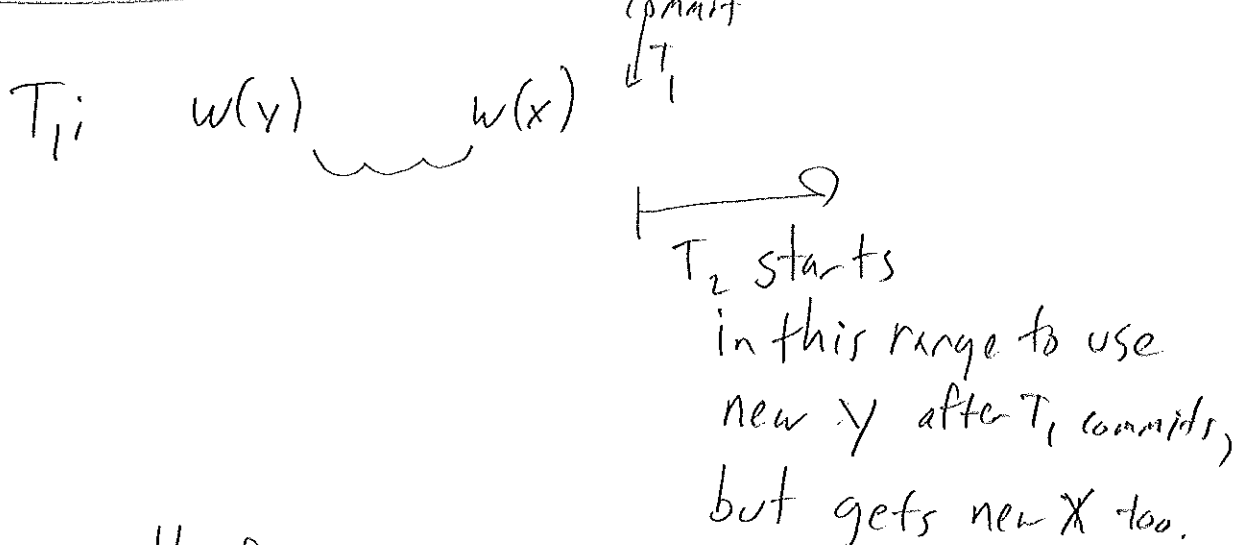
Yes, T_2 could see the old value of x, but new value of y. regardless of order T_1 writes



or



Snapshop isolation:



1.5

No, not possible for T_2 to read new y, but old x.