CS-245 Database System Principles – Winter 2004
Assignment 6

Due at the beginning of class on Thursday, February 26th

- State all assumptions and show all work.
- You can email questions to cs245-staff@cs.stanford.edu

Problem 1 (20 points)

For each of the following schedules:

a) $S_a = r_1(A); r_2(B); w_1(B); w_2(C); r_3(C); w_3(A);$
b) $S_b = r_1(A); r_2(A); r_1(B); r_2(B); r_3(A); r_4(B); w_1(A); w_2(B);$

Answer the following questions:

i. What is the precedence graph for the schedule?
ii. Is the schedule conflict-serializable? If so, what are all the equivalent serial schedules?

Problem 2 (15 points)

In a lock table, the system keeps a "group mode" that records the "strongest" lock type of the transactions that have currently locked an object. In particular, say object *O* is locked in modes M_1, ... M_j and let the group mode of *O* be *GM(O)*. Then, for any possible lock mode N,

- *GM(O) and N are not compatible if and only if there is an M_i element of {M_1, ..., M_j} such that N and M_i are not compatible;*
- *GM(O) and N are compatible if and only if for all M_i element of {M_1, ..., M_j}, N and M_i are compatible.*

When a new lock request arrives, for lock mode *N*, the system can simply check if *N* is compatible with *GM(O)*, instead of checking *N* against all locks currently held on object *O*.
Consider the multiple-granularity locking mechanism (Section 9.6 [18.6]). In each of sub-problems (a) through (e) below, the modes of currently held locks on an object *O* are given. (For instance, in case (a), object *O* is locked in mode *IX* by two transactions and in mode *IS* by two transactions.) For each case, give the group mode, if there is one. (Be careful, some of the cases below are impossible! In those cases, just say there is no group mode and explain why it is so).

a) SIX, IS
b) IX, IS, IX, IS
c) S, IS, IX, SIX
d) S, IS

e) IX, S, IS

Problem 3 (24 points)

Consider the following two transactions:
T1 = w1(C) r1(A) w1(A) r1(B) w1(B);
T2 = r2(B) w2(B) r2(A) w2(A)
Say our scheduler performs exclusive locking only (i.e., no shared locks). For each of the following three instances of transactions T1 and T2 annotated with lock and unlock actions, say whether the annotated transactions:

1. obey two-phase locking,
2. will necessarily result in a conflict serializable schedule (if no deadlock occurs),
3. will necessarily result in a recoverable schedule (if no deadlock occurs),
4. will necessarily result in a schedule that avoids cascading rollback (if no deadlock occurs),
5. will necessarily result in a strict schedule (if no deadlock occurs),
6. will necessarily result in a serial schedule (if no deadlock occurs), and
7. may result in a deadlock.

a)
T1 = **L1**(B) **L1**(C) w1(C) **L1**(A) r1(A) w1(A) r1(B) w1(B) Commit **U1**(A) **U1**(C) **U1**(B)
T2 = **L2**(B) r2(B) w2(B) **L2**(A) r2(A) w2(A) Commit **U2**(A) **U2**(B)
b)
T1 = **L1**(C) **L1**(A) r1(A) w1(C) w1(A) **L1**(B) r1(B) w1(B) **U1**(A) **U1**(C) **U1**(B) Commit
T2 = **L2**(B) r2(B) w2(B) **L2**(A) r2(A) w2(A) Commit **U2**(A) **U2**(B)
c)
T1 = **L1**(C) w1(C) **L1**(A) r1(A) w1(A) **L1**(B) r1(B) w1(B) Commit **U1**(A) **U1**(C) **U1**(B)
T2 = **L2**(B) r2(B) w2(B) **L2**(A) r2(A) w2(A) Commit **U2**(A) **U2**(B)

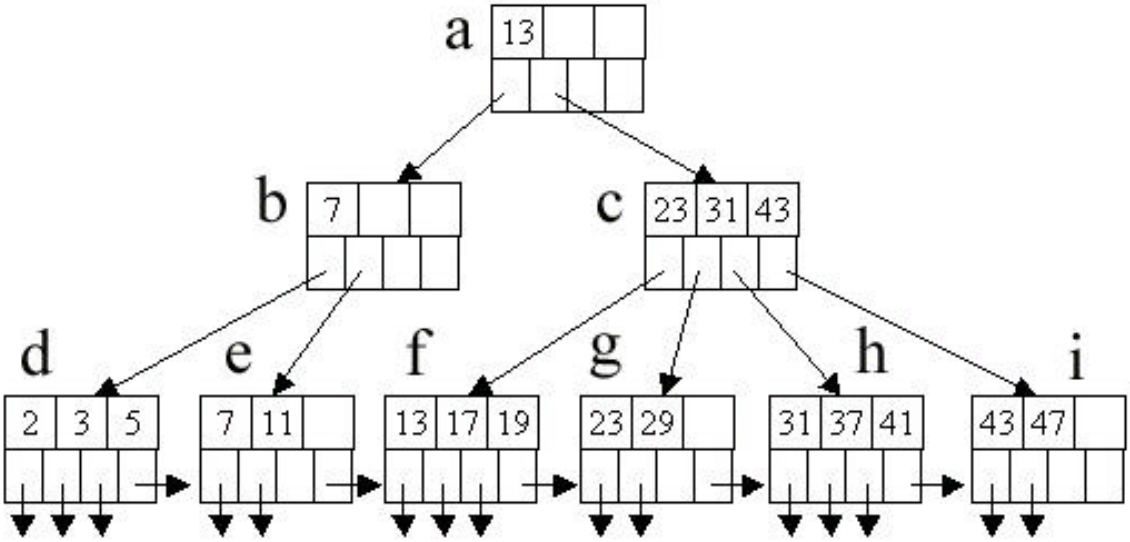
Format your answer in a table with Yes/No entries.

Problem 4 (20 points)

In the following sequences of events, we use $R_i(X)$ to mean "transaction T_i starts, and its read set is the list of the database elements X." Also, V_i means "T_i attempts to validate," and $W_i(X)$ means that "T_i **finishes**, and its write set was X.". State what happens when each sequence is processed by a validation-based scheduler. In particular, state which set intersections are performed for each V_j action and indicate if the validation is successful or not.

a) $R_1(A,B)$; $R_2(B,C)$; $R_3(C)$; V_1; V_2; V_3; $W_1(A)$; $W_2(B)$; $W_3(C)$;
b) $R_1(A,B)$; $R_2(B,C)$; $R_3(B)$; V_1; V_2; $W_1(C)$; V_3; $W_2(B)$; $W_3(C)$;

Problem 5 (20 points)



Suppose we perform the following actions on the B-tree given in the figure. Assume there is only one kind of lock available (i.e., no read/write/update locks). If we use the tree protocol, when can we release the lock on each of the nodes accessed? Note that we would like to unlock a node as early as possible to maximize concurrency. We also would like to maximize throughput; i.e., releasing a higher-level node has priority over releasing a lower level node. Use the notation L(node), UL(node), R(node), W(node) to indicate locking, unlocking, reading and writing a node respectively. Use Create(node) to indicate creation of a new node, if necessary. List the actions in the order they occur, and add short explanations if necessary. E.g., start part a. with the sequence
Lock and read root a: L(a) R(a)
Lock and read node b: L(b) R(b)
…

The actions to be performed:

a. Delete 5 (assume records are borrowed from right sibling if node underfull)
b. Insert 39 (assume overfull nodes are split)

Problem 6 (1 point)
Name a movie star mentioned in the book, or any other movie star.