CPS216 Advanced Database Systems - Fall 2007
Assignment 3, Part 2

- Due date: Dec 4, 2007, 11.59 PM. Late submissions will not be accepted.
- Submission: In class, or email solution in pdf or plain text to shivnath@cs.duke.edu.
- Do not forget to indicate your name on your submission.
- State all assumptions. For questions where descriptive solutions are required, you will be graded both on the correctness and clarity of your reasoning.
- Email questions to shivnath@cs.duke.edu.

Question 1 **Points 10**

Consider a database system with three types of locks: S(shared), I(increment), X(exclusive). We wish to extend the system to handle multiple-granularity locks by adding "intention" locks IS, II and IX. Locks IS and IX are the same as discussed in class. Intention lock II on an object at level i indicates the intention of the lock holder to lock objects at level i + 1 in I mode. Give the compatibility matrix for the proposed scheme.

Question 2 **Points 15**

Schedule S1 is said to be *conflict-equivalent* to schedule S2 if S2 can be derived from S1 by a sequence of swaps of non-conflicting actions. For example, the schedule S1 = r1(A), r2(A), w2(A), w1(A), r2(B), w2(B) is conflict-equivalent to the schedule S2 = r2(A), r1(A), w2(A), r2(B), w1(A), w2(B), since S2 can be derived from S1 as shown below:

```
S1 = r1(A), r2(A), w2(A), w1(A), r2(B), w2(B); swap(r1(A),r2(A))
   = r2(A), r1(A), w2(A), w1(A), r2(B), w2(B); swap(w1(A), r2(B))
S2 = r2(A), r1(A), w2(A), r2(B), w1(A), w2(B)
```

Prove or disprove each of the following statements.

1. If two schedules are conflict equivalent, then their precedence graphs are identical.
2. If two schedules involve the same set of transactions, and have identical precedence graphs, than they are conflict equivalent.

Question 3 **Points 10**

Suppose that we run the following six transactions using the validation protocol. Table 1 lists the read and write sets for each transaction.

The following sequence of events takes place. No other transaction runs before or concurrently with T1, ..., T6.

1. T1, T2, T3, T4 start (in this order)
2. T3 initiates validation

Transaction	Read Set	Write Set
T1	{a,b}	{b,c}
T2	{a,b,c}	{h}
T3	{b}	{d,e}
T4	{c}	{f,g}
T5	{a}	{d,f}
T6	{g}	{e,g}

Table 1: Read and write sets for T1-T6

3. T5, T6 start (in this order)
4. T1 initiates validation
5. T5 initiates validation
6. T4 initiates validation
7. T2 initiates validation
8. T1, T2, T3 finish (if they were not aborted during validation)
9. T6 initiates validation
10. T4, T5, T6 finish (if they were not aborted during validation)

For each transaction write down whether it validates successfully or gets aborted during validation.

Question 4 **Points 14**

A multi-granularity hierarchical locking scheme is used in an object-oriented database. In particular, the objects for a class C_1 are stored in two pages P_1 and P_2. Objects o_1, o_2, and o_3 are stored in Page P_1, while objects o_4 and o_5 are stored in Page P_2. The hierarchy is as shown in Figure 1.

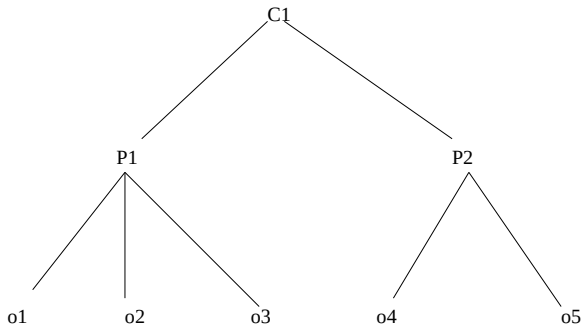

Figure 1: Object hierarchy for tree-based locking

Table 2 shows the state of the system at a particular time when four transactions are active. Each entry identifies the transaction holding a particular lock at this time. For example, Transactions 1 and 2 hold IS locks on class C_1, while Transaction 3 holds an IX lock on C_1. Transaction 4 does not hold any locks at this time.

Using the same table, indicate what are all the possible next lock actions in this scenario. For example, Transaction 3 could next lock object o_5 in X mode, so the cell [o_5, X] should have a "3" in it. This same cell could have another number n if Transaction n could also get this lock. Note that Transactions 3 and n cannot both hold the X lock on o_5; a cell with two or more transactions in your answer will simply mean that any of these transactions could get the corresponding lock next.

Note: Do not forget Transaction 4. Also, do not show entries that are not useful even though they do not create a conflict. For example, it does not make sense for Transaction 1 to request an S or IS lock on o_2 next.

	IS	IX	S	SIX	X
C_1	1,2	3			
P_1	2		1		
o_1			2		
o_2					
o_3					
P_2		3			
o_4					3
o_5					

Table 2: Locks held currently by Transactions 1-4

Question 5 **Points 15**

This question is based on the "fancier" tree-based locking protocol that is presented in Slide 75 of Notes 12. (That is, the one with the "monkey bars" strategy.)

1. Prove or disprove the following statement: if two transactions T_1 and T_2 that follow this protocol lock a set S of nodes in the tree in common, then all nodes in S are either locked by T_1 before any node in S is locked by T_2, or they are locked by T_2 before any of them is locked by T_1.

2. Prove or disprove the following statement: Rule 4 is not needed for conflict-serializability. (Rule 4 in Slide 75 of Notes 12 says that a transaction is not allowed to relock a node after it has unlocked it once.)

3. True or false: Deadlocks can arise even if all transactions follow this protocol. No formal proof is needed for this question; an intuitive argument or example will suffice.