

Fall 2007 CS186 Discussion Section:
Week 14, 11/26 - 11/30

Your friendly TAs

November 25, 2007

1 Transactions and Concurrency Control

1. In general, is it possible to have a deadlock when the regular two-phase-locking (i.e., non-strict) protocol is obeyed? If yes, give an example; if not, explain briefly. What happens with strict 2PL and conservative 2PL?
2. For each of the following schedules:
 - (a) $S_a = r_1(A); w_1(B); r_2(B); w_2(C); r_3(C); w_3(A);$
 - (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:

- (a) What is the precedence graph for the schedule?
 - (b) Is the schedule conflict-serializable? If so, what are all the equivalent serial schedules?
3. Consider the following two transactions: $T_1 = w_1(C) r_1(A) w_1(A) r_1(B) w_1(B); T_2 = r_2(B) w_2(B) r_2(A) w_2(A)$. Say our scheduler performs exclusive locking only (i.e., no shared locks). For each of the following three instances of transactions T_1 and T_2 annotated with lock and unlock actions, say whether the annotated transactions:
 - (a) obey two-phase locking,
 - (b) will necessarily result in a conflict serializable schedule (if no deadlock occurs),
 - (c) will necessarily result in a recoverable schedule (if no deadlock occurs),
 - (d) will necessarily result in a schedule that avoids cascading rollback (if no deadlock occurs),
 - (e) will necessarily result in a strict schedule (if no deadlock occurs),
 - (f) will necessarily result in a serial schedule (if no deadlock occurs), and
 - (g) may result in a deadlock.
 - $T_1 = L_1(C) w_1(C) L_1(A) r_1(A) w_1(A) L_1(B) r_1(B) w_1(B) \textit{Commit} U_1(A) U_1(C) U_1(B)$
 $T_2 = L_2(B) r_2(B) w_2(B) L_2(A) r_2(A) w_2(A) \textit{Commit} U_2(A) U_2(B)$
 - $T_1 = L_1(B) L_1(C) w_1(C) L_1(A) r_1(A) w_1(A) r_1(B) w_1(B) \textit{Commit} U_1(A) U_1(C) U_1(B)$
 $T_2 = L_2(B) r_2(B) w_2(B) L_2(A) r_2(A) w_2(A) \textit{Commit} U_2(A) U_2(B)$
 - $T_1 = L_1(C) L_1(A) w_1(C) r_1(A) w_1(A) L_1(B) r_1(B) w_1(B) U_1(A) U_1(C) U_1(B) \textit{Commit}$
 $T_2 = L_2(B) r_2(B) w_2(B) L_2(A) r_2(A) w_2(A) \textit{Commit} U_2(A) U_2(B)$

	2PL	Necessarily Conflict Serializable	Necessarily Recoverable	Necessarily Avoid Cascading Abort	Necessarily Strict Schedule	Necessarily Serial Schedule	May Result in Deadlock
(a)	Y	Y	Y	Y	Y	N	Y
(b)	Y	Y	Y	Y	Y	Y	N
(c)	Y	Y	N	N	N	Y*	Y

Figure 1: Answer of question 3.

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

T1	T2	T3	T4
R(salary)			R(tax)
	R(tax)		W(tax)
R(tax)	W(tax)		
W(salary)		R(salary)	
W(tax)		W(salary)	
			R(salary)
			W(salary)

Figure 2: Interleaved transactions of question 4.

4. Examine the schedule given below. There are four transactions, T1, T2, T3, and T4.

- (a) Draw the precedence graph for this schedule.
- (b) What is the equivalent serialization order for this schedule? If no order is possible, then state 'none'.
- (c) Assume that transaction T4 did not run at all. What is the precedence graph in this case?
- (d) What is the equivalent serialization order for this second schedule? If no order is possible, then state 'none'.