CS3223: Database Management Systems
Tutorial 9
(Week of 28th March 2022)

1. (On compatibility matrix)
(A) Consider the 2 PL protocol for S/X locks (as described in our lecture slide 64). Upgrading from S to X locks can potentially lead to deadlocks. Illustrate this with an example (Hint: Think of two transactions both reading object A, and both wants to upgrade at the same time). A naïve solution is to simply acquire X lock whenever a transaction wants to read but may (not necessarily) update. This limits the degree of concurrency. An alternative approach is to introduce an UPDATE (U) lock mode. So, the DBMS now supports 3 kinds of locks:
    a. if a transaction reads but do not write an object, then it should acquire an S lock;
    b. if a transaction definitely writes on an object, then it should acquire an X lock;
    c. if a transaction needs to read but may update/write an object, then it acquires an U lock.

How would the compatibility matrix for these locks look like? Fill the table below:

	S	X	U
S	T	F	
X	F	F	
U			

(B) Assume that, in addition to the operations read(x) and write(x), a database has the operation copy(x, y), which (atomically) copies the value stored in record x into record y. Design a compatibility matrix for these operations.

	R(x)	W(x)	C(x, ?)	C(?, x)
R(x)				
W(x)				
C(x, ?)				
C(?, x)				

2. Consider a DBMS that employs a new lock model for the INCREMENT operation. The INCREMENT operation is “special” because it is commutative, e.g., transactions T_1 and T_2 both increment element A; and the final result of A is the same regardless of which transaction operates on A first. Complete the compatibility matrix below if we were to add such an operation. Here, S, X and I denote shared (read), exclusive (write) and increment locks respectively.

	S	X	I
S	T	F	
X	F	F	
I			

We are given the following schedule (incr represents an increment action). Is the schedule serializable and what is the equivalent serial schedule?

$$S = r_1(A);\ r_2(B);\ r_3(C);\ incr_2(C);\ r_3(B);\ incr_1(C);\ incr_2(A);$$

3. Consider the following schedule:

$$X_1(B)\ X_4(A)\ S_3(C)\ S_1(A)\ X_2(D)\ X_2(C)\ X_3(B)\ S_4(D)$$

(A) Give the wait-for-graph for the lock requests for the sequence of actions in the schedule under 2PL (with S/X locks). Is there a deadlock?

(B) To prevent deadlock, we use a lock manager (LM) that adopts the Wait-Die policy. We assume the four transactions have priority: T1 < T2 < T3 < T4. Determine which lock request will be granted (`g'), blocked (`b') or aborted (`a'); for `abort', specify which transaction is aborted - e.g., `a' (T1 is aborted):

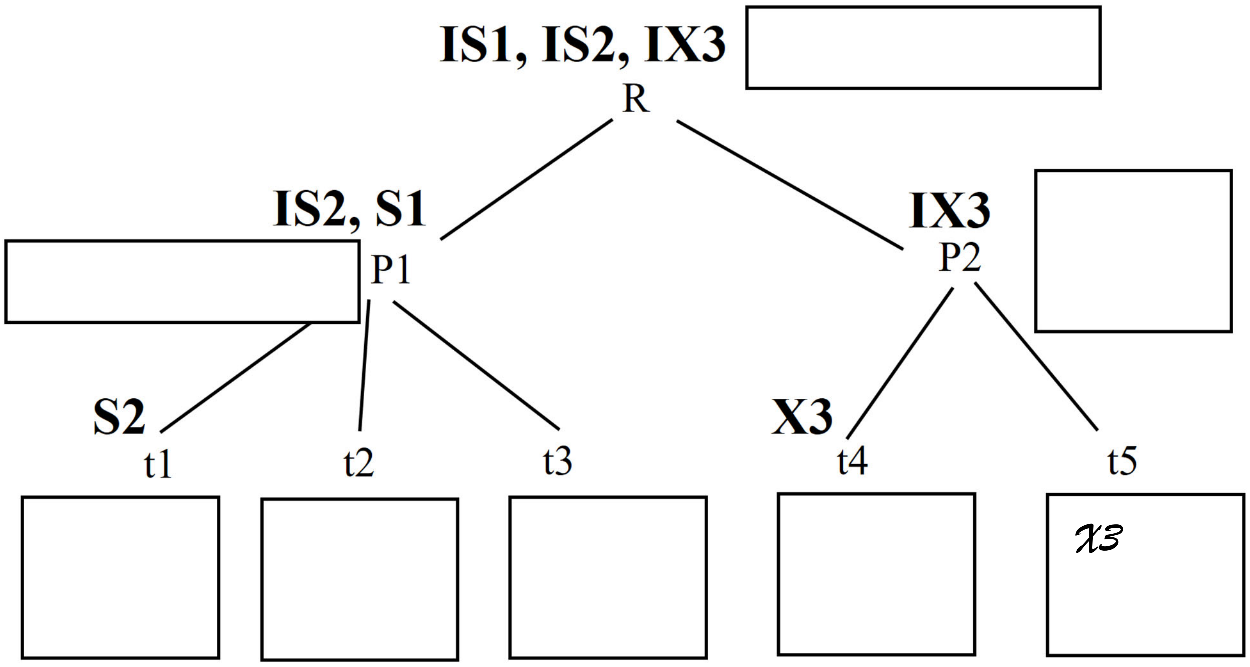
- $X_1(B)$: granted/abort/blocked?
- $X_4(A)$:
- $S_3(C)$:
- $S_1(A)$:
- $X_2(D)$:
- $X_2(C)$:
- $X_3(B)$:
- $S_4(D)$:

(C) Now, we use a lock manager (LM) that adopts the Wound-Wait policy. We assume the four transactions have priority: T1 < T2 < T3 < T4. As in (B), determine which lock request will be granted (`g'), blocked (`b') or aborted (`a').

4. A DBMS uses the multi-granularity locking scheme with three layers as shown in the figure below: R is a table, P_1 and P_2 are two pages, and t_1-t_5 are tuples stored in these pages as reflected in the hierarchy. The figure shows the state of the system at a particular time when FOUR transactions are active. For example, transactions T_1 and T_2 hold IS locks on table R, transaction T_3 holds an IX lock on R. **At this moment, transaction T_4 does not hold any locks yet.** Indicate in the boxes in the figure (besides the objects) the possible **NEXT** lock actions that each of the **FOUR** transactions can possibly take. For example, transaction T_3 could next lock t_5 on X mode, so you should write X_3 (shown as *X3*) in that box. Note that this same box could have another action, say X_n, if another transaction T_n could also get this lock in its next step. Note that transactions T_3 and T_n could not both hold the X lock on t_5; a box with two such transactions simply means that one or the other could get the lock next. Furthermore, note that if a transaction could possibly hold different kinds of locks on an object in its next step, you should list all of them. Finally, a transaction could have multiple

next actions on different objects. Again, you should list all of them in the corresponding boxes.

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 T_1 to request for an S lock or IS lock on t_2, so there should be no IS_1 or S_1 in the box for t_2.


IS1, IS2, IX3
R
IS2, S1
P1
IX3
P2
S2
t1
t2
t3
X3
t4
t5
X3