CS186 Discussion 7

(Transactions, Concurrency Control, Lock Granularity)

Matthew Deng

Transactions

Transactions

- DBMS abstract view of program or activity
- Sequence of reads and writes
- All actions must commit or abort entirely
- Has ACID properties

ACID

- Atomicity
  - All or none
- Consistency
  - Stay consistent
- Isolation
  - Isolated from other transactions
- Durability
  - Commit effects persist

Atomicity and Durability

- Transactions are committed or aborted
- Committed transactions are permanent

Consistency

- A transaction will bring one consistent state to another
- Transactions must satisfy integrity constraints

Isolation (Concurrency)

- Actions of different transactions do not interfere
- Each transaction executes as if ran by itself

Serializability

- Serial schedule
  - Transactions are run one at a time, with no intervention
- Equivalence
  - Same transactions with same actions
  - Leave DB in same state
- Serializable
  - Equivalent to a serial schedule

Conflict Serializability

- Conflict
  - Same object
  - Different transactions
  - At least one is a write

- Conflict Equivalent
  - Same transactions with same actions
  - Conflicts are in the same order

- Conflict Serializable
  - Conflict equivalent to a serial schedule

{conflict serializable schedules} $\subseteq$ {serializable schedules}

Dependency Graph

(Precendence Graph)

- Node
  - Transaction
- Directed edge
  - O_i of T_i conflicts with O_j of T_j, and O_i appears earlier than O_j
  - (T_i, T_j) if T_j depends on T_i

Schedule is conflict serializable if and only if its dependency graph is acyclic.

Transactions Worksheet #2

Transaction Exercises

2. Consider the following schedules:

T1		R(A)	W(A)	R(B)					
T2					W(B)	R(C)	W(C)	W(A)	
T3	R(C)								W(D)

(a) Draw the dependency graph (precedence graph) for the schedule.

Transaction Exercises

2. Consider the following schedules:

T1		R(A)	W(A)	R(B)					
T2					W(B)	R(C)	W(C)	W(A)	
T3	R(C)								W(D)

(a) Draw the dependency graph (precedence graph) for the schedule.

T3 -> T2 [(R(C), W(C)];

T1 -> T2 [R(A), W(A) and W(A),W(A) and R(B),W(B)]

Transaction Exercises

2. Consider the following schedules:

T1		R(A)	W(A)	R(B)					
T2					W(B)	R(C)	W(C)	W(A)	
T3	R(C)								W(D)

(b) Is the schedule conflict serializable? If so, what are all the (conflict) equivalent serial schedules? If not, why not?

Transaction Exercises

2. Consider the following schedules:

T1		R(A)	W(A)	R(B)					
T2					W(B)	R(C)	W(C)	W(A)	
T3	R(C)								W(D)

(b) Is the schedule conflict serializable? If so, what are all the (conflict) equivalent serial schedules? If not, why not?

Yes. Serial schedules: T3, T1, T2; T1, T3, T2.

Transaction Exercises

2. Consider the following schedules:

T1	R(A)		R(B)				W(A)	
T2		R(A)		R(B)				W(B)
T3					R(A)			
T4						R(B)		

(c) Draw the dependency graph (precedence graph) for the schedule.

Transaction Exercises

2. Consider the following schedules:

T1	R(A)		R(B)				W(A)	
T2		R(A)		R(B)				W(B)
T3					R(A)			
T4						R(B)		

(c) Draw the dependency graph (precedence graph) for the schedule.

T3 -> T1 [R(A), W(A)];

T2 -> T1 [R(A), W(A)];

T1 -> T2 [R(B), W(B)];

T4 -> T2 [R(B), W(B)]

Transaction Exercises

2. Consider the following schedules:

T1	R(A)		R(B)				W(A)	
T2		R(A)		R(B)				W(B)
T3					R(A)			
T4						R(B)		

(d) Is the schedule conflict serializable? If so, what are all the (conflict) equivalent serial schedules? If not, why not?

Transaction Exercises

2. Consider the following schedules:

T1	R(A)		R(B)				W(A)	
T2		R(A)		R(B)				W(B)
T3					R(A)			
T4						R(B)		

(d) Is the schedule conflict serializable? If so, what are all the (conflict) equivalent serial schedules? If not, why not?

No. Why not: cycle in the precedence graph (T1 must precede T2, T2 must precede T1)

Concurrency Control

Locks

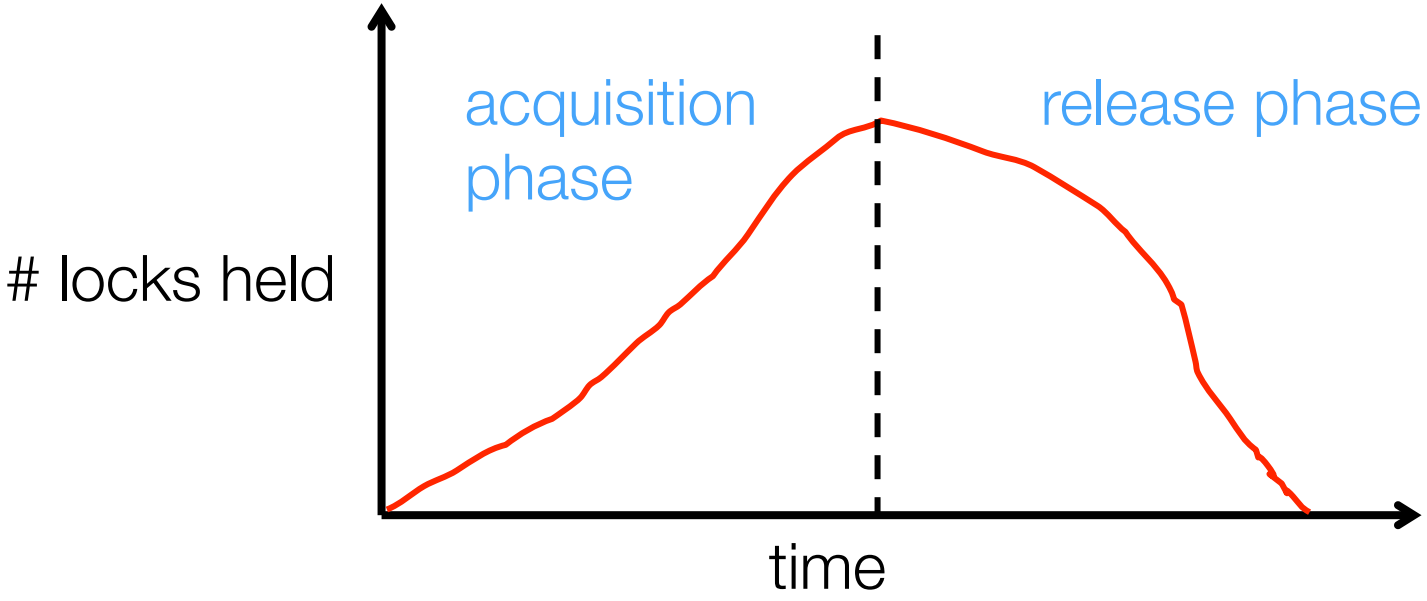
- S (shared lock) for reading
- X (exclusive lock) for writing

	S	X
S	√	–
X	–	–

Lock Compatibility Matrix

Two-Phase Locking (2PL)

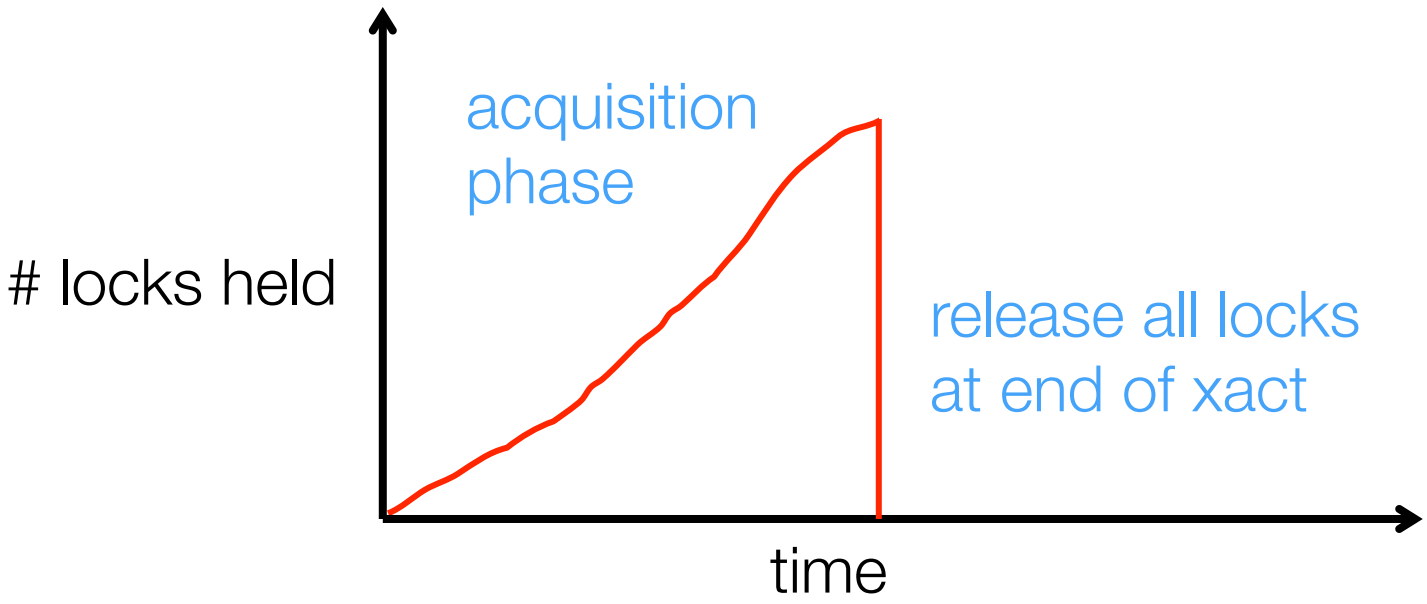
- Transaction cannot get new locks after releasing any locks
- Most common scheme for enforcing conflict serializability



- Guarantees conflict serializability
- Does not prevent cascading aborts

Strict 2PL

- Transaction releases all locks when it commits or aborts



- Guarantees conflict serializability
- prevents cascading aborts

Deadlock Detection

- Waits-For Graph
- Node
  - Transaction
- Directed edge
  - O_i of T_i conflicts with O_j of T_j, and O_j appears earlier than O_i
  - (T_i, T_j) if T_i waits for T_j

Concurrency Control Worksheet #3, 4

Concurrency Control Exercises

3. (a) What will be printed in the following execution (B=3, F=300)?

Concurrency Control Exercises

3. (a) What will be printed in the following execution (B=3, F=300)?

3030

Concurrency Control Exercises

3. (b) Does the execution use 2PL or Strict 2PL?

Concurrency Control Exercises

3. (b) Does the execution use 2PL or Strict 2PL?

Neither, because transaction 2 unlocks F before locking B. In 2PL, a transaction cannot get any new locks after another lock has been released.

Concurrency Control Exercises

3. (c) How would you change the above execution to use 2PL? How would you change it to use Strict 2PL? Does the output change?

Concurrency Control Exercises

3. (c) How would you change the above execution to use 2PL? How would you change it to use Strict 2PL? Does the output change?

   There are many ways to get 2PL/Strict 2PL schedules, but here is one:

   2PL: You could move T2's Unlock(F) to any point after it's Lock_S(B)

   Strict 2PL: You would have to move T1's Unlock(B) and T2's Unlock(F) to the end of execution; In both of these cases, the output will not change.

Concurrency Control Exercises

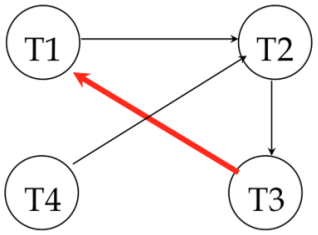
4. Consider the following schedule. Draw a “waits-for” graph and state whether or not there is a possibility of deadlock. Assume that no locks are released within the timeframe we are looking.

T1	S(A)	S(D)		S(B)					
T2			X(B)				X(C)		
T3					S(D)	S(C)			X(A)
T4								X(B)	

Concurrency Control Exercises

4. Consider the following schedule. Draw a “waits-for” graph and state whether or not there is a possibility of deadlock. Assume that no locks are released within the timeframe we are looking.

T1	S(A)	S(D)		S(B)					
T2			X(B)				X(C)		
T3					S(D)	S(C)			X(A)
T4								X(B)	

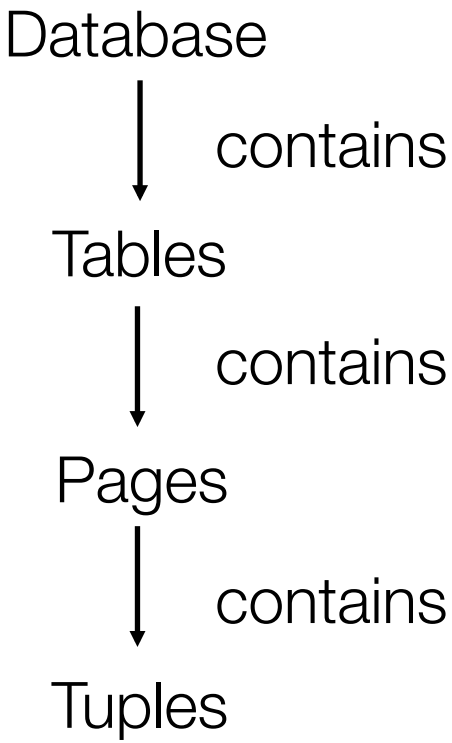


Yes, deadlock is possible since there is a cycle in the waits-for graph.

Lock Granularity

Lock Granularity

- Select amount to lock
- Used with 2PL to guarantee serializability
- Steps:
  1. Start at root
  2. Get locks top-down
  3. Release locks bottom-up



Intent Locks

- S: shared lock for reading
- X: exclusive lock for writing (and reading)
- IS: intent to get S lock(s) at finer granularity
- IX: intent to get X lock(s) at finer granularity
- SIX: shared lock with intent to get X lock(s) at finer granularity

Lock Compatibility Matrix

	S	X
S	√	–
X	–	–

Lock Compatibility Matrix

	IS	IX	SIX	S	X
IS					
IX					
SIX					
S				√	–
X				–	–

Lock Compatibility Matrix

	IS	IX	SIX	S	X
IS	√	√	√	√	–
IX	√	√	–	–	–
SIX	√	–	–	–	–
S	√	–	–	√	–
X	–	–	–	–	–

Lock Granularity Worksheet #1

Lock Granularity Exercises

1. Suppose a transaction, T1, wants to scan a table R and update a few of its tuples. What kind of locks should T1 have on R, its pages, and the tuples that are updated?

Lock Granularity Exercises

1. Suppose a transaction, T1, wants to scan a table R and update a few of its tuples. What kind of locks should T1 have on R, its pages, and the tuples that are updated?

   SIX lock on R

   IX lock on pages

   X lock on updated tuples

Lock Granularity Exercises

2. Is an S lock compatible with an IX lock? Explain why or why not. Make your description as simple as possible.

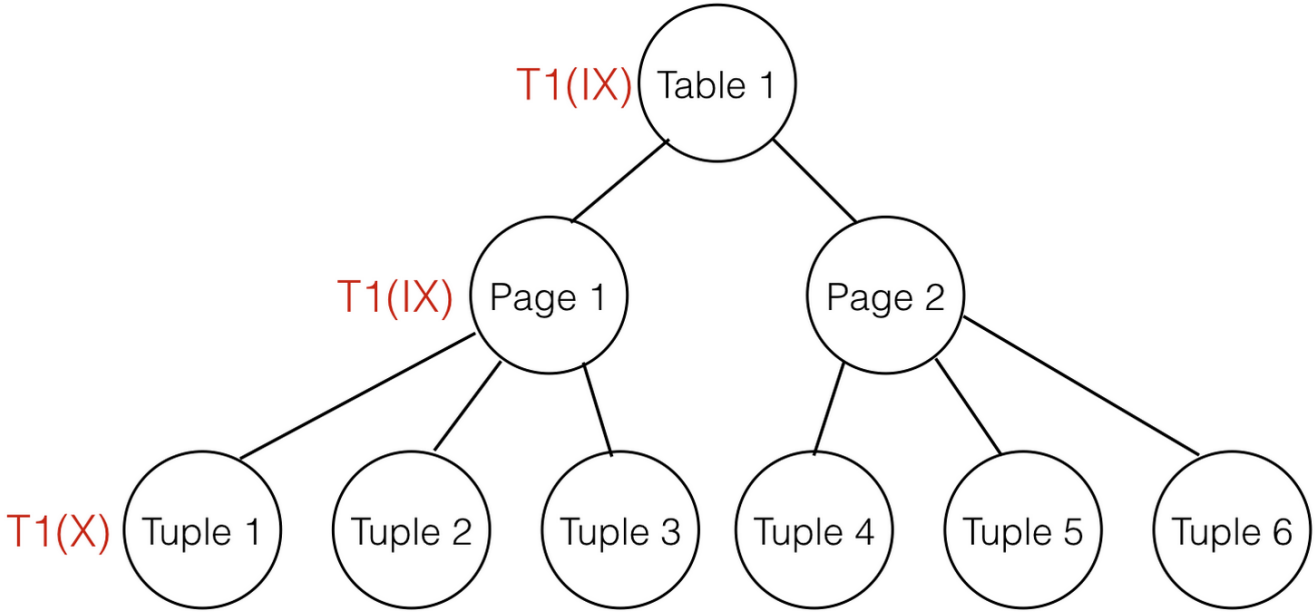
Lock Granularity Exercises

2. Is an S lock compatible with an IX lock? Explain why or why not. Make your description as simple as possible.

   Suppose T1 wants an S lock on an object, O, and T2 wants an IX lock on the same object O. An S lock implies that T1 will read the entire object (all of its sub-objects). An IX lock implies that T2 will write some of the sub-objects of the object. This means that there is some sub-object of O that T1 will read and T2 will write. This is not valid, so the S and IX locks must be incompatible.

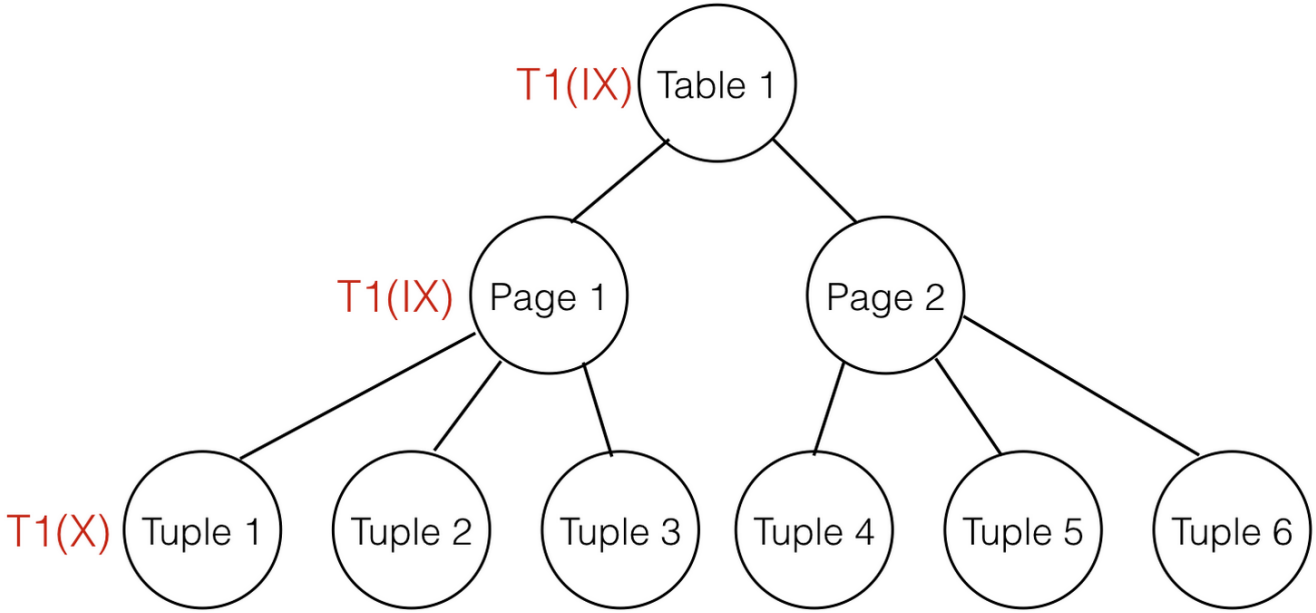
Lock Granularity Exercises

3. Given that transaction T1 has an IX lock on table 1, an IX lock on page 1, and an X lock on tuple 1, which locks could be granted to transaction T2 for tuple 2?



Lock Granularity Exercises

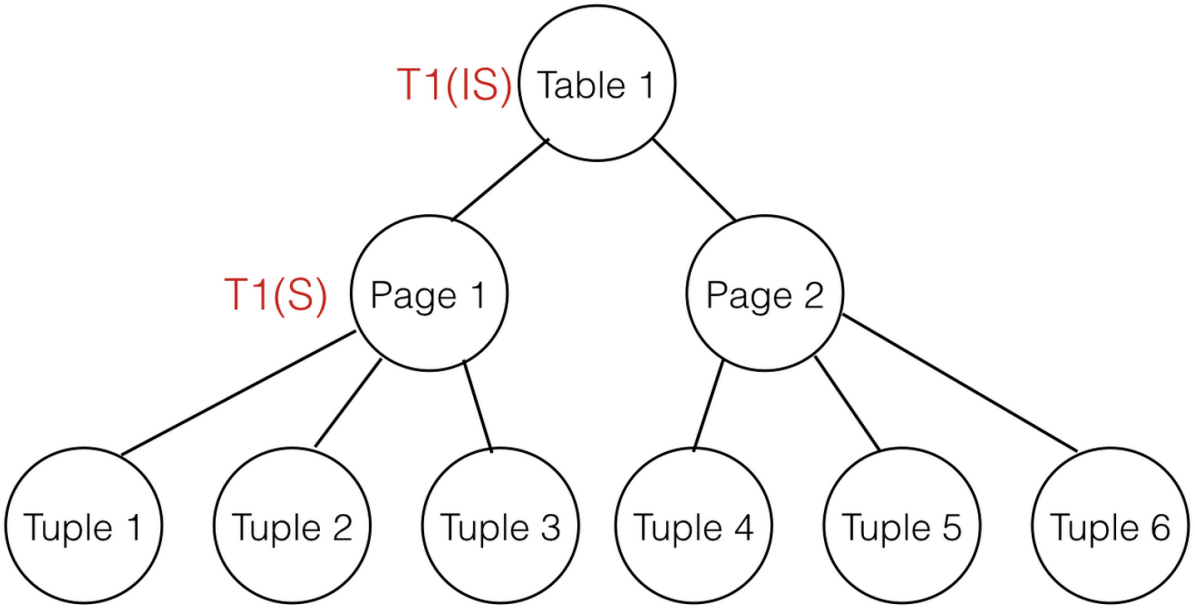
3. Given that transaction T1 has an IX lock on table 1, an IX lock on page 1, and an X lock on tuple 1, which locks could be granted to transaction T2 for tuple 2?



X, S

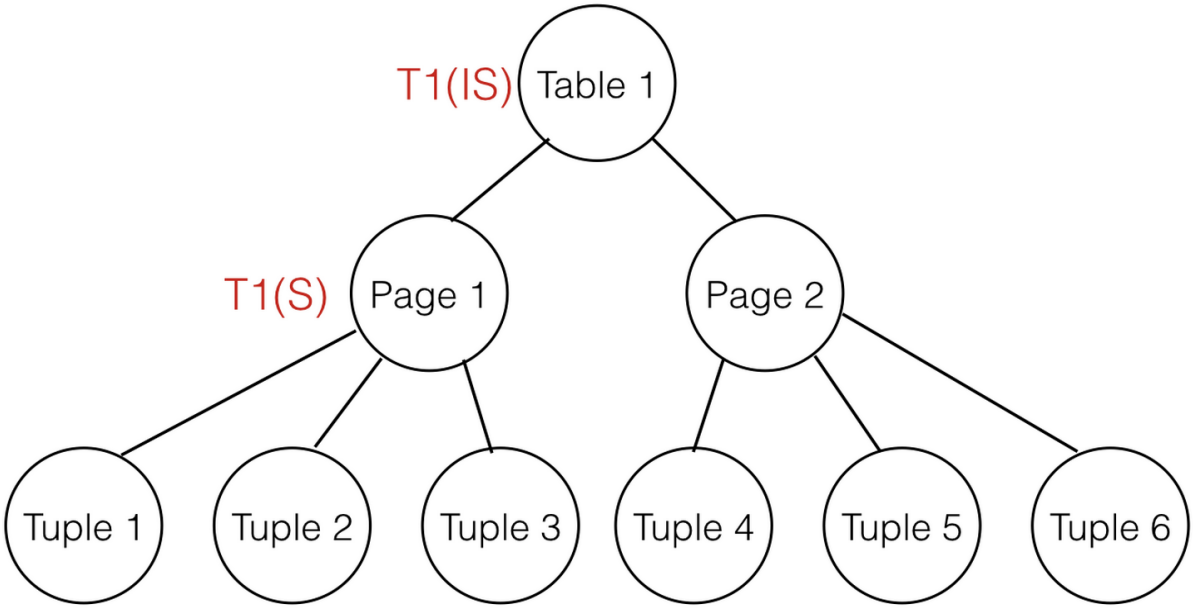
Lock Granularity Exercises

4. Given that T1 has an IS lock on table 1 and an S lock on page 1, what locks could be granted to T2 for page 1?



Lock Granularity Exercises

4. Given that T1 has an IS lock on table 1 and an S lock on page 1, what locks could be granted to T2 for page 1?



S, IS