Intro to DB

CHAPTER 15 TRANSACTIONS

Chapter 15: Transactions

- Lock-Based Protocols
- Deadlock Handling
- Multiple Granularity
- Timestamp-Based Protocols
- Validation-Based Protocols
- Multiversion Schemes
- Snapshot Isolation
- Insert and Delete Operations
- Concurrency in Index Structures

Concurrency Control

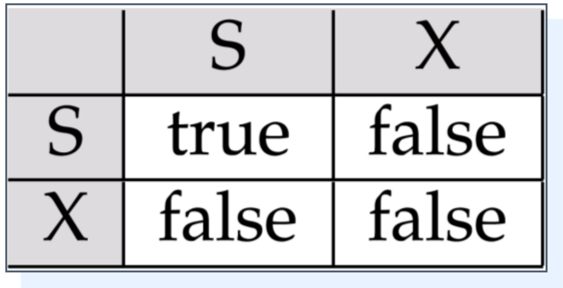
- Schedules must be serializable and recoverable (for database consistency)
  - and preferably cascadeless
- A policy in which only one transaction can execute at a time generates but provides a poor degree of concurrency.
- Concurrency-control schemes tradeoff between the amount of concurrency they allow and the amount of overhead that they incur.
-

Lock-Based Protocols

- A lock is a mechanism to control concurrent access to a data item
- Two modes :
  1. *exclusive (X) mode*: both read and write
     (lock-X instruction)
  2. *shared (S) mode*: only read
     (lock-S instruction)
- Lock requests are made to
- Transaction can proceed only after request is granted.

Granting of Locks

- Lock-compatibility matrix



	S	X
S	true	false
X	false	false

- A transaction may be granted a lock on an item if the lock(s) already held on the item by other transactions
- Any number of transactions can hold shared locks on an item
- If any transaction holds an exclusive on the item no other transaction may hold any lock on the item.

Example

```
T₂:   lock-S(A);
      read (A);
      unlock(A);
      lock-S(B);
      read (B);
      unlock(B);
      display(A+B)
```

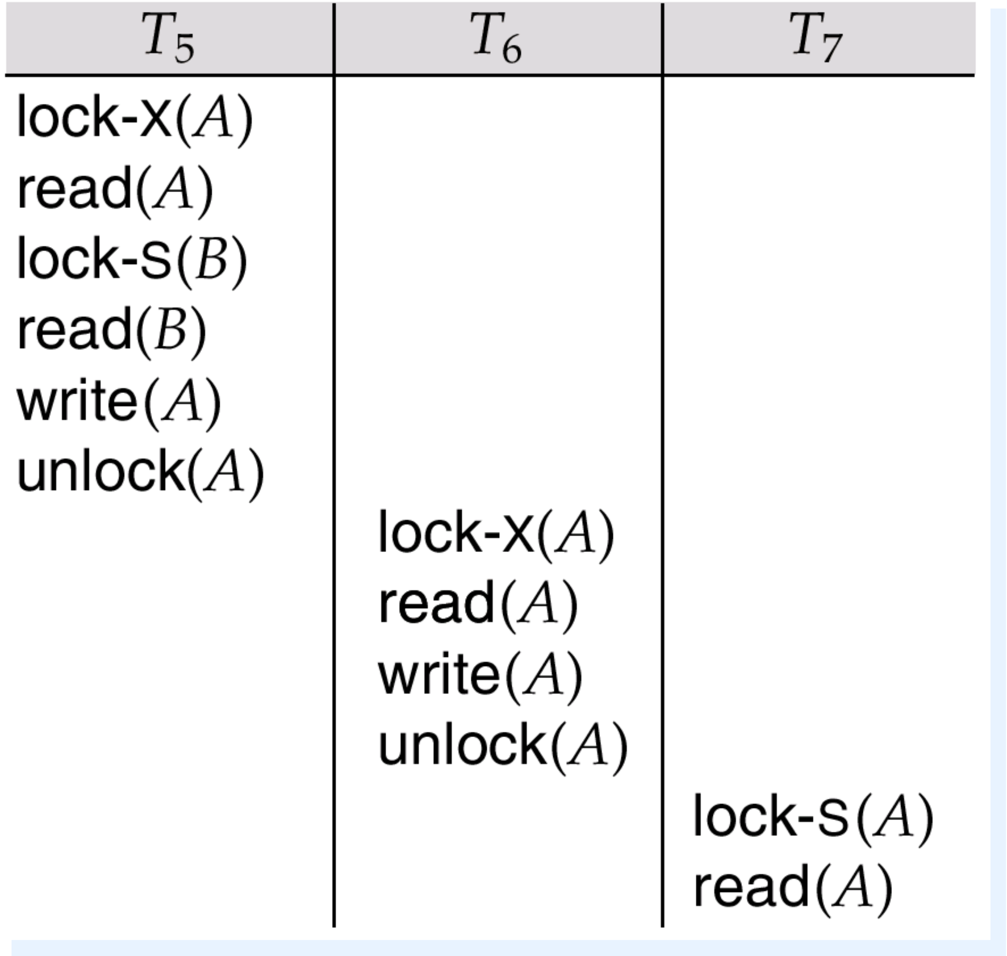
- Locking as above is not sufficient to guarantee serializability

  — if *A* and *B* get updated in-between the read of *A* and *B*, the displayed sum would be wrong.

Two-Phase Locking Protocol (2PL)

- Locking Protocol
  -
  - Locking protocols restrict the set of possible schedules.
- 2PL
  - Phase 1: Growing Phase
    -
      - can acquire a **lock-S** or **lock-X** on item
      - can convert a **lock-S** to a **lock-X** (**upgrade**)
    - transaction may not release locks
  - Phase 2: Shrinking Phase
    -
      - can release a **lock-S** or **lock-X**
      - can convert a **lock-X** to a **lock-S** **(downgrade**)
    - transaction may not obtain locks

Example



T_5	T_6	T_7
lock-X(A)		
read(A)		
lock-S(B)		
read(B)		
write(A)		
unlock(A)		
	lock-X(A)	
	read(A)	
	write(A)	
	unlock(A)	
		lock-S(A)
		read(A)

Features of 2PL

- The protocol assures (conflict) serializability
  - transactions can be serialized in the order of their lock points
    (the point where a transaction acquired its final lock).
  - There can be conflict serializable schedules that cannot be obtained if 2PL is used
- Deadlocks:

  2PL □ freedom from deadlocks
  - starvation also possible
- Cascading rollback:

  □ under 2PL

Strict / Rigorous 2PL

- Strict 2PL
  - A transaction must hold all its *exclusive* locks until it commits/aborts

- Rigorous 2PL
  - *all* locks are held until commit/abort

END OF CHAPTER 15