**Two-Phase Locking (2PL) Protocol**

* **A transaction is two-phase locked if:**
  + before reading x, it sets a read lock on x
  + before writing x, it sets a write lock on x
  + it holds each lock until after it executes the corresponding operation
  + after its first unlock operation, it requests no new locks
* **Each transaction sets locks during a growing phase and releases them during a shrinking phase.**
* Phase 1: Growing phase
  + Transaction may obtain locks
  + But may not release them
* Phase 2: Shrinking phase
  + Transaction may only release locks
  + Can be shown that this achieves *conflict-serializability*
  + *lock-point*: the time at which a transaction acquired last lock
  + if *lock-point*(T1) < *lock-point*(T2), there can’t be an edge from T2 to T1 in the *precedence graph*
* Guarantees *conflict-serializability*, but not cascade-less recoverability

Locking granularity

* Locking granularity
  + What are we taking locks on ? Tables, tuples, attributes ?
* Coarse granularity
  + e.g. take locks on tables
  + less overhead (the number of tables is not that high)
  + very low concurrency
* Fine granularity
  + e.g. take locks on tuples
  + much higher overhead
  + much higher concurrency
  + What if I want to lock 90% of the tuples of a table ?
    - Prefer to lock the whole table in that case