Concurrency Control

- *Instructor*

  Dr. Sanam Shahla Rizvi

  PhD in Information and Communication from Ajou University, Korea

Strict Two-Phase Locking (2PL)

- Rules:
  - If a transaction T wants to read (respectively, modify) an object, it first requests a shared (respectively, exclusive) lock on the object.
  - If a transaction holds an exclusive lock on an object, no other transaction holds a shared or exclusive lock on the same object.
  - All locks held by a transaction are released when the transaction is completed.
  - A transaction cannot request additional locks once it releases any lock.

Lock Management

- Lock manager keeps track of locks issued to transactions in a lock table.
- Lock table entry for an object contains:
  - The number of transactions currently holding a lock on the object (in shared mode)
  - The nature of a lock (shared or exclusive)
  - A pointer to a queue of lock requests.

Implementing Lock and Unlock Requests

- Request shared lock
  - Request queue is empty
  - Object is not currently locked in exclusive mode
  - Grant lock
- Request exclusive lock
  - Request queue is empty
  - Grant lock
- Otherwise
  - Request added in queue

Implementing Lock and Unlock Requests

T1 → S(O)

T2 → requests X(O)

Include in queue

T3 → requests S(O)

Include in queue

T1 commits

T2 → granted lock

Deadlocks

T1 → X(A)

T2 → X(B)

T1 requests X(B)

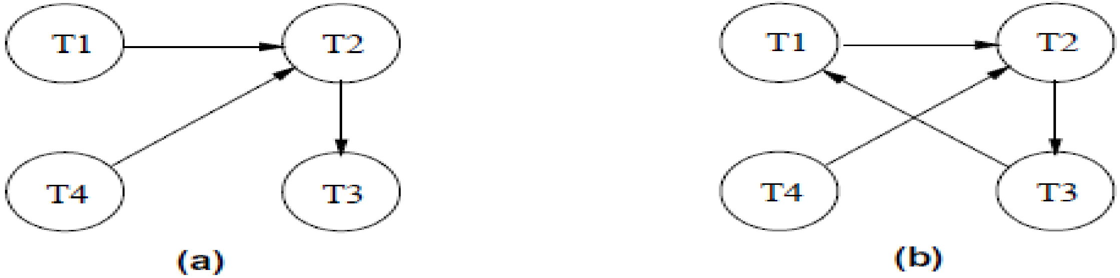
T2 requests X(A)

Deadlock Prevention

- Give each transaction a timestamp when it starts up.
- The oldest transaction has the highest priority.
- If a transaction Ti requests a lock and transaction Tj holds the requested object.
- Two policies:
- Wait-die
  - If Ti has higher priority, it is allowed to wait; otherwise it is aborted. (Ti is older than Tj)
- Wound-wait
  - If Ti has higher priority, abort Tj; otherwise Ti waits (Ti is older than Tj)

Deadlock Detection

- **Waits-for graph:** A deadlock is resolved by aborting a transaction that is on a cycle and releasing its locks; this action allows some of the waiting transactions to proceed.



(a)

(b)

- **Use a timeout mechanism:** If a transaction has been waiting too long for a lock, we can assume (pessimistically) that it is in a deadlock cycle and abort it.

Concurrency Control without Locking

- **Optimistic concurrency control:** The basic premise is that most transactions will not conflict with other transactions, and the idea is to be as permissive as possible in allowing transactions to execute.
- Transactions proceed in three phases:

1. **Read:** The transaction executes, reading values from the database and writing to a private workspace.
2. **Validation:** If the transaction decides that it wants to commit, the DBMS checks whether the transaction could possibly have conflicted with any other concurrently executing transaction. If there is a possible conflict, the transaction is aborted; its private workspace is cleared and it is restarted.
3. **Write:** If validation determines that there are no possible conflicts, the changes to data objects made by the transaction in its private workspace are copied into the database.