CSE 344 Introduction to Data Management

Section 9: Transactions
ACID Revisit

- **Atomicity**: Either all changes performed by transaction occur or none occurs

- **Consistency**: A transaction as a whole does not violate integrity constraints

- **Isolation**: Transactions appear to execute one after the other in sequence

- **Durability**: If a transaction commits, its changes will survive failures
Serializability
Serial Schedule

Transactions are executed one after the other, in some sequential order.

- Safe
- But inefficient!
A schedule is serializable if it is equivalent to a serial schedule.
Conflicts

Two actions by same transaction $T_i$: $r_i(X); w_i(Y)$

Two writes by $T_i, T_j$ to same element: $w_i(X); w_j(X)$

Read/write by $T_i, T_j$ to same element: $w_i(X); r_j(X)$
$r_i(X); w_j(X)$
Conflict-Serializable Schedule

if it has the *same conflicts* as a serial schedule
Testing for conflict-serializability

Precedence graph:
• A node for each transaction Ti,
• An edge from Ti to Tj whenever an action in Ti conflicts with, and comes before an action in Tj
• The schedule is conflict-serializable iff the precedence graph is acyclic
Locking

- **Two Phase Locking (2PL):** In every transaction, all lock requests must precede all unlock requests.

- **Strict 2PL:** All locks are held until the transaction commits or aborts.
Locking exercise

- L1(A), W1(A), U1(A), Co1
  - Is this schedule possible under 2PL?
  - L1(A), W1(A), R1(A), Co1, U1(A), L2(A), W2(A), Co2, U2(A)
    - Strict 2PL?
      - Possible under strict 2PL?
        - W1(A), R1(A), Co1, W2(A), Co2
          - Possible under strict 2PL?
            - L1(A), W1(A), W2(A), Co1, U1(A)
              - Possible under strict 2PL?
                - W1(A), W2(A), Co1
                  - Possible under strict 2PL?
Worksheet