Transaction Management

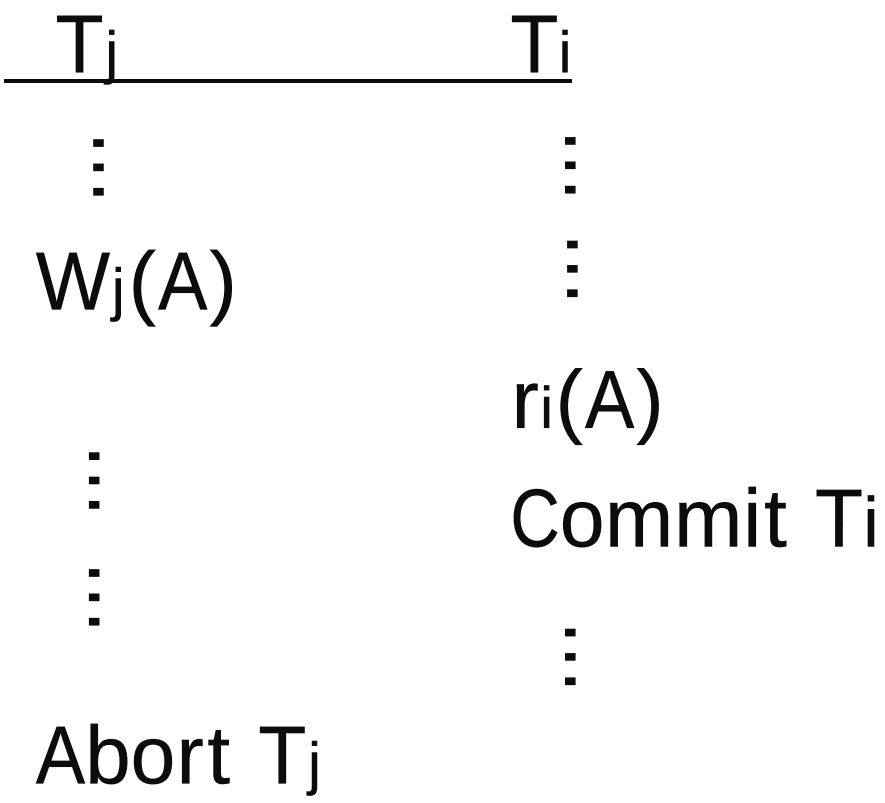
Deadlocks

More About Transaction Management

- Topics
  - More on schedules
    - Recoverable schedule
    - Cascading rollback schedule
    - Strict schedule
  - Deadlocks
    - Prevention
    - Detection

Concurrency Control & Recovery

Example:

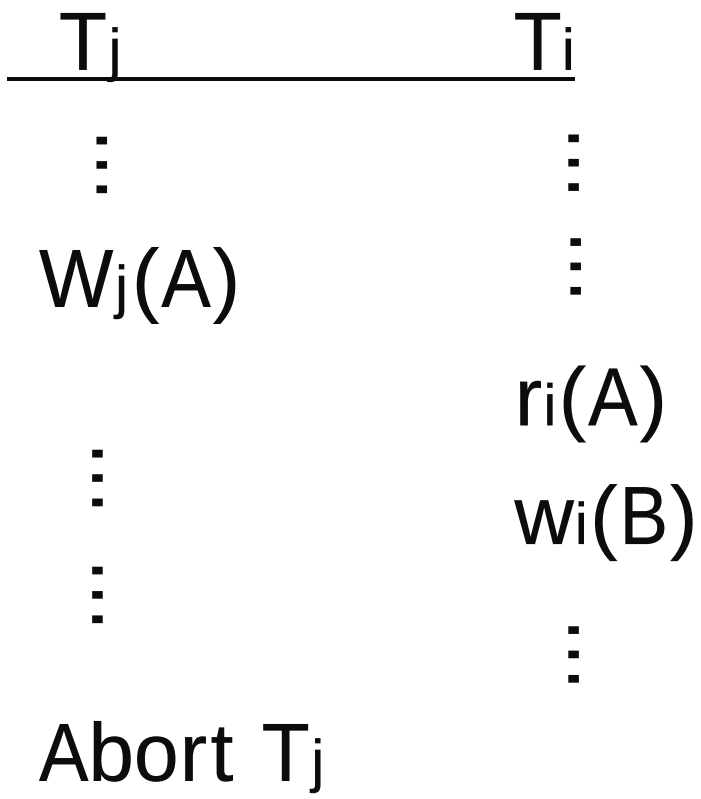


☛ Non-Persistent Commit (Bad!)

avoided by recoverable schedules

Concurrency Control & Recovery

Example:



☛ Cascading rollback (Bad!)

avoided by avoids-cascading-rollback (ACR) schedules

Recoverable Schedule

- To model this, two new actions:
  - c_i - transaction T_i commits
  - a_i - transaction T_i aborts
- A schedule is recoverable if each transaction commits only after each transaction from which it has read has committed.
- Examples:
  - S1: w1(A);w1(B);w2(A);r2(B);c1;c2;
- Notes:
  - A serializable schedule may not be a recoverable schedule.
    - S: w1(A);w1(B);w2(A);r2(B);c2;c1;
  - A recoverable schedule may not be a serializable schedule.
    - S: w2(A);w1(B);w1(A);r2(B);c1;c2;

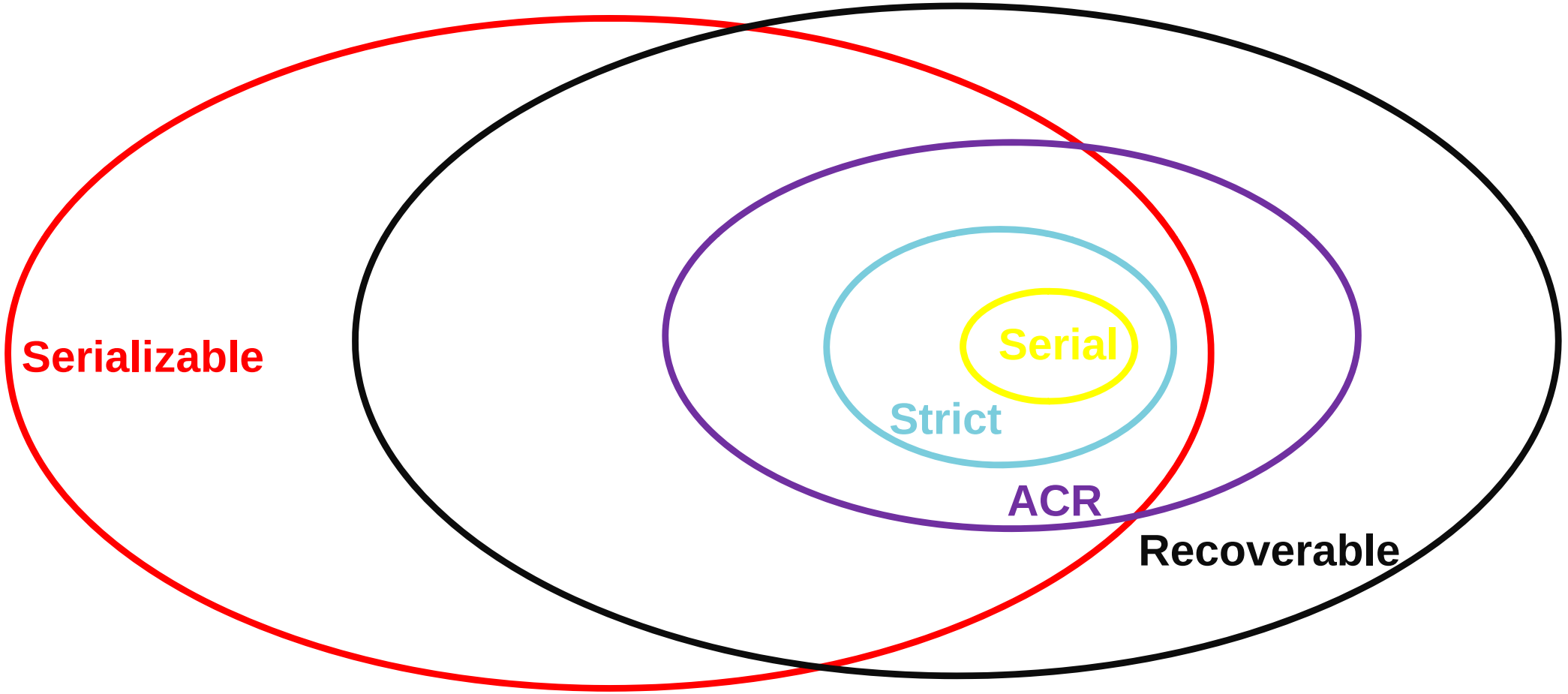
ACR Schedule

- A schedule avoids cascading rollback (ACR) if transactions may read only values written by committed transactions.
- Examples:
  - S2:w1(A);w1(B);w2(A);c1;r2(B);c2;
- Notes:
  - Every ACR schedule is recoverable.

Strict Schedule

- Managing rollbacks using locking
  - Strict locking: a transaction must not release any X locks until the transaction has either committed or aborted, and the log record has been flushed to disk.
- A schedule follows the strict-locking rule is called a strict schedule.
- Examples:
  - S3: w1(A);w1(B);c1;w2(A);r2(B);c2;
- Notes:
  - Every strict schedule is ACR.
  - Every strict schedule is serializable.

Relations Among Schedules



Deadlocks

- Deadlock: a set of transactions where each transaction is waiting for another in the set, and none of them can make progress
- Two major methods to handle deadlocks
  - Detection
    - Timeout
    - Waits-for graph
  - Prevention: ensure that the system will never enter a deadlock state –commonly used when the probability that the system would enter a deadlock state is high.
    - Waits-for graph
    - Resource ordering
    - Wait-die
    - Wound-wait
- Both methods have to roll back some transactions!

Timeout

- If transaction waits more than L seconds, roll it back!
- Simple scheme
- Hard to select L!

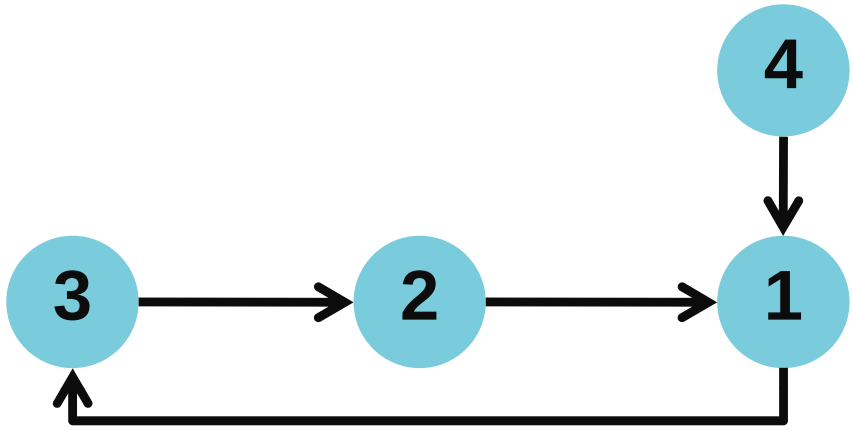
Wait-For Graph

- Build a wait-for graph:
  - Each vertex is a transaction;
  - Edge T1→T2: T1 waits for a lock held by T2;
  - Deadlock: a cycle in the wait-for graph!
- Recovery: rolling back some transactions
- Example:

T1: l1(A);r1(A);l1(B);w1(B);u1(A);u1(B);
T2: l2(C);r2(C);l2(A);w2(A);u2(C);u2(A);
T3: l3(B);r3(B);l3(C);w3(C);u3(B);u3(C);
T4: l4(D);r4(D);l4(A);w4(A);u4(D);u4(A);

Wait-For Graph

	T1	T2	T3	T4
(1)	l1(A);r1(A);			
(2)		l2(C);r2(C);		
(3)			l3(B);r3(B);	
(4)				l4(D);r4(D);
(5)		l2(A);Denied		
(6)			l3(C);Denied	
(7)				l4(A);Denied
(8)	l1(B);Denied			



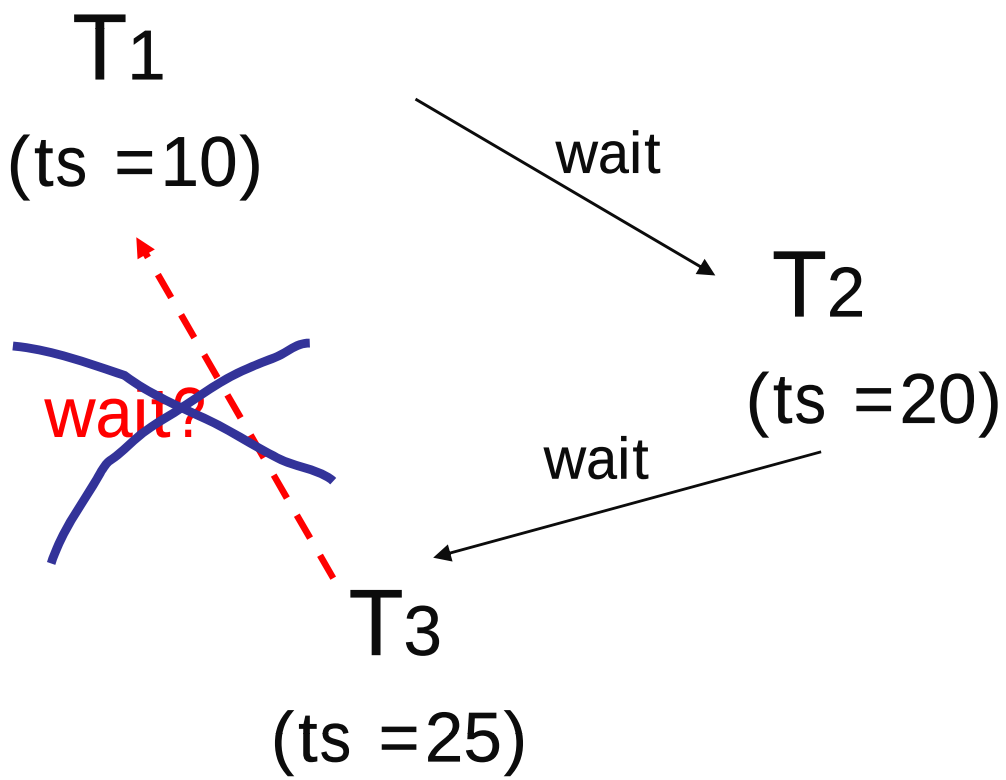
Resource Ordering

- Order all elements A1, A2, …, An
- A transaction T can lock Aj after Ai only if j > i!

Problem : Ordered lock requests not realistic in most cases

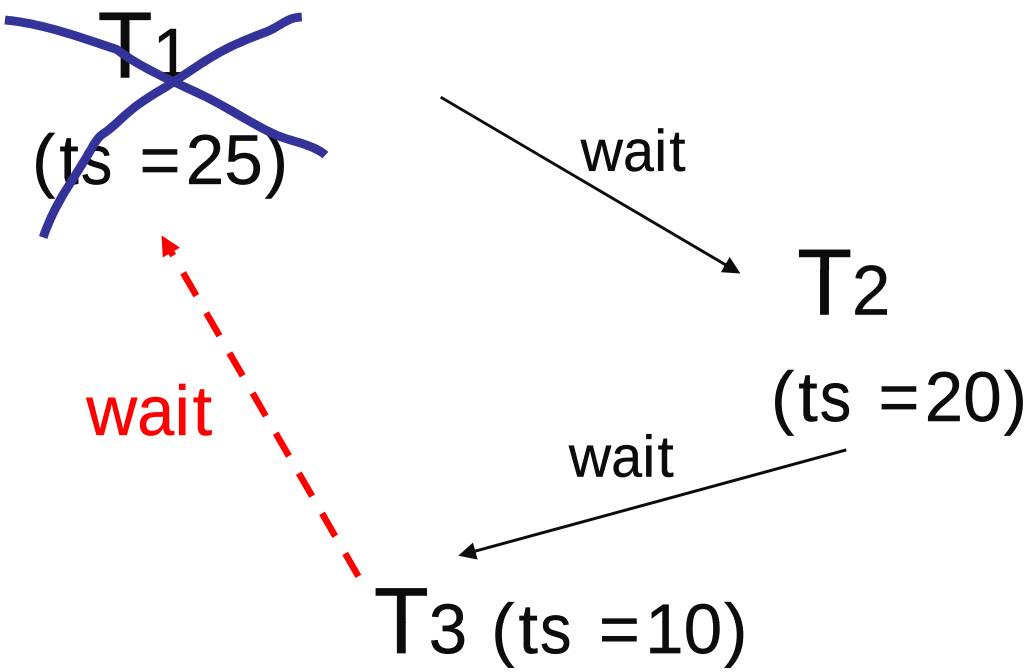
Wait-Die

- Transactions given a timestamp when they arrive …. ts(Ti)
- Ti can only wait for Tj if ts(Ti)< ts(Tj)...else die



Wound-Wait

- Transactions given a timestamp when they arrive … ts(Ti)
- Ti wounds Tj if ts(Ti)< ts(Tj); else Ti waits
  - "Wound": Tj rolls back and gives lock to Ti



Summary

- Transaction Management
  - ACID properties of transactions
  - Logging
    - Concepts: Undo logging; Redo logging; Undo/Redo logging;
    - Recovery using logging
      - Checkpointing
  - Concurrency control
    - Different types of schedule: serial, serializable, conflict-serializable, recoverable, ACR, strict;
    - Two major types of concurrency control
      - Pessimistic concurrency control, e.g., locking
      - Optimistic concurrency control, e.g., validation
    - Deadlock issues