Final Exam Details

- **Exam Day:** Monday, March 18
- **Exam classroom:** V32
- **Total points:** 50
- **Total Exam Time:** 3 hrs
- **Time Window:** 9:30am – 12:30pm
- T-SQL is not part of the exam

Points Distribution

- Lesson 9 : Serializability and Recoverability ~ 10
- Lesson 10 : 2PL and Deadlock ~ 13
- Lesson 11 : Timestamping ~ 10
- Lesson 12 : DB Recovery ~ 9
- Lesson 13 : NoSQL DB: Cassandra ~ 8

Important Topics

- **Lessons 9 - Transactions: Serizalizability & Recoverability**
  - ACID properties of Transactions
  - Conflict-serializability and Recoverability
    - Precedence graph, swapping rules

Conflict - Serializability for Concurrency Control

Swapping Rules

- If the adjacent operations in the schedule are of the same transaction, then DO NOT swap.
- If the adjacent operations use the same DB element, and at least one of the operations is a write, then DO NOT swap.

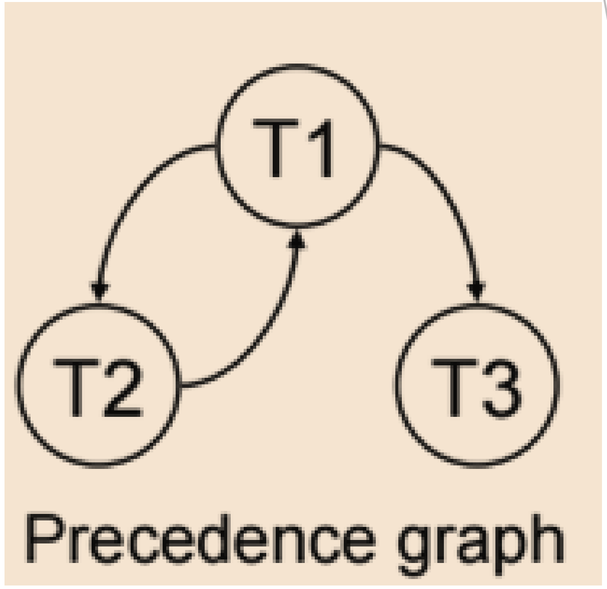
Precedence (Serialization) Graph

- A node for each transaction
- A directed edge (Ti →Tj) if Ti **writes** a value before Tj **reads/writes** it
- A directed edge (Ti → Tj) if Ti **reads** a value before Tj **writes** it

If the precedence graph contains a cycle, then the schedule is not conflict serializable.

Draw Precedence Graph

- T1 Read(X)
- T2 Read(Y)
- T1 Write(X)
- T2 Read(X)
- T3 Read(Z)
- T3 Write(Z)
- T1 Read(Y)
- T3 Read(X)
- T1 Write(Y)



A directed edge (Ti → Tj) if Ti writes a value before Tj **reads/writes** it

A directed edge (Ti → Tj) if Ti reads a value before Tj **writes** it

Serializability & Recoverability

- **Serializability:**
  - For determining Serializability, we don't need to look at commit or abort operations.
- **Recoverability:**
  - For determining Recoverability, we do need to look at transaction commit and abort operations given in the schedule.
  - If a transaction T2 has read an updated value from T1, then for the schedule to be recoverable, T1 commit (or abort) must occur before T2 commit.

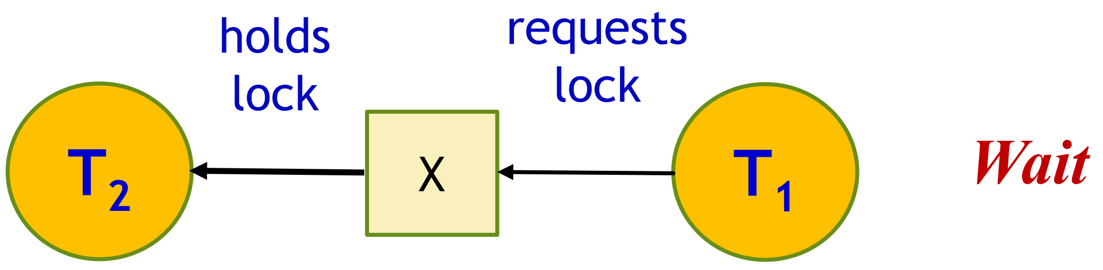
Important Topics

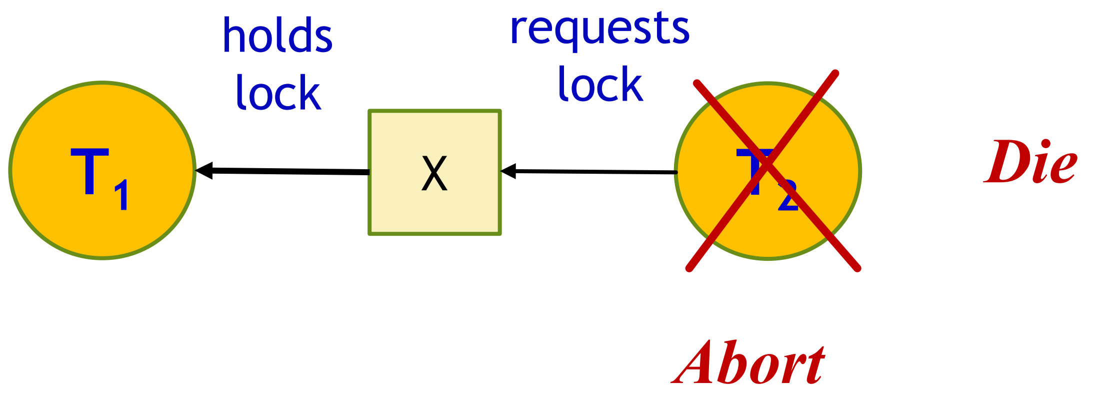
- **Lessons 10 - Concurrency Control Protocol: 2PL and Deadlock**
  - Details of Concurrency Control Protocol– 2PL
  - Deadlock
    - Wait-for graph
    - DL prevention policies – Wait-Die, Wound-Wait

Rules for Two-Phase Locking (2PL)

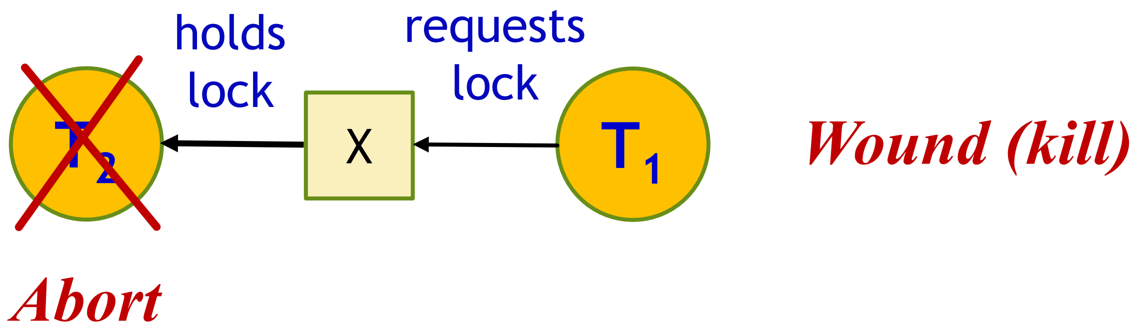
- A transaction must acquire a lock on an item before operating on the item. The lock may be read or write, depending on the type of access needed.
- A transaction follows the two-phase locking protocol if all locking operations precede the first unlock operation in the transaction.
- Once the transaction releases a lock, it can never acquire any new locks.

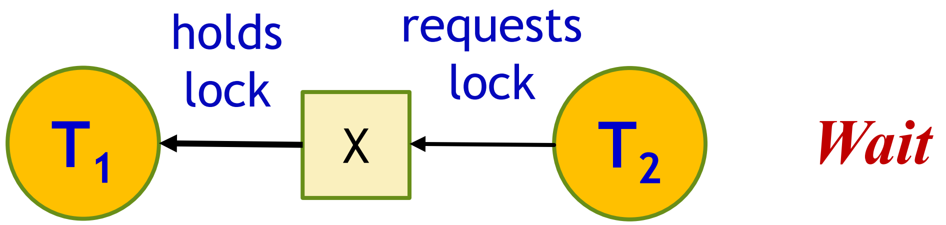
Deadlock Prevention : Wait-Die





Deadlock Prevention : Wound-Wait





2PL Example Problem

- Consider the following sequence of actions, listed in the order the actions are presented to the DBMS. Assume that the concurrency control mechanism is 2PL with "Wound-Wait" deadlock prevention strategy.
- Describe how the concurrency control mechanism handles the sequence of actions.
- Acquire locks as late as possible and release locks as early as possible. Waiting transactions continued and brought up-to-date as early as possible.

T1: R(X), T2: W(X), T2: W(Y), T3: W(Y), T1: W(Y), T3:R(Z), T3:W(Z), T1: Commit, T2: Commit, T3: Commit

Solution

T1: R(X), T2: W(X), T2: W(Y), T3: W(Y), T1: W(Y), T3:R(Z), T3:W(Z), T1: Commit, T2: Commit, T3: Commit

t1 - T1 wants shared lock on X and gets it →**T1:R(X)**

t2 - T2 wants exclusive lock on X, but since T2 is younger than T1, **T2 waits** (wound-wait policy)

t3 - T3 wants exclusive lock on Y and gets it → **T3:W(Y)**

t4 - T1 wants exclusive lock on Y ; as T3 is holding the lock and is younger than T1, **T3 will be aborted**. T1 will be granted the lock.

t5 – T1 releases read lock on X (locks should be released as early as possible)

t6 – T2 continues and gets write lock on X → **T2 :W(X)** (waiting transactions continued and brought up to date as early as possible)

t7 – T1 continues → **T1 :W(Y)**

t8 – T1 commits and unlocks Y

t9 – T2 gets write lock on Y

t10 – T2 unlocks X (locks should be released as early as possible)

t11 – T2 continues → **T2 :W(Y)**

t12 - T2 commits and releases lock Y

t13 – T3 restarts and gets write lock on Y → **T3:W(Y)**

t14 – T3 asks for write lock on Z and gets it

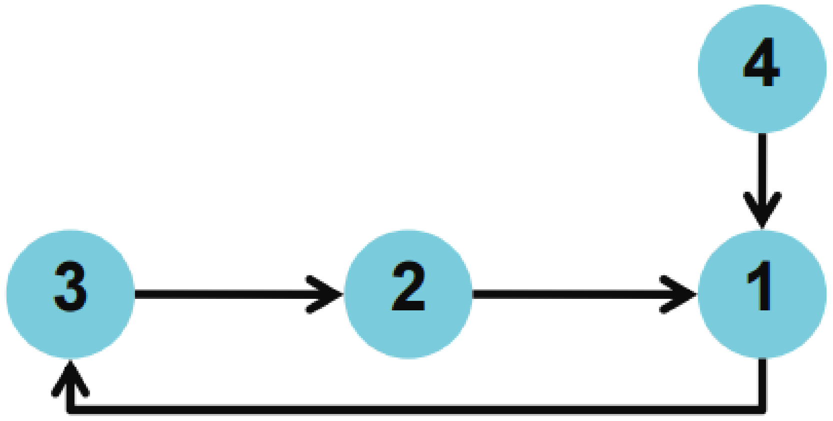
t15 – T3 releases Y lock (locks should be released as early as possible)

t16 – T3 proceeds → **T3:R(Z), T3:W(Z)**

t17 - T3 commits & releases lock Z

Draw 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			



Important Topics

- **Lessons 11 - Concurrency Control Protocol: Timestamping**
  - Multiple Granularity Locking
    - Lock compatibility matrix
  - Details of Concurrency Control Protocol– Timestamping
    - Single-version
    - Multiversion

Multiple Granularity Locking

Lock Compatibility Matrix

	IS	IX	S	SIX	X
IS	✔	✔	✔	✔	✗
IX	✔	✔	✗	✗	✗
S	✔	✗	✔	✗	✗
SIX	✔	✗	✗	✗	✗
X	✗	✗	✗	✗	✗

IS : Intention Shared
IX : Intention Exclusive
S : Shared
X : Exclusive
SIX : Shared & Intention Exclusive

Single version timestamping

```
Read(X)
    If X is written by a younger transaction
            Abort, Rollback, Restart with new timestamp
    Else
            Read   //read_ts  = max{ts(T), read_ts}
Write(X)
    If X is read by a younger transaction
            Abort, Rollback, Restart
    Else If X is written by younger transaction
            Abort, Rollback, Restart with new timestamp
    Else
            Write    // set write_ts as ts(T)
```

Multiversion timestamping

```
Read(X)
    Pick the version.
        The version with largest write_ts <= ts(T)
    Read   // read_ts  = max{ts(T), read_ts} //Read never
                                                fails

Write(X)
    Pick the version.
        The version with largest write_ts <= ts(T)
    If read_ts <= ts(T)     //not read by younger transaction
        Create new version
        //set read and write timestamp as ts(T)
    Else
        Abort, Rollback, Restart
```

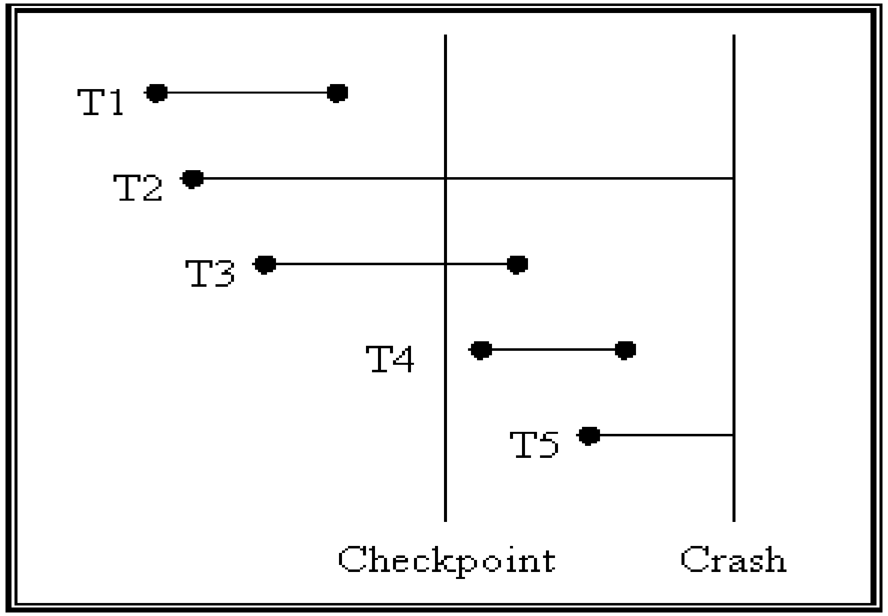
Important Topics

- **Lesson 12 - DB Recovery**
  - Logs File
  - Checkpointing
  - Recovery techniques
    - Deferred update
    - Immediate update
  - Use of Undo/Redo for recovery

Recovery Scenarios

	Deferred Update	Immediate Update
Transaction was committed before checkpoint	No Rollback (no undo) No Roll forward (no redo) No Restart	No Rollback (no undo) No Roll forward (no redo) No Restart
Transaction was committed after checkpoint	Roll forward (Redo)	Roll forward (Redo)
Transaction never committed	No Rollback (no undo) No Roll forward (no redo) Restart is needed	Rollback (Undo) Restart is needed

Assuming that database buffers are only written to disk at checkpoints.



	Deferred Update	Immediate Update
T1	Nothing; already in database	Nothing; already in database
T2	Not in database; lost; restart	Partly in database; undo
T3	Not in database; redo	Partly in database; redo
T4	Not in database; redo	Not in database; redo
T5	Not in database; lost; restart	Not in database; lost; restart

Important Topics

- **Lesson 13 - NoSQL DB: Cassandra**
  - Convert a relational database to a Cassandra database
  - Model tables based on the queries
  - Cassandra Query Language (CQL)
  - Create Table (like we did in the HW) (define the primary key)
  - Primary key = partition key + clustering columns
    (together they must be unique)
  - Partition key decides the node to access
  - Queries can only filter on attributes of the PK
  - UUID for generated keys, not simple integers or identity column

Exam Rules and Regulations

- All materials including backpacks, cell phones, wrist watches, class notes, textbooks and any other notes must be left at the front of the class. You will not have access to those items during the exam. Failure to do so will result in an NC grade.
- You must bring your own pens, pencils, erasers & sharpeners.
- Any form of communication with other students is not allowed during the exam. Every verbal and non-verbal form of communication with another student will result in the loss of one letter grade.
- In order to get any form of assistance from the proctor, you must raise your hand. No verbal communication is allowed until the proctor gives explicit permission.
- Once you leave the exam room, there is no reentry.
- You can request for a postponement of the exam if you are sick. In that case, you must produce a letter from a qualified physician.