# VanillaCore Walkthrough Part 6

Introduction to Databases

DataLab

CS, NTHU

# Outline

- Lock-Based Concurrency Control
  - 2PL
  - S2PL
  - Conservative Locking
- Code Tracing
  - S2PL in VanillaDB

# Outline

- Lock-Based Concurrency Control
  - 2PL
  - S2PL
  - Conservative Locking
- Code Tracing
  - S2PL in VanillaDB

# Lock-Based Concurrency Control

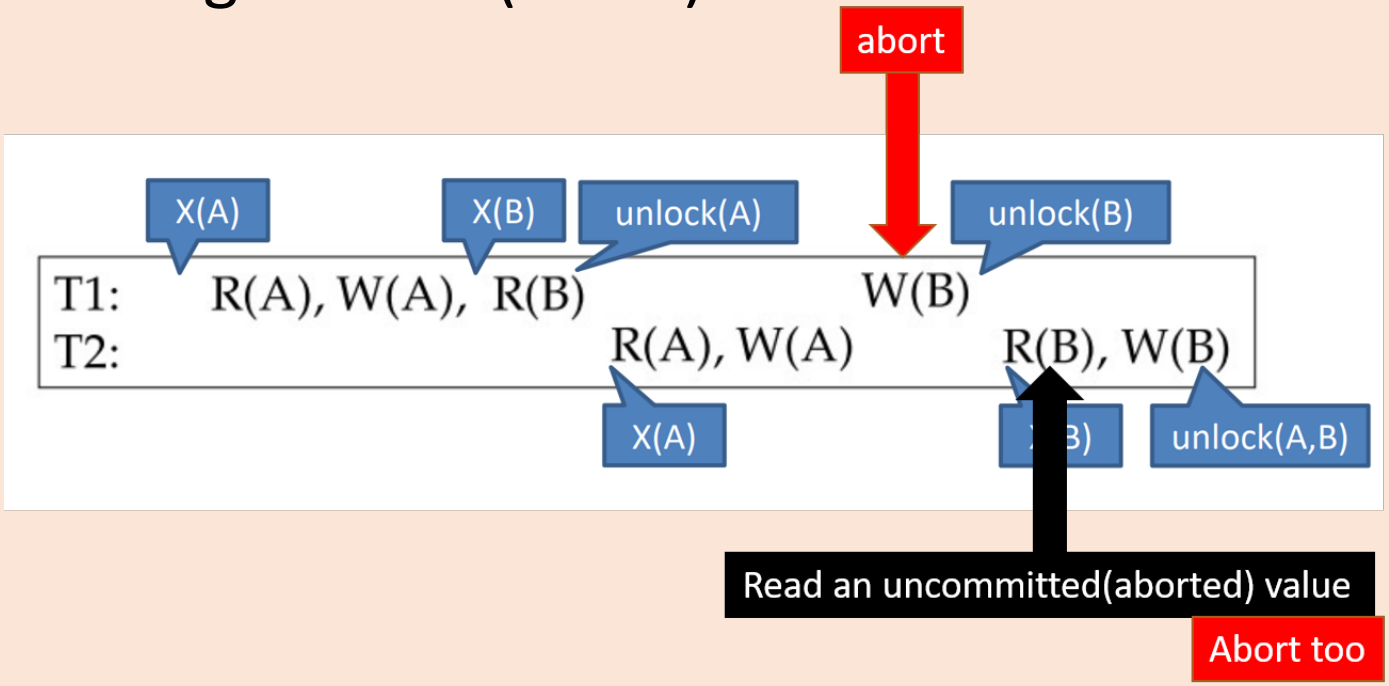
- For isolation and consistency, a DBMS should only allow serializable, recoverable schedules
  - No WR
  - No RW
  - No WW
- Locks are useful in this scenario

# 2 Phase Locking (2PL)

- 2 types of locks
  - Shared (S) lock
  - Exclusive (X) lock
- Phase 1: Growing Phase
  - Must obtain locks before read/write
- Phase 2: Shrinking Phase
  - Releases locks ☑
  - ~~Acquires locks~~ ☒

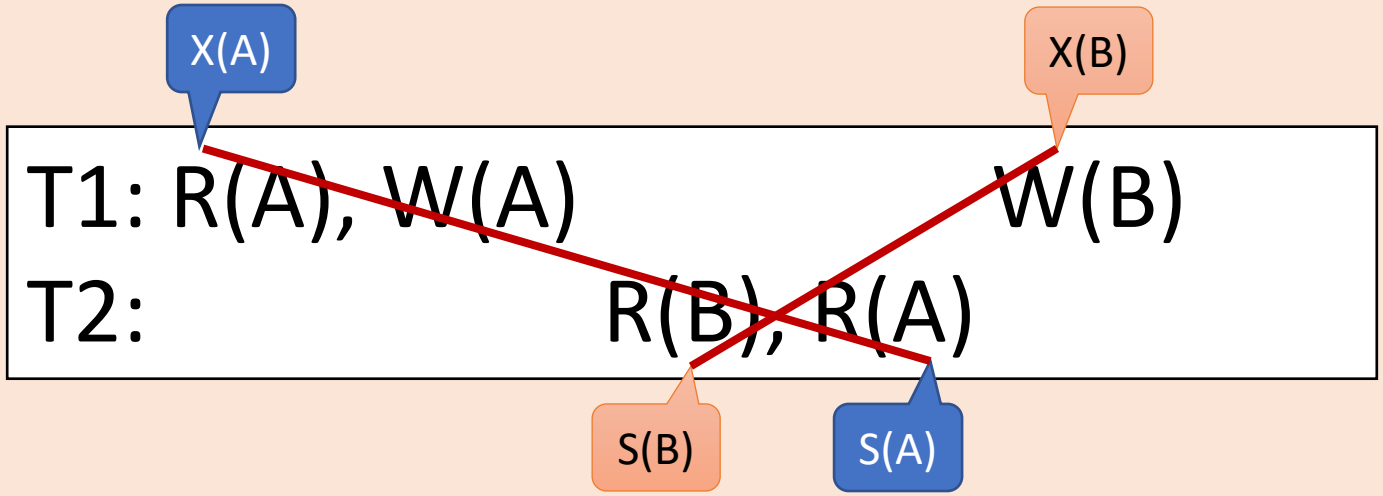
# Problems of 2PL

- Cascading rollback(abort)



# Problems of 2PL

- Deadlock



# Let’s fix cascading rollback

# Strict 2PL

- Holds all locks until the tx commits

# S2PL Example

2PL Cascading rollback

T1: R(A), W(A) , R(B) W(B)

T2: R(A), R(A) R(B), W(B)

S2PL

Release X(A), X(B)

T1: R(A), W(A) , R(B), W(B)

T2: R(A), R(A), R(B), W(B)

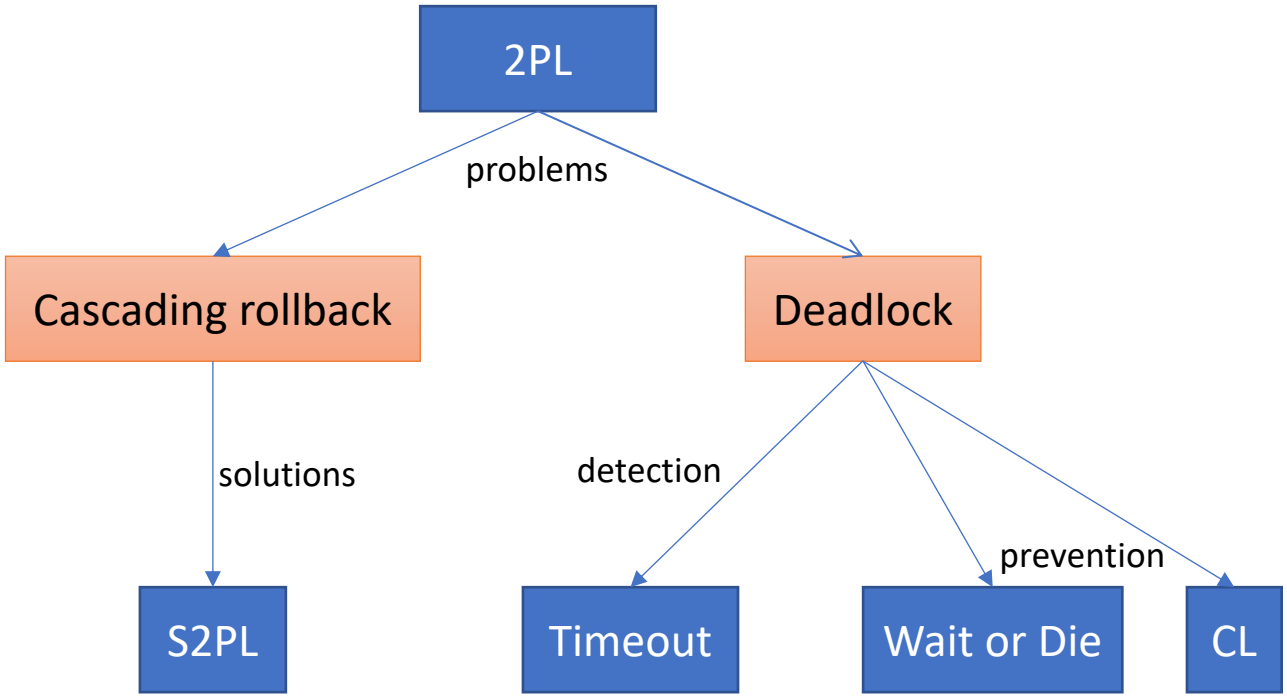
Do NOT release X(A)

# Let’s fix deadlock

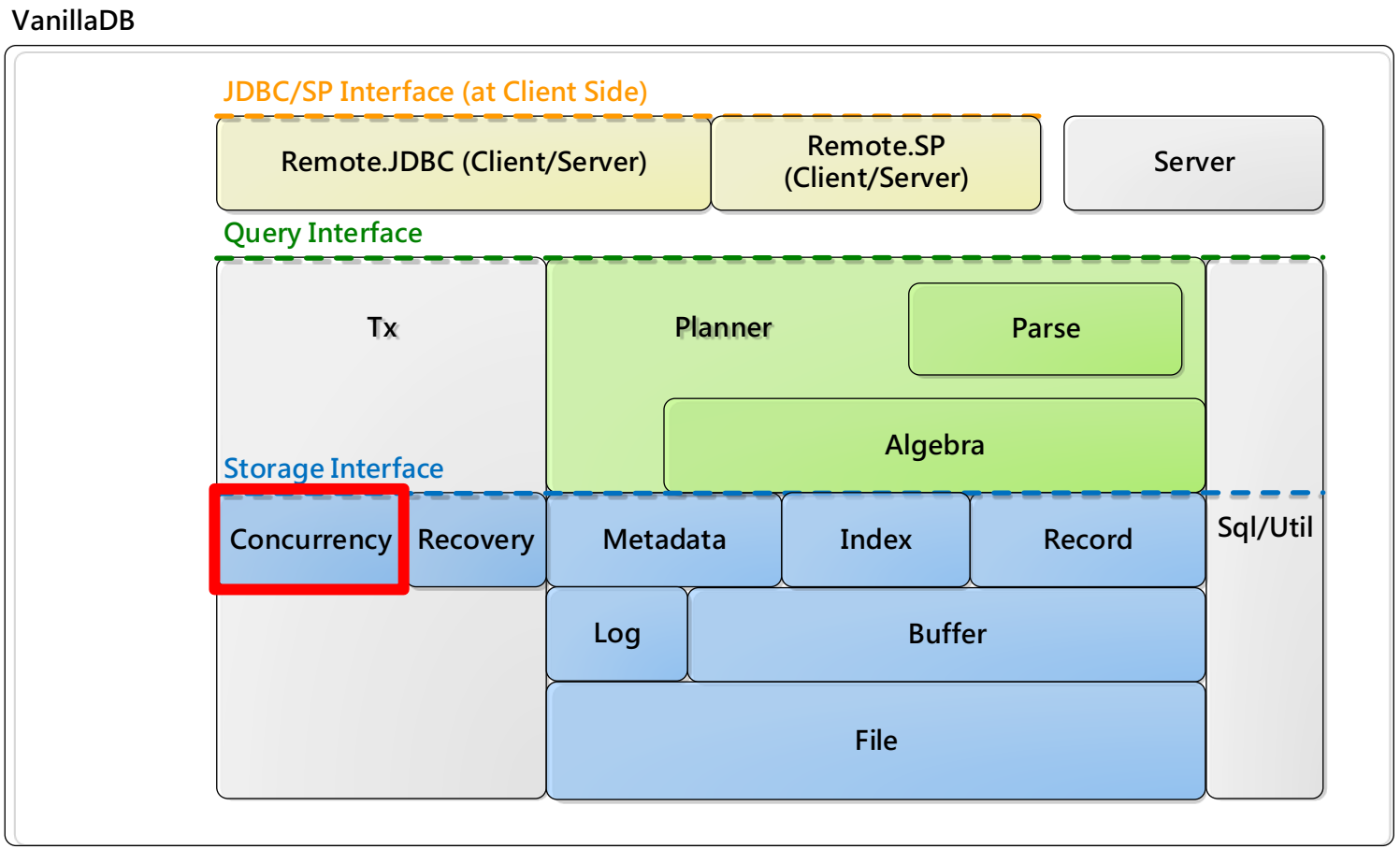
# Deadlock detection & prevention

- Timeout (deadlock detection)
- Wait-Die (deadlock prevention)
  - Tx number as ages
    - Old man wait
    - Young men go die (abort)
- Conservative locking (deadlock prevention)
  - Locks all objects at once
  - However, we may not know which objects to lock
    - Stored procedure ☑
      - we've known the read/write set
    - Ad-hoc queries ☒

# Summary of lock-based CC



# Code Tracing



# Outline

- Lock-Based Concurrency Control
  - 2PL
  - S2PL
  - Conservative Locking
- Code Tracing
  - S2PL(early lock release) in VanillaDB

# Package Structure

Singleton

tx package

Concurrency package

implement

TransactionLifecycleListener

ConcurrencyMgr

has a

LockTable

extends

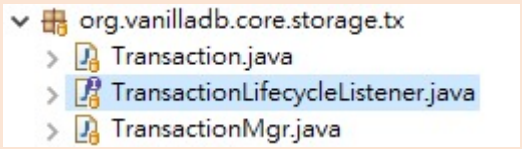
throw

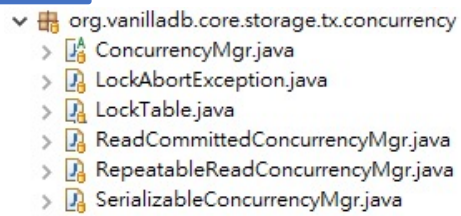
LockAbortException

ReadCommittedConcurrecyMgr

RepeatableReadConcurrecyMgr

SerializableConcurrecyMgr



# TransactionLifecycleListener

```
16 package org.vanilladb.core.storage.tx;
17
18 public interface TransactionLifecycleListener {
19
20     void onTxCommit(Transaction tx);
21
22     void onTxRollback(Transaction tx);
23
24     void onTxEndStatement(Transaction tx);
25
26 }
27
```

Release all locks on tx commit

```
public class SerializableConcurrencyMgr extends ConcurrencyMgr {

    public SerializableConcurrencyMgr(long txNumber) {
        txNum = txNumber;
    }

    @Override
    public void onTxCommit(Transaction tx) {
        lockTbl.releaseAll(txNum, false);
    }

    @Override
    public void onTxRollback(Transaction tx) {
        lockTbl.releaseAll(txNum, false);
    }
```

# Event-Driven Architecture

```
public Transaction(TransactionMgr txMgr, TransactionLifecycleListener concurMgr,
        TransactionLifecycleListener recoveryMgr, TransactionLifecycleListener bufferMgr, boolean readOnly,
        long txNum) {
    this.concurMgr = (ConcurrencyMgr) concurMgr;
    this.recoveryMgr = (RecoveryMgr) recoveryMgr;
    this.bufferMgr = (BufferMgr) bufferMgr;
    this.txNum = txNum;
    this.readOnly = readOnly;

    lifecycleListeners = new LinkedList<TransactionLifecycleListener>();
    addLifecycleListener(txMgr);
    addLifecycleListener(recoveryMgr);
    addLifecycleListener(concurMgr);
    addLifecycleListener(bufferMgr);
}
```

```
public void commit() {
    for (TransactionLifecycleListener l : lifecycleListeners)
        l.onTxCommit(this);

    if (logger.isLoggable(Level.FINE))
        logger.fine("transaction " + txNum + " committed");
}
```

# How to use concurrencyMgr?

- RecordPage

```
private Constant getVal(int offset, Type type) {
    if (!isTempTable())
        tx.concurrencyMgr().readRecord(new RecordId(blk, currentSlot));
    return currentBuff.getVal(offset, type);
}
```

- SerializableConcurrencyMgr

```
public void readRecord(RecordId recId) {
    lockTbl.isLock(recId.block().fileName(), txNum);
    lockTbl.isLock(recId.block(), txNum);
    lockTbl.sLock(recId, txNum);
}
```