CS3DB3/SE4DB3/SE6M03 TUTORIAL

Mei Jiang

March 20/22, 2013

Outline

- Timeline
- Types of Schedules
  - Serializable
  - Conflict Serializable
  - Recoverable
  - Avoid Cascading Abort (ACA)
  - Two Phase Locking Protocol (2PL)
  - Strict Two Phase Locking (Strict 2PL)
- Example

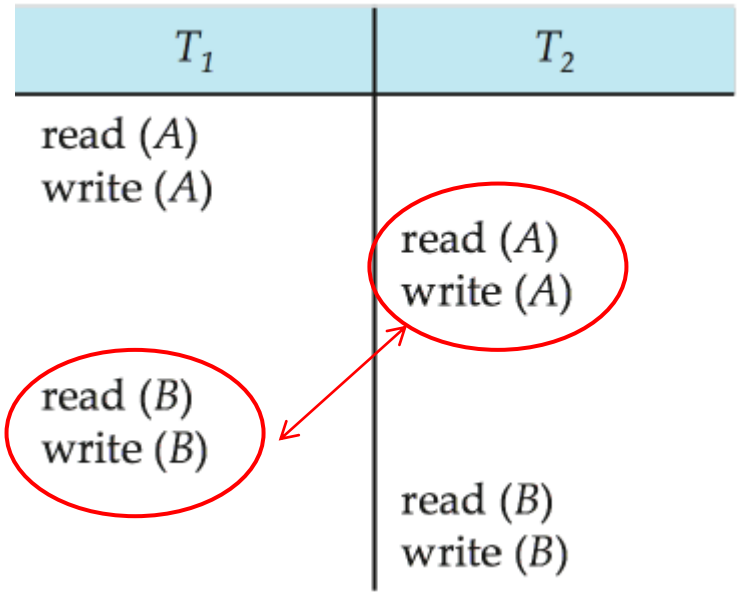
Timeline

- March 20/22
  - Types of schedules review
- March 27(Term test 2)/29(Good Friday, no class)
  - No tutorial this week!
- April 3/5
  - Assignment 2 solutions
  - Term test 2 solutions (maybe)
- April 10 (last day of class)
  - Tutorial becomes office hour at ITB116, 10:30-11:20

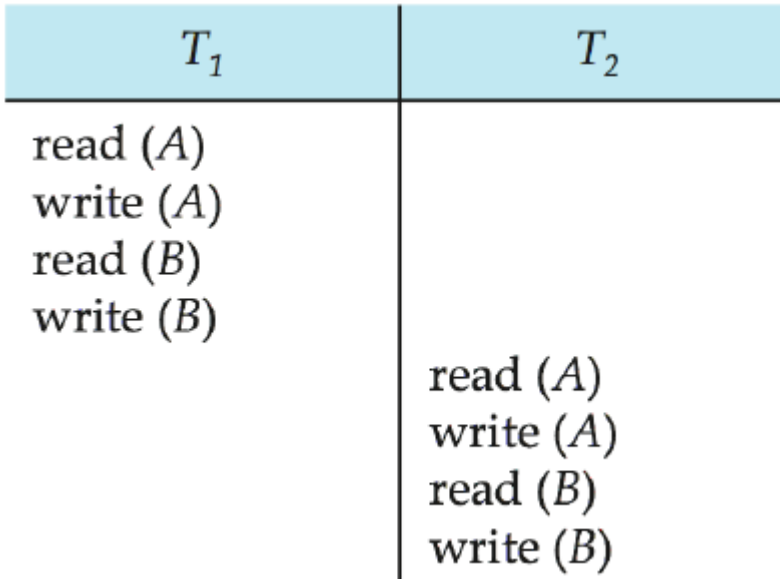
Serializable

- Schedule: a set of the operations from a set of transactions, preserving the order of the operations for each transaction.
- Serial Schedule: Schedule that does not interleave the operations of different transactions.
- Equivalent Schedules: For any database state, the effect (on the set of objects in the database) of executing the first schedule is identical to the effect of executing the second schedule.
- Serializable Schedule: A schedule that is equivalent to some serial execution of the transactions.

Serializable (cont.)



S1



S2

S1 is serializable since the net effect of S1 and S2 are the same.

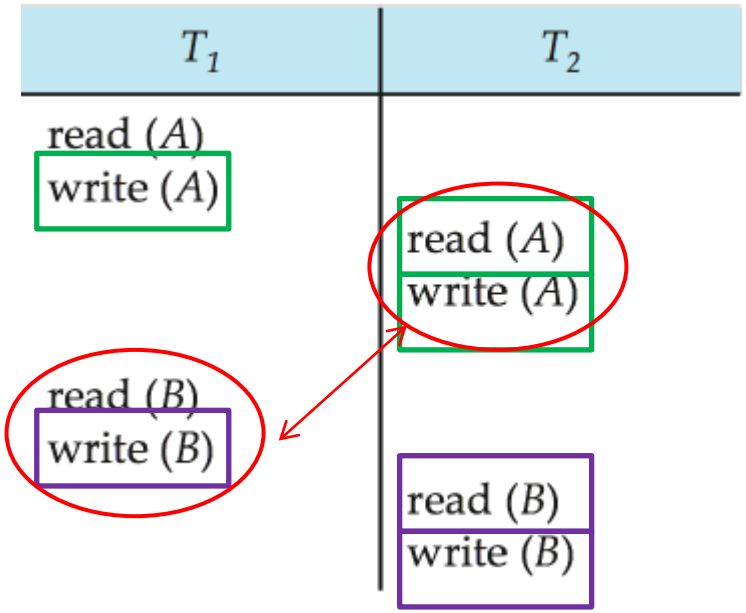
Conflict Serializable

- Conflict Operations: two operations in a schedule are said to be conflict if they satisfy all three of the following conditions:
  1. They belong to different transactions;
  2. They access the same item A;
  3. At least one of the operations is a write(A).
- Conflicts types:
  - Reading uncommitted data (WR), "dirty reads"
  - Unrepeatable reads (RW)
  - Lost Update (WW)

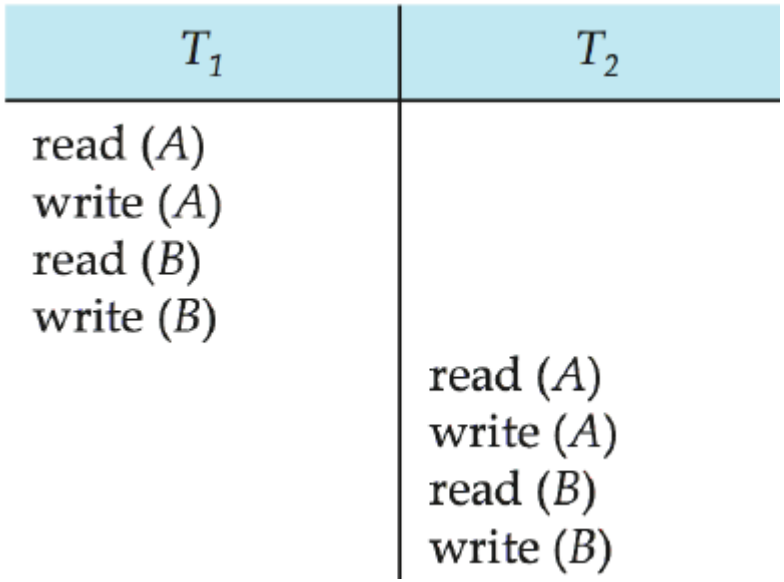
Conflict Serializable (cont.)

- If two consecutive operations in a schedule do not conflict, their results would remain the same even if they had been interchanged in the schedule.
- If a schedule S can be transformed into a schedule S´ by a series of swaps of non-conflicting instructions, we say that S and S´ are Conflict Equivalent.
- A schedule is Conflict Serializable if it is conflict equivalent to some serial schedule.
- Every conflict serializable schedule is serializable.

Conflict Serializable (cont.)



S1



S2

S1 and S2 are conflict equivalent, and S1 is conflict serializable .

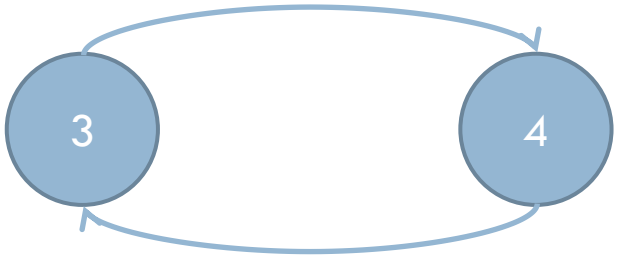
Conflict Serializable (cont.)

- We can use Precedence Graph Test to determine conflict serializable schedule. If the graph has no cycles, then it is conflict serializable.

T_1	T_2
read (A)	
write (A)	
	read (A)
	write (A)
read (B)	
write (B)	
	read (B)
	write (B)

YES

T_3	T_4
read (Q)	
	write (Q)
write (Q)	



NO

Recoverable

- Recoverable Schedule: A Xact commits only after all the Xacts it "depends on" (i.e. it reads from) commit.

T_8	T_9
read (A)	
write (A)	
	read (A)
	commit
read (B)	

Not Recoverable, since T9 depends on T8, and T9 commits while T8 has not.

Avoid Cascading Abort (ACA)

- Avoid Cascading Abort (ACA): A Xact only reads data from committed Xacts.
- ACA implies Recoverable, but not vice-versa.

T_8	T_9
read (A)	
write (A)	
read (B)	
commit	
	read (A)
	commit

ACA and Recoverable

T_8	T_9
read (A)	
write (A)	
read (B)	
	read (A)
commit	
	commit

Recoverable, but not ACA

Locking Protocol

- Lock: a mechanism to control concurrent access to a data object.
- Types of locks
  - Shared (S) lock: for reading
  - Exclusive (X) lock: for writing, and of course, also for reading
- Locking Protocol: is a set of rules followed by all transactions while requesting and releasing locks.
- Locking protocols restrict the set of possible schedules.

Two Phase Locking Protocol (2PL)

- Two rules:
  1. Each Xact must obtain a S (shared) lock on object before reading, and an X (exclusive) lock on object before writing.
  2. A transaction cannot request additional locks once it releases any lock.
- Each transaction is executed in two phases:
  - Phase 1: Growing phase
    - transaction may obtain locks, may not release locks
  - Phase 2: Shrinking phase
    - transaction may release locks, may not obtain locks
- Allow Xacts to release locks before the end (commit/abort).
- Transactions can be serialized in the order of their lock points (i.e. the point where a transaction acquired its final lock).

Strict Two Phase Locking (Strict 2PL)

- "Stricter" version of 2PL
  - Locking are done in two phases as 2PL.
  - All locks taken by a transaction are held until that transaction commits/aborts.
- No other transaction can see or modify the data object until the transaction holding the lock on it is complete.

Example

- $W_1(X), R_2(X), W_2(X), W_1(Y), C_2, C_1$
  - Serializable — YES
  - Conflict Serializable — YES
  - Recoverable — NO
  - ACA — NO
  - 2PL — **NO**
  - Strict 2PL — NO

It's not 2PL because $W_1(Y)$ cannot request X lock on Y after it releases the lock on X to $R_2(X)$ and $W_2(X)$, since it's already in shrinking phase.

Note: After talk to Dr. Chiang, the answer is changed to be NO for simplicity reason and consistent with the revised answer of question 6 in Practice Term Test 2.

References

- Database System Concepts (6th edition) by A. Silberschatz, H. Korth, S. Sudarshan
- http://www.cas.mcmaster.ca/~fchiang/courses/db3/index.html