Database Concurrency Control

Robert M. Keller
Harvey Mudd College
April 2004

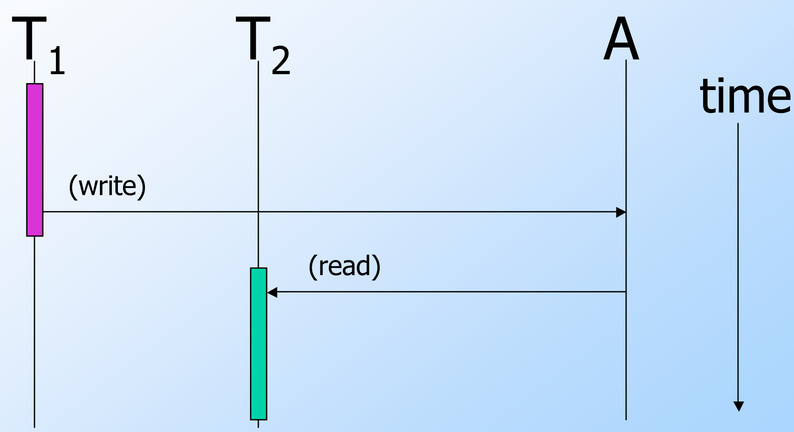
1

Modeling with Sequence Diagrams

- Sequence diagrams are from UML
- Each line in a diagram corresponds to some object, in our case
  - Database items (relations, tuples, …)
  - Transaction
- We use message direction to show what is being read vs. written, **not** the originator of the message. (All messages originate from transactions.)

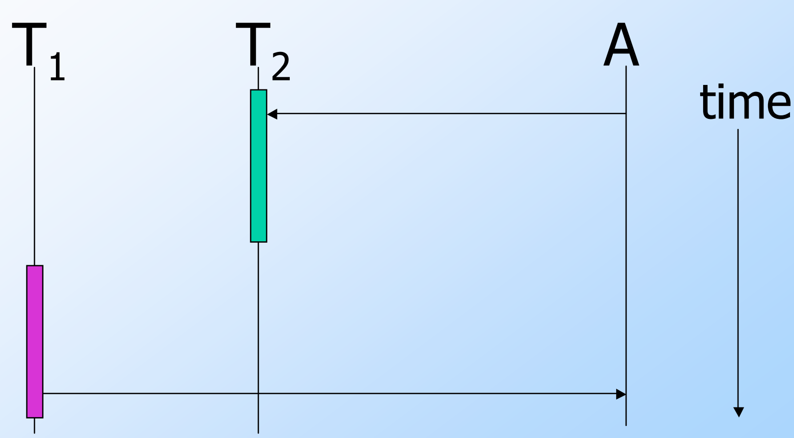
2

Example 1



T_1 writes A, (then) T_2 reads A

Example 2

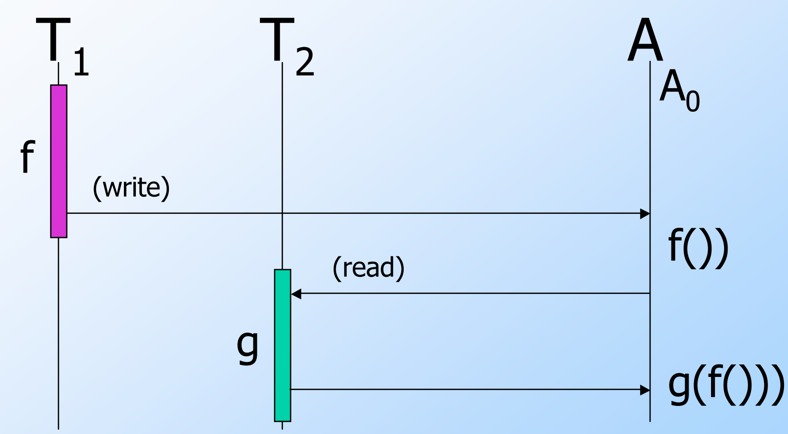


T_2 reads A, T_1 writes A

Assumption

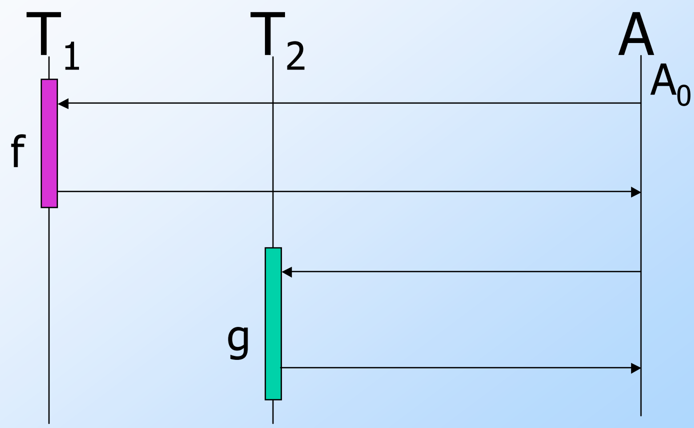
- When a transaction writes, the value it writes is an unknown (but fixed) **function** of the values it has read (and only those).
- In Example 1, T_1 writes f() to A, then T_2 reads f() as the value of A.
- In Example 2, T_2 reads A_0 (the initial value) as the value of A, then T_1 writes f() to A.

Example 3



Final value of A is g(f())

Example 4



Final value of A is _______

7

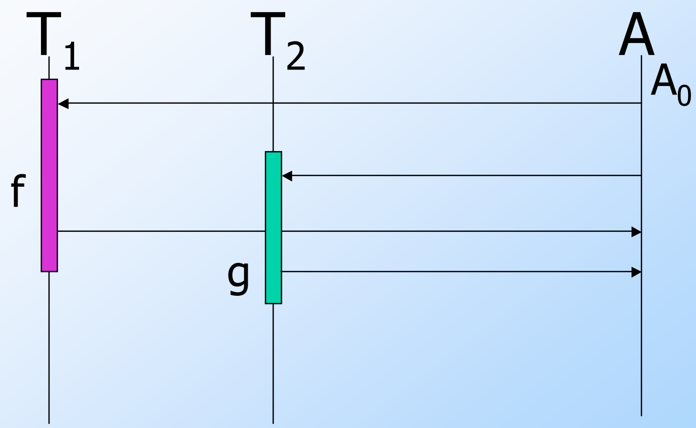
Possible Phenomena

- Lost Update
- Inconsistent Retrieval

8

Lost Update Example

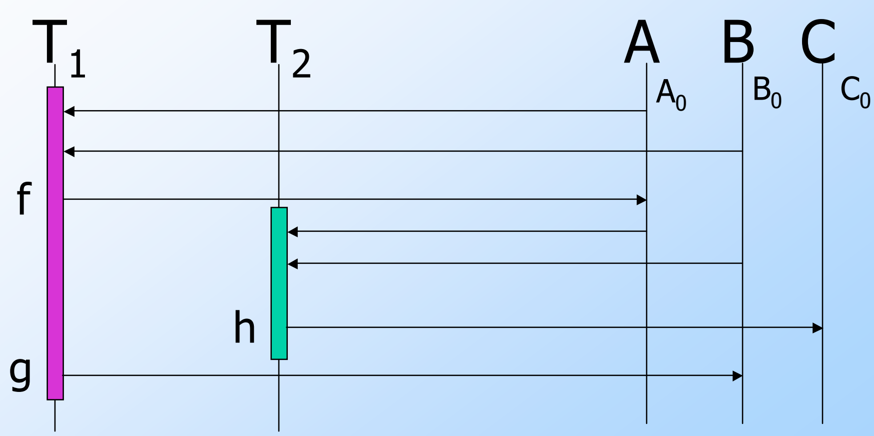
(compare Example 3)



Final value of A is ______

9

Inconsistent Retrieval Example



Final value of C is ______

10

Correctness

- A set of transactions is assumed to operate **correctly** if the net effect is the same as ***some*** **serial** execution.

11

Interleavings

- An interleaving consists of the steps of all transactions in some order.
- All steps of each transaction must be included.
- Within each transaction, the steps are in the same order as in the serial case.

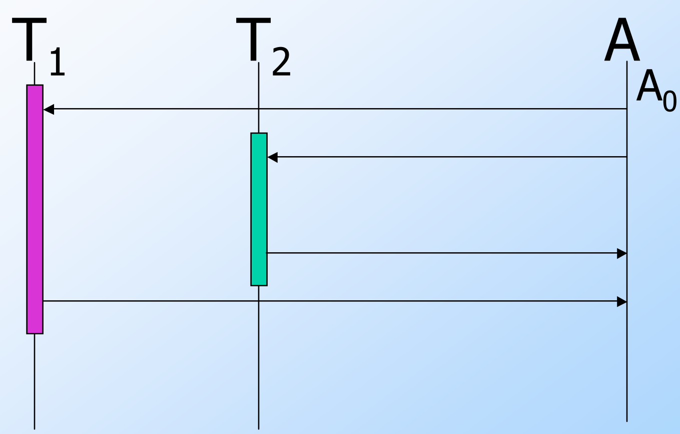
12

Serializable Interleaving

- An interleaving is serializable if the net effect is the same as some serialization of the transactions.

13

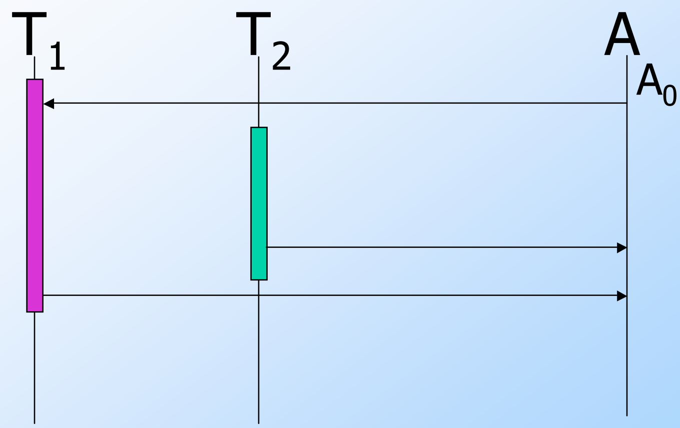
Is this interleaving serializable?



Equivalent to

14

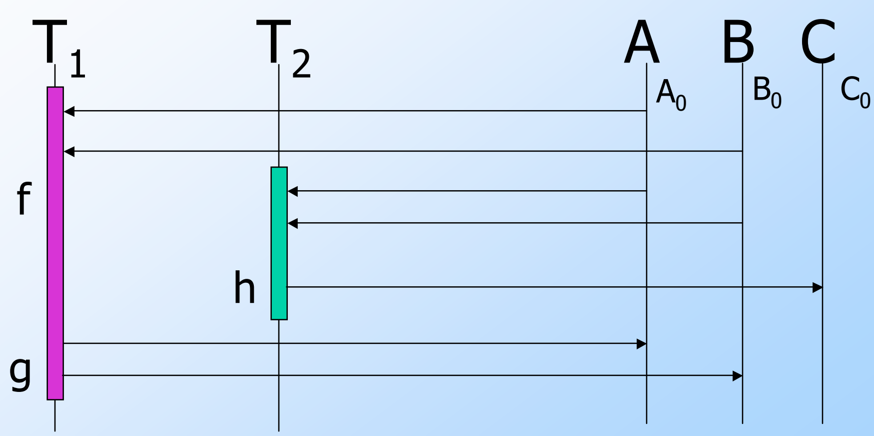
How about this one?



Equivalent to

15

And This?



Equivalent to

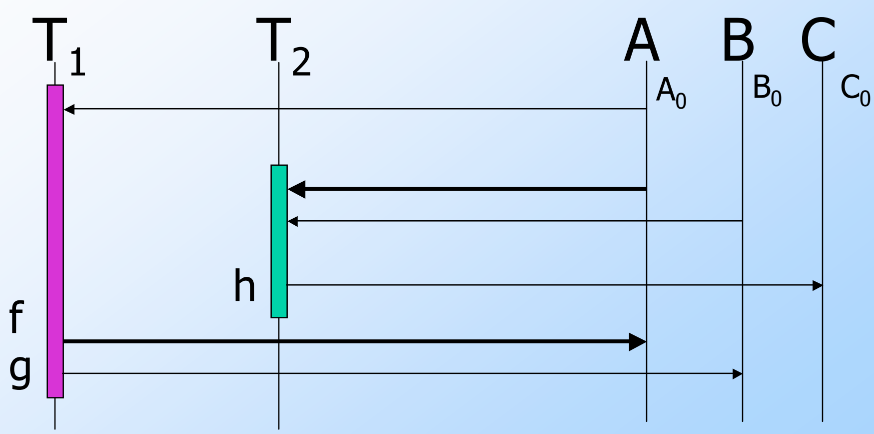
16

Conflicts

- Two operations of different transactions conflict if either:
  - One reads from a data item to which the other reads (R-W conflict)
  - One writes to a data item to which the other writes (W-W conflict)
- If two overlapping transactions have conflicting operations, then there is an order-dependence among the operations.
- We want to preserve such an order in any serialized interleaving.

17

Conflict Example



Must preserve the order T_2T_1

18

Conflict Analysis

- When two operations in different transactions conflict, the order of those two operations must be **preserved** in any **equivalent** interleaving.

- We may have "conflicting" requirements of this form.

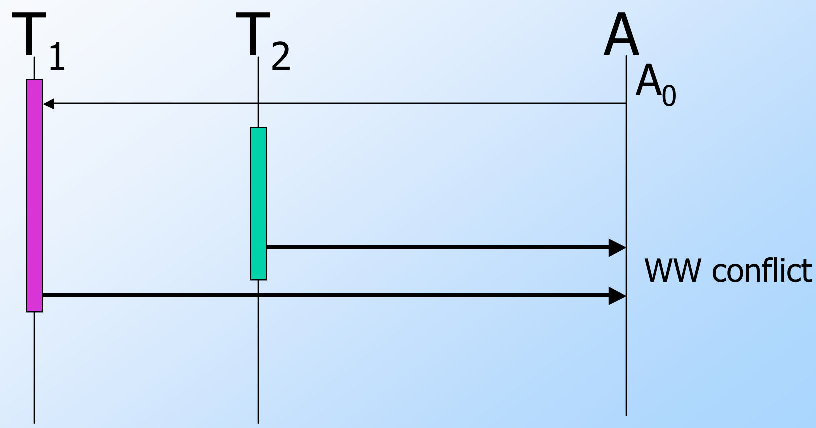
19

Visualizing Conflicts

- Scan the read/write sequences for each data item.
- If a write operation to the item is present with other reads or writes in different transactions, then there is a conflict between those transactions.

20

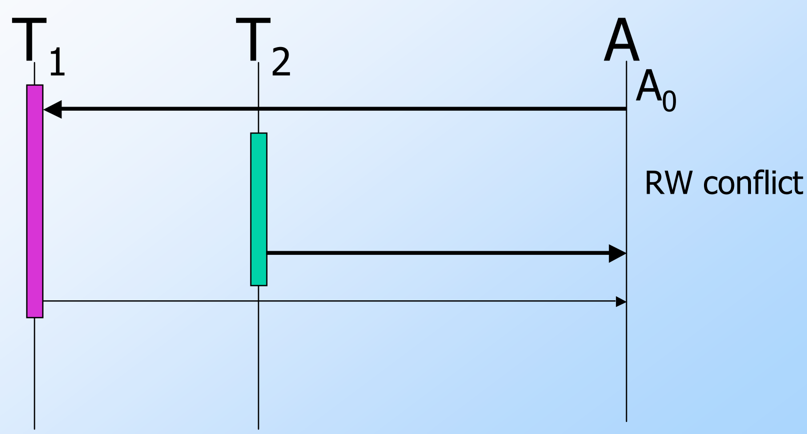
Conflict Orders



Requires T_2T_1.

21

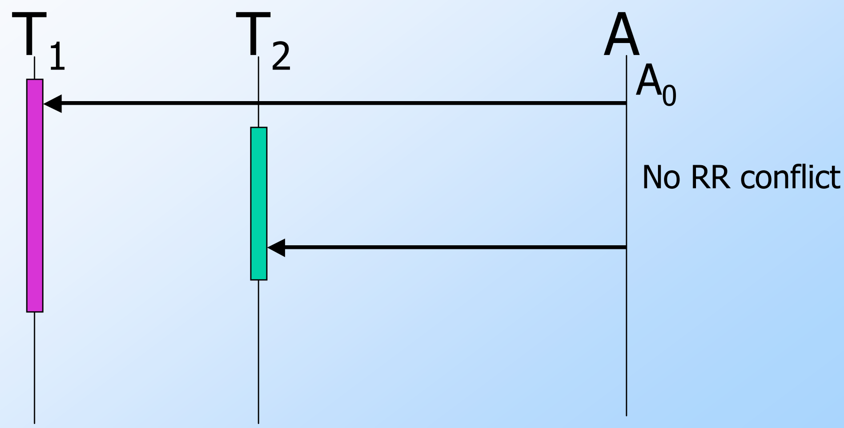
Conflicting Conflict Orders



Requires T_1T_2 ; we can’t have this and T_2T_1.

22

Non-Conflict Orders



Requires nothing.

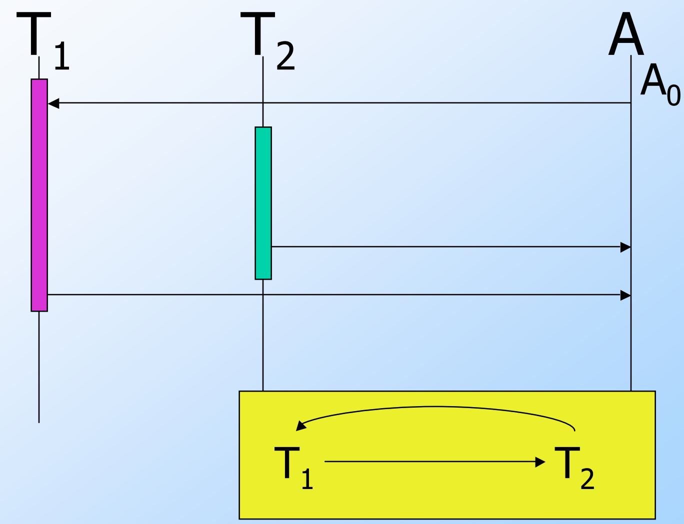
23

Conflict Graph for an Interleaving

- The conflict requirements can be expressed as a graph:
  - Transactions are nodes
  - There is an arrow if the transactions have conflicting operations, indicating the necessary serial order for equivalence with the interleaving.

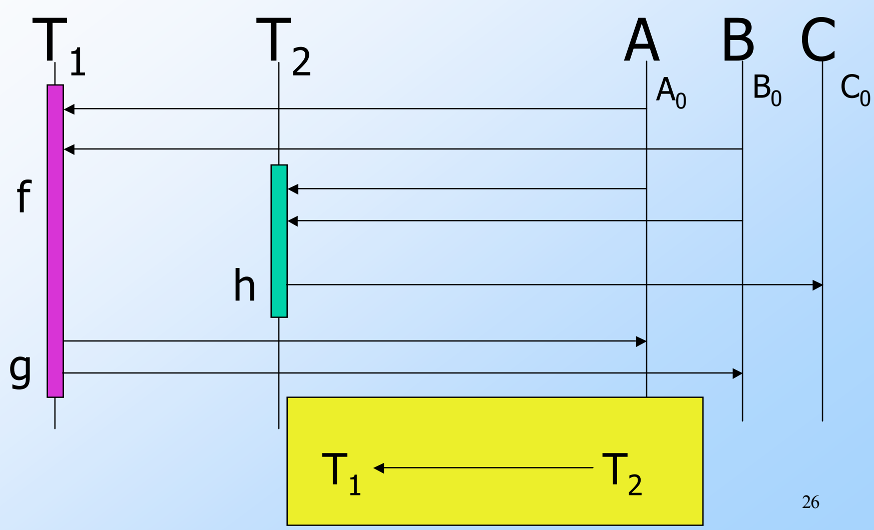
24

Conflict Graph Example



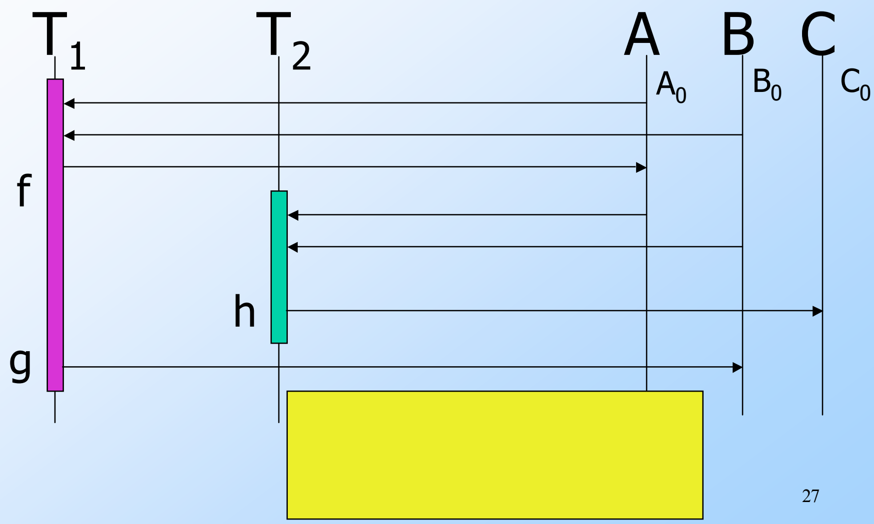
25

Conflict Graph Example



26

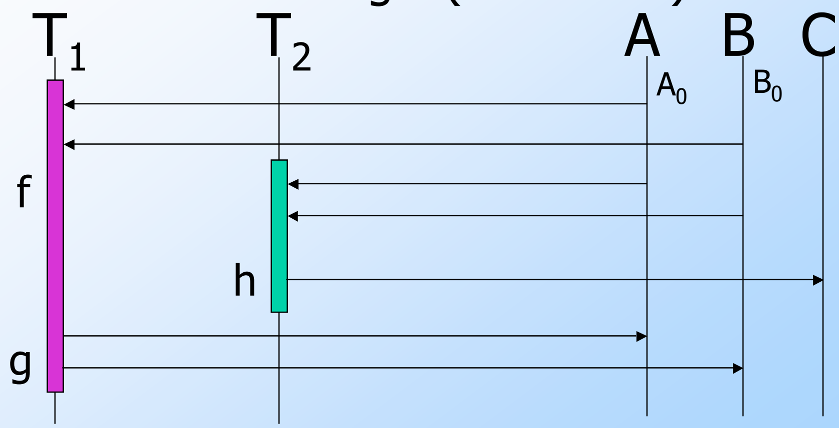
Conflict Graph Example



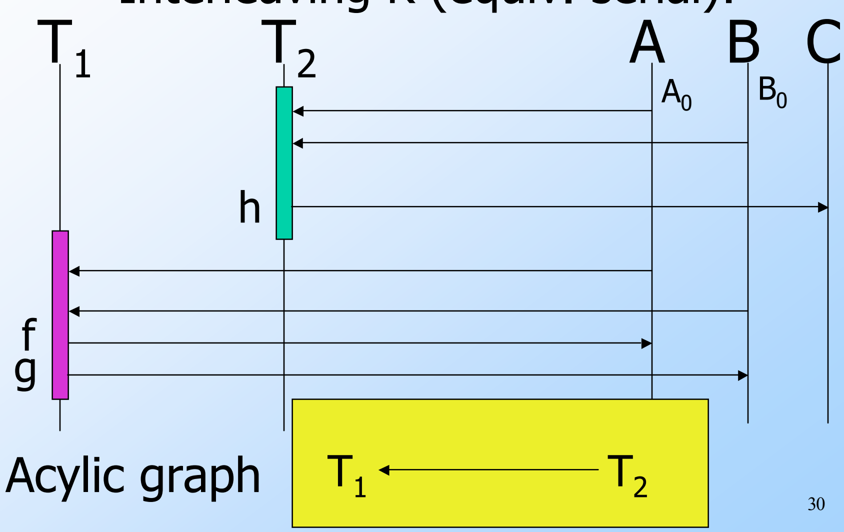
Result 1

- If there is a serializable interleaving, then the conflict graph is acyclic.
- Proof:
  - Assume that there is a serializable interleaving J.
  - Let K be a *serial* interleaving equivalent to J.
  - If operations in two different transactions conflict in K, then the implied arrows in the conflict graph must all be the same direction, since **one transaction occurs before the other** in K.
  - Therefore, if there is an arrow from T_i to T_j in the conflict graph, it must be the case that T_i **precedes** T_j in K.
  - But K is a *linear* order, so there can be no cycle.

Example of Result 1: Interleaving J (non-serial):



Example of Result 1: Interleaving K (equiv. serial):

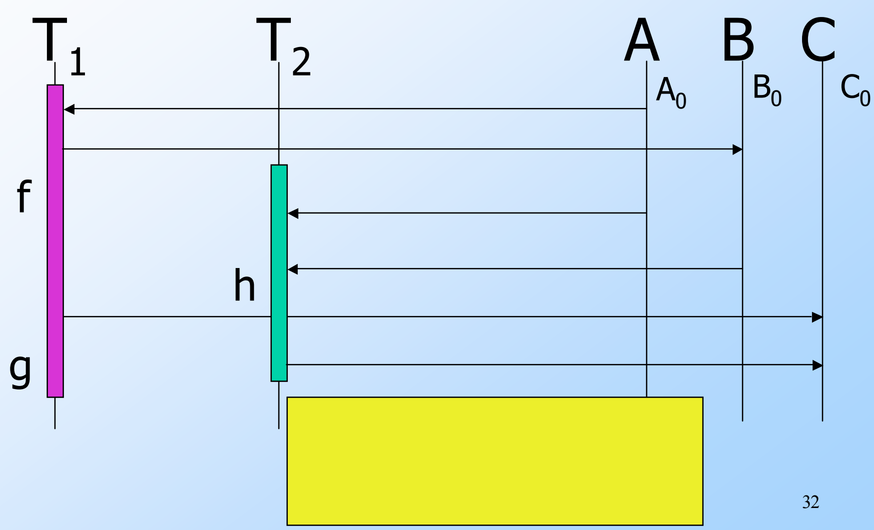


Result 2

- If the conflict graph is acyclic, then the interleaving is serializable.
- Proof:

31

Example of Result 2:



32

Summary of Results 1&2

- An interleaving is serializable iff the conflict graph is acyclic.

33

Serializability Requirement

- The execution of a set of transactions must be serializable.

34

Alternatives for Ensuring Serializability

- Allow only one transaction at a time.
- Some more liberal policy that guarantees serializability.
- Analyze transactions statically; only allow compatible transactions to run concurrently.
- Construct conflict graph dynamically; abort (rollback) a transaction if it can create a cycle.
- Use locking.
- Use time stamps.

35

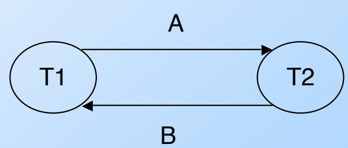
Locking

- Simple model:
  - Only one kind of lock
  - Lock data item before any read or write
  - Unlock data item after reads and writes
- By itself, doesn't guarantee serializability:
  - Example?
- Can produce deadlocks if no further protocol imposed.

36

Locking Implies Reading or Writing

T1	**T2**
LOCK A	
UNLOCK A	
	LOCK A
	UNLOCK A
	LOCK B
	UNLOCK B
LOCK B	
UNLOCK B	



37

2-Phase Locking Policy (2PL)

Every transaction is divided into two phases that occur in sequence:

Phase I: All requesting of locks (no releasing)

Phase II: All releasing of locks (no further requesting)

[Similar-sounding, but distinct idea: **2-Phase Commit** used in distributed databases.]

38

2PL Theorem

- Under 2-phase locking, every interleaving is serializable.
- Proof:
  - Define the **lock point** of a transaction within an interleaving to be the point at which it acquires the last of the locks it requests.
  - Order the transactions by lock point, earliest to latest.
  - **Claim**: The serial interleaving in which transactions are done in lock point order is equivalent to the original interleaving.

39

Proof of Claim

- Claim: The serial interleaving in which transactions are done in lock point order is equivalent to the original interleaving.

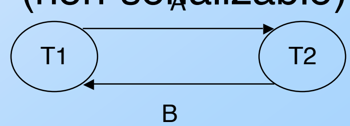
- Proof: If T_i is before T_j in the lock point ordering,then T_j cannot read or write an item A until after T_i has released the lock on A.
- Therefore, in the conflict graph there can be no arrow from T_j to T_i.
- Hence the conflict graph is a partial order consistent with the linear lock point order. In particular, the conflict graph is acyclic.

40

Non-2PL Example

T1	T2
LOCK A	
UNLOCK A	
	LOCK A
	UNLOCK A
	LOCK B
	UNLOCK B
LOCK B	
UNLOCK B	

conflict graph
(non-serializable)



41

2PL Example

(lock points underlined)

T1	T2	T3
Lock A		
Lock B		
	Lock A	Lock B
<u>Lock C</u>	(waiting)	(waiting)
Unlock A,B,C	|	|
	|	(B acquired)
	Lock C	<u>Lock C</u>
	(waiting)	Unlock B, C
	<u>(C acquired)</u>	
	Unlock A,C	

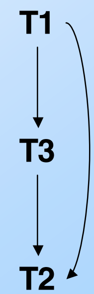
42

2PL Example

(lock points underlined)

T1	T2	T3
Lock A		
Lock B		
	Lock A	Lock B
Lock C	(waiting)	(waiting)
Unlock A,B,C	I	I
	I	(B acquired)
	Lock C	Lock C
	(waiting)	Unlock B, C
	(C acquired)	
	Unlock A,C	

conflict graph



consistent
with lockpoint
linear order

43

Same Transactions, Different Lock Point Order

T1	T2	T3
Lock A	Lock A	Lock B
(waiting)	Lock C	Lock C
I	Unlock A, C	(waiting)
(A acquired)		(C acquired)
Lock B		
(waiting)		Unlock B, C
(B acquired)		
Lock C		
Unlock A, B, C		

conflict graph

44

Deadlock Danger

- 2PL itself does not avoid deadlock
  - Example?
- Need to impose additional mechanism to prevent deadlock

45

Serialization with Read- and Write-Locks

- So far, we have assumed all locks are the same (= write-locks).
- This is overly-restrictive:
  inhibits concurrent reads.
- When both types of locks are used, the cycle criterion for serializability can be applied.
- The *conflict graph construction is modified*, however.
- 2PL still works!

46

Modified Conflict Graph Construction (for read/write locks)

- If Ti read- or write-locks A, and Tj is the **next** transaction in the interleaving to write-lock A, then there is an arc from Ti to Tj.

- If Ti write-locks A, and Tk read-locks A after Ti unlocks A, but before any other transaction write-locks A, then there is an arc from Ti to Tk.

Example: Is this serializable?

T1	T2	T3	T4
		Wlock A	
			Rlock B
		Unlock A	
Rlock A			
			Unlock B
		Wlock B	
	Rlock A		
		Unlock B	
Wlock B			
	Unlock A		
Unlock A			
			Wlock A
Unlock B			
	Rlock B		
			Unlock A
	Unlock B		