Database

Transaction Management
/Concurrency Control
Overview

Outline

1. Transaction
2. ACID Property
3. Schedule
4. Serial Schedule
5. Conflict Serializability
6. View Serializability
7. Strict Two-Phase Locking
8. A question

Transaction Concept

- A ***transaction*** is a *unit* of program execution that accesses and possibly updates various data items.

- A transaction must see a consistent database.
  - During transaction execution, the database may be inconsistent.
  - When the transaction is committed, the database must be consistent.

Transaction example

- Transaction to transfer $50 from account *A* to account *B*:
  1. **read**(*A*)
  2. *A* := *A* – 50
  3. **write**(*A*)
  4. **read**(*B*)
  5. *B* := *B* + 50
  6. **write**(*B*)

- Consistency requirement – the sum of *A* and *B* is unchanged by the execution of the transaction.

- Atomicity requirement — if the transaction fails after step 3 and before step 6, the system should ensure that its updates are not reflected in the database, else an inconsistency will result.

Outline

1. Transaction
2. ACID Property
3. Schedule
4. Serial Schedule
5. Conflict Serializability
6. View Serializability
7. Strict Two-Phase Locking
8. A question

2. ACID Property

- Atomicity
  - In a transaction, either all operations are carried out or none are.
- Consistency
  - Regardless of other transactions, each transaction must preserve the consistency of the database
- Isolation
  - User can understand a transaction without considering the effect of other transactions
- Durability
  - The effect of transaction should persist forever whenever the transaction is completed/committed.

Outline

1. Transaction
2. ACID Property
3. Schedule
4. Serial Schedule
5. Conflict Serializability
6. View Serializability
7. Strict Two-Phase Locking
8. A question

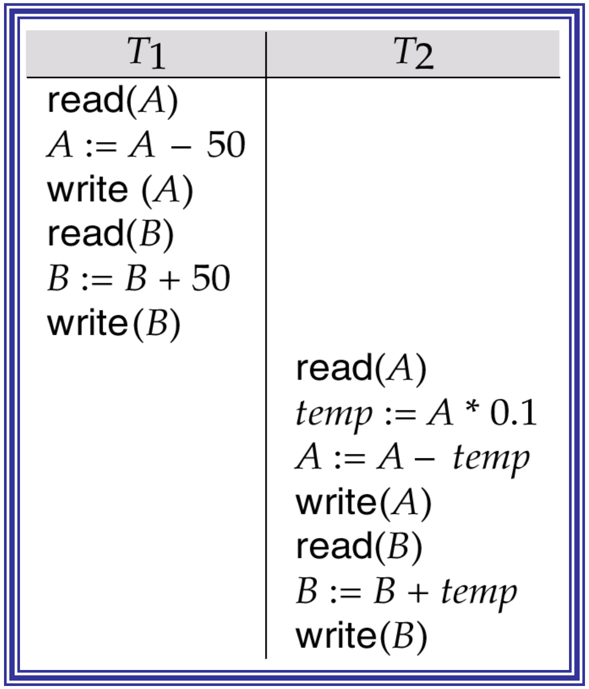
3. Schedules

- Schedule
  - A sequence of operations in a set of transactions $\{T_1, T_2, \ldots, T_n\}$
  - E.g., a set of transactions is $\{T_1, T_2\}$,

Example Schedule

- Let T_1 transfer \$50 from A to B, and T_2 transfer 10% of the balance from A to B. The following is a schedule, in which T_1 is followed by T_2.

Schedule 1

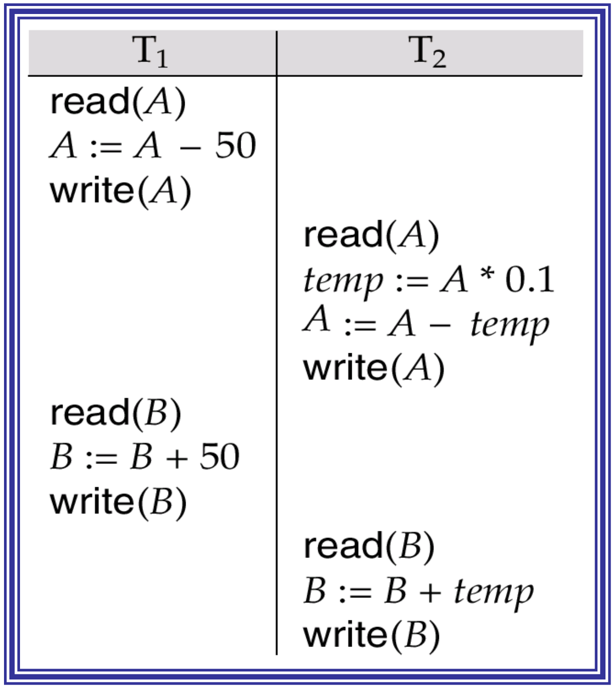


T_1	T_2
read(A)	
$A := A - 50$	
write (A)	
read(B)	
$B := B + 50$	
write(B)	
	read(A)
	$temp := A * 0.1$
	$A := A - temp$
	write(A)
	read(B)
	$B := B + temp$
	write(B)

Example Schedule (Cont.)

- Let T_1 and T_2 be the transactions defined previously. The following schedule 2 is *equivalent* to Schedule 1.

Schedule 2



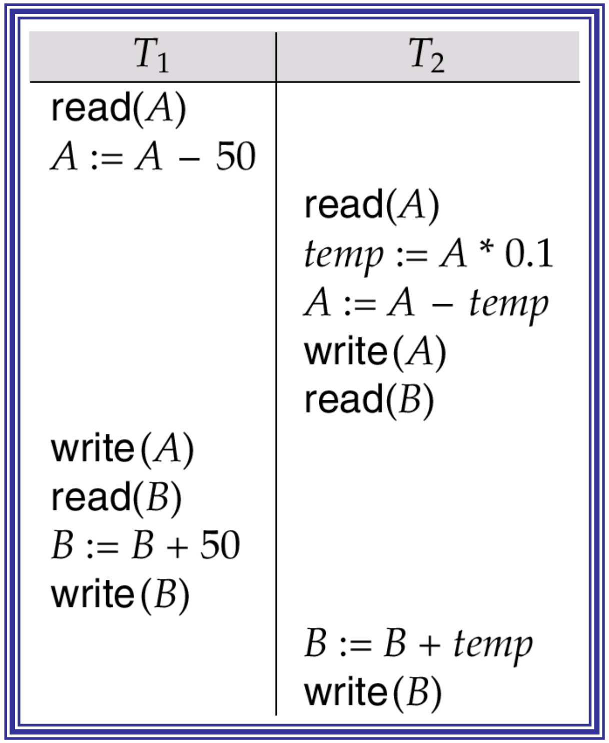
T_1	T_2
read(A)	
$A := A - 50$	
write(A)	
	read(A)
	$temp := A * 0.1$
	$A := A - temp$
	write(A)
read(B)	
$B := B + 50$	
write(B)	
	read(B)
	$B := B + temp$
	write(B)

In both Schedule 1 and 2, the sum A + B is preserved.

Example Schedules (Cont.)

- The following concurrent schedule 3 does not preserve the value of the sum *A* + *B*.

schedule 3



T_1	T_2
read(A)	
$A := A - 50$	
	read(A)
	$temp := A * 0.1$
	$A := A - temp$
	write(A)
	read(B)
write(A)	
read(B)	
$B := B + 50$	
write(B)	
	$B := B + temp$
	write(B)

Outline

1. Transaction
2. ACID Property
3. Schedule
4. Serial Schedule
5. Conflict Serializability
6. View Serializability
7. Strict Two-Phase Locking
8. A question

4. Serial Schedules

- Serial schedule
  - A schedule which the operations belonging to one single transaction appear together
  - E.g. H_1 is a serial schedule (to T_1T_2)
    H_2 and H_3 are not serial schedule
- Serializable schedules
  - equivalent to some serial schedule
  - E.g. H_1 and H_2 are serializable schedules (to T_1T_2)
  - H_3 is a serializable schedule (to T_2T_1).

H_1:

T_1:	R(A),	W(A),		
T_2:			R(B),	W(B)

H_2:

T_1:	R(A),		W(A),	
T_2:		R(B),		W(B)

H_3:

T_1:		R(A),		W(A)
T_2:	R(B),		W(B),	

4. Serial Schedules

- E.g. T_1 : R(A), W(A)
  T_2 : R(B), W(B)

H_4:

T_1:			R(A),	W(A)
T_2:	R(B),	W(B),		

- Is schedule H_4 a serial schedule?

  Yes.

  H_4: T_2 T_1

4. Serial Schedules

- If T_3: Read(A), A=A+1, Write(A)
  T_4: Read(A), A=A+1, Write(A)
- consider a serial schedule (T_3 T_4):

T_3	T_4	Value of A in DB
Read(A) 5		5
A=A+1 6		5
Write(A) 6		6
	Read(A) 6	6
	A = A+1 7	6
	Write(A) 7	7

4. Serial Schedules

- If T_3: Read(A), A=A+1, Write(A)
  T_4: Read(A), A=A+1, Write(A)
- consider a non-serial schedule:

T_3	T_4	Value of A in DB
Read(A) 5		5
	Read(A) 5	5
A=A+1 6		5
	A = A+1 6	5
	Write(A) 6	6
Write(A) 6		6

Outline

1. Transaction
2. ACID Property
3. Schedule
4. Serial Schedule
5. Conflict Serializability
6. View Serializability
7. Strict Two-Phase Locking
8. A question

5. Conflict Serializability

- Two operations are **conflict** if
  1. They are operations of different transactions on the **same** data object
  2. At least one of them is a **Write** operation
  - E.g.

T_i:	W(X),	
T_j:		R(X)

T_i:	R(X),	
T_j:		W(X)

T_i:	W (X),	
T_j:		W(X)

- Two operations are **non-conflict**
  - E.g.

T_i:	R(X),	
T_j:		R(X)

T_i:	W(X),	
T_j:		R(Y)

T_i:	R(X),	
T_j:		W(Y)

T_i:	W (X),	
T_j:		W(Y)

5.1 Conflict Equivalent

- Two schedules S_1 and S_2 are **conflict equivalent** if
  - S_1 and S_2 involve the same operations of the same transaction
  - Every pair of conflicting operations is ordered in the same way in S_1 and S_2

□ Two schedules involve the same operations of the same transaction

□ Every pair of conflicting operations is ordered in the same way in two schedules

5.1 Conflict Equivalent

- E.g.1. Is H_8 and H_9 conflict equivalent?

H_8:

T_1:	W (X),		R(Y),	
T_2:		R(Y),		R(X)

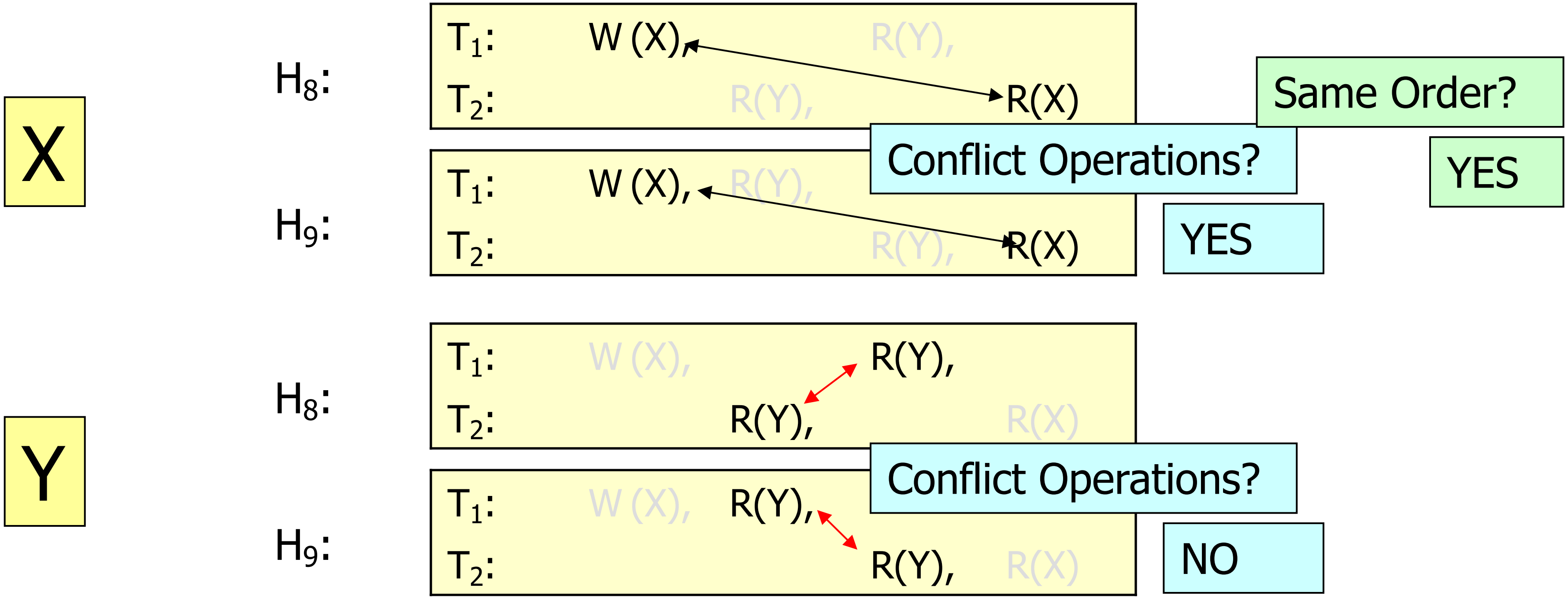
H_9:

T_1:	W (X),	R(Y),		
T_2:			R(Y),	R(X)

- □ Two schedules involve the same operations of the same transaction
- □ Every pair of conflicting operations is ordered in the same way in two schedules

5.1 Conflict Equivalent

- E.g.1. Is H_8 and H_9 conflict equivalent?



X

H_8:
T_1: W (X), R(Y),
T_2: R(Y), R(X)

H_9:
T_1: W (X), R(Y),
T_2: R(Y), R(X)

Conflict Operations?

YES

Same Order?

YES

Y

H_8:
T_1: W (X), R(Y),
T_2: R(Y), R(X)

H_9:
T_1: W (X), R(Y),
T_2: R(Y), R(X)

Conflict Operations?

NO

H_8 and H_9 are conflict equivalent

□ Two schedules involve the same operations of the same transaction

□ Every pair of conflicting operations is ordered in the same way in two schedules

5.1 Conflict Equivalent

- E.g.2. Is H_8 and H_{10} conflict equivalent?

H_8:

T_1:	W (X),		R(Y),	
T_2:		R(Y),		R(X)

H_{10}:

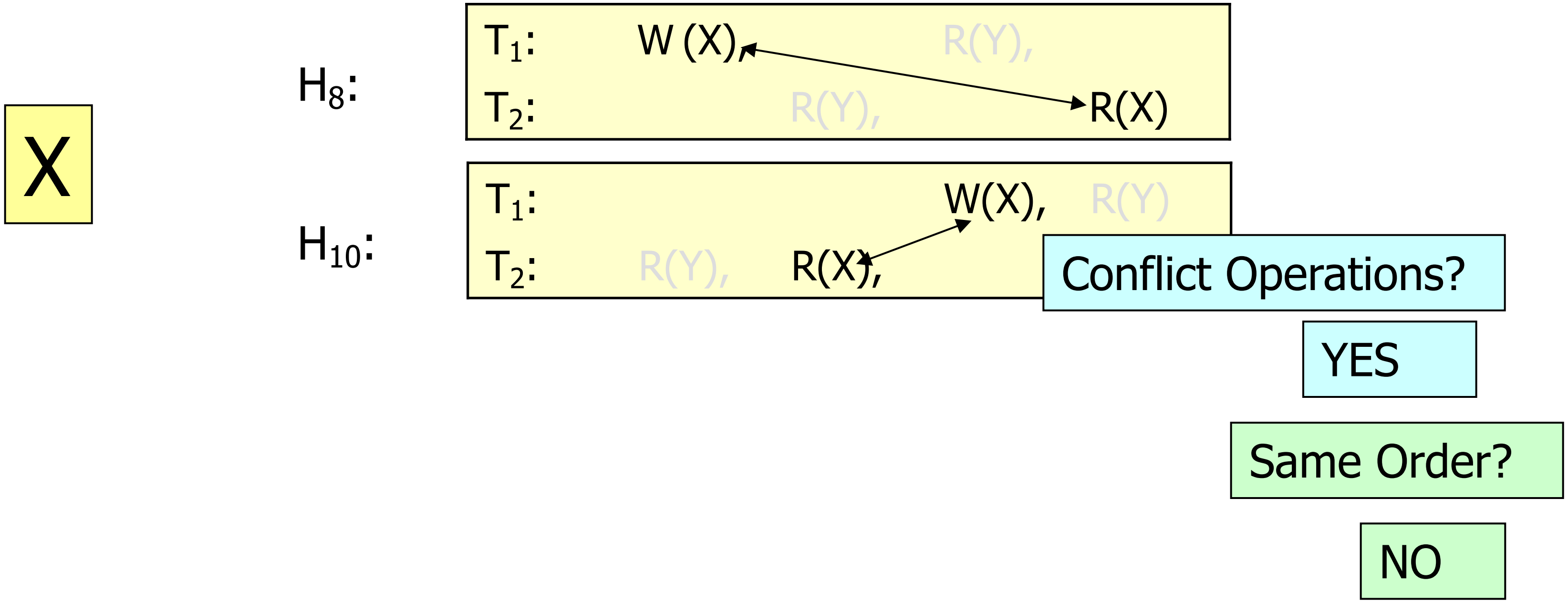
T_1:			W(X),	R(Y)
T_2:	R(Y),	R(X),		

□ Two schedules involve the same operations of the same transaction

□ Every pair of conflicting operations is ordered in the same way in two schedules

5.1 Conflict Equivalent

- E.g.2. Is H_8 and H_{10} conflict equivalent?



H_8:

T_1: W (X), R(Y),

T_2: R(Y), R(X)

H_{10}:

T_1: W(X), R(Y)

T_2: R(Y), R(X),

X

Conflict Operations?

YES

Same Order?

NO

H_8 and H_{10} are NOT conflict equivalent

5.2 Conflict Serializability

- S is **conflict serializable** if it is conflict equivalent to a serial schedule
- E.g.

H_8:

T_1:	W (X),		R(Y),	
T_2:		R(Y),		R(X)

H_9:

T_1:	W (X),	R(Y),		
T_2:			R(Y),	R(X)

- H_8 and H_9 are conflict equivalent
- H_9 is a serial schedule
- H_8 is conflict serializable

5.3 Precedence Graph

- Test for conflict serializability
- A directed graph G=(V,E), where
  - V includes all transactions involved in the schedule
  - E consists of all edges $T_i \rightarrow T_j$ for which one of three conditions holds:

Conflict Operations

    - T_i executes write(X) before T_j executes read(X)
    - T_i executes read(X) before T_j executes write(X)
    - T_i executes write(X) before T_j executes write(X)

T_i:	W(X),	
T_j:		R(X)

T_i:	R(X),	
T_j:		W(X)

T_i:	W (X),	
T_j:		W(X)

5.3 Precedence Graph

- The serialization order is obtained through **topological sorting**
- A schedule S is conflict serializable iff G(S) is **acyclic** **(i.e. no cycle)**

5.3 Precedence Graph

- E.g. Consider again the schedule H_8:

T_1:	W (X),		R(Y),	
T_2:		R(Y),		R(X)

X

T_1:	W (X),		R(Y),	
T_2:		R(Y),		R(X)

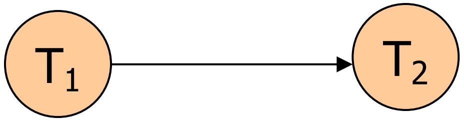
Conflict Operation

Y

T_1:	W (X),		R(Y),	
T_2:		R(Y),		R(X)

No Conflict Operation

The Precedence graph is:



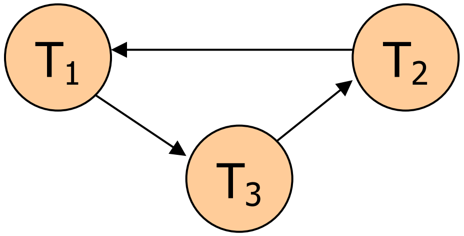
H_8 is conflict serializable and is conflict equivalent to T_1,T_2

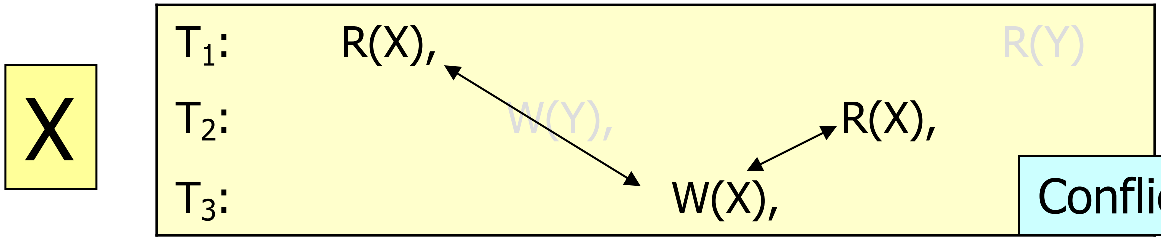
5.3 Precedence Graph

- E.g. Consider the schedule H_{11}:

T_1:	R(X),				R(Y)
T_2:		W(Y),		R(X),	
T_3:			W(X),		

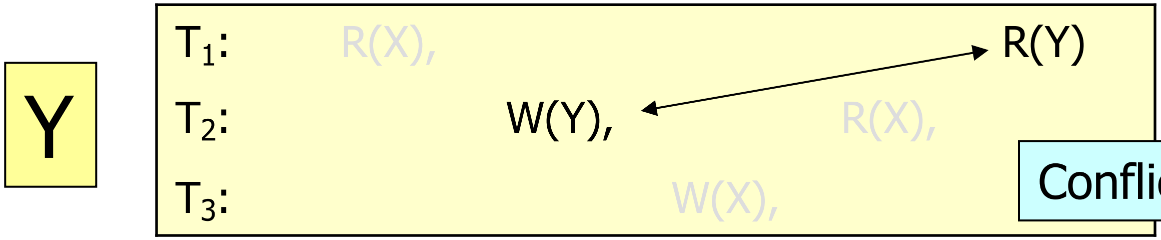
The Precedence graph is:





T_1:	R(X),				R(Y)
T_2:		W(Y),		R(X),	
T_3:			W(X),		

Conflict Operation? YES



T_1:	R(X),				R(Y)
T_2:		W(Y),		R(X),	
T_3:			W(X),		

Conflict Operation? YES

H_{11} is NOT conflict serializable as the Precedence graph contains a cycle

Outline

1. Transaction
2. ACID Property
3. Schedule
4. Serial Schedule
5. Conflict Serializability
6. View Serializability
7. Strict Two-Phase Locking
8. A question

6. View Serializability

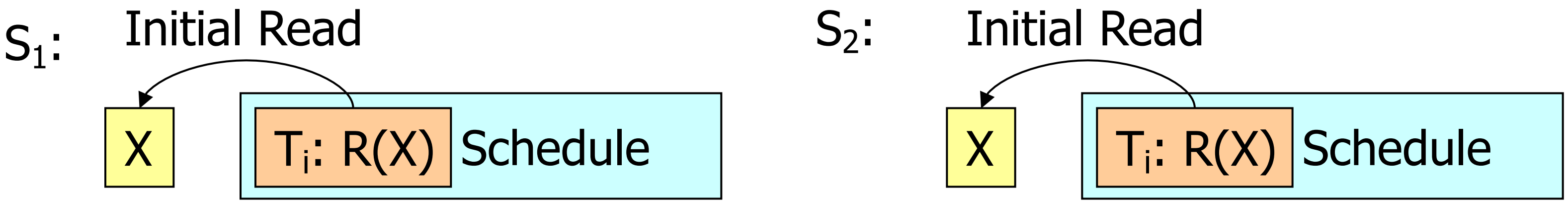
- Two schedules S_1 and S_2, where the same set of transactions participates in both schedules. They are said to be **view equivalent**

1. If T_i reads the *initial* value of a data item in S_1, T_i also reads the *initial* value of the item in S_2.
2. If T_i reads an item produced by T_j in S_1, T_i also reads the item produced by T_j in S_2.
3. If T_i writes the *final* value of a data item in S_1, T_i also writes the *final* value of the item in S_2.

6.1 View Equivalent

S_1 and S_2 are view equivalent

1



S_1: Initial Read

X

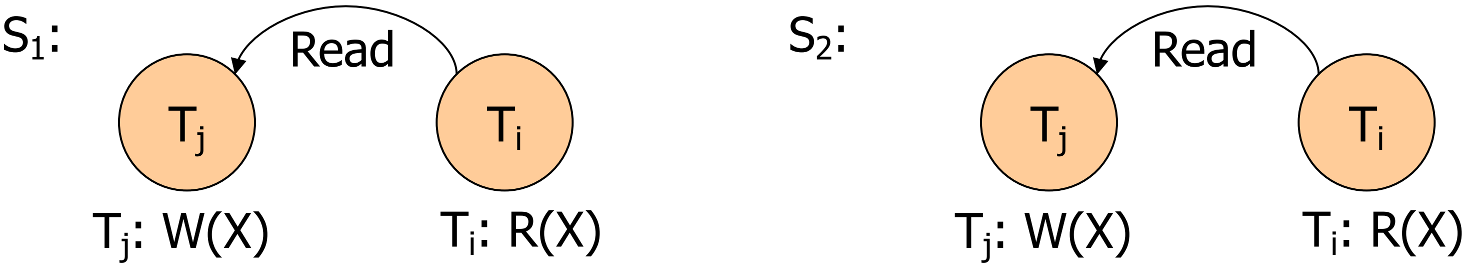
T_i: R(X) Schedule

S_2: Initial Read

X

T_i: R(X) Schedule

2



S_1:

Read

T_j

T_i

T_j: W(X)

T_i: R(X)

S_2:

Read

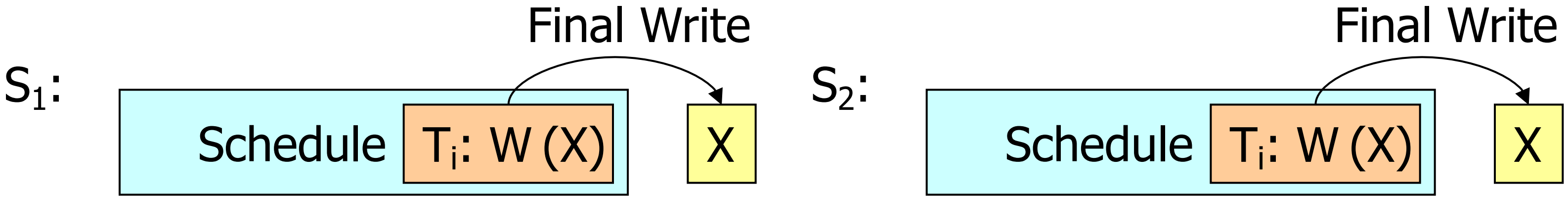
T_j

T_i

T_j: W(X)

T_i: R(X)

3



S_1: Final Write

Schedule T_i: W (X)

X

S_2: Final Write

Schedule T_i: W (X)

X

6.2 View Serializability

- S is **view serializable** if it is view equivalent to **a serial schedule**

- Suppose S_1 is view equivalent to S_2.
- S_2 is a serial schedule.
- In other words,
  S_1 is view equivalent to a serial schedule
  S_1 is said to be **view serializable**.

6.2 View Serializability

- E.g. Consider the schedule H_5:

H_5:

T_1:	R(X),				R(Y)
T_2:		R(Y),		R(X),	
T_3:			W(X),		

- Is it view serializable?

Consider a serial schedule H_6 :$T_1T_3T_2$

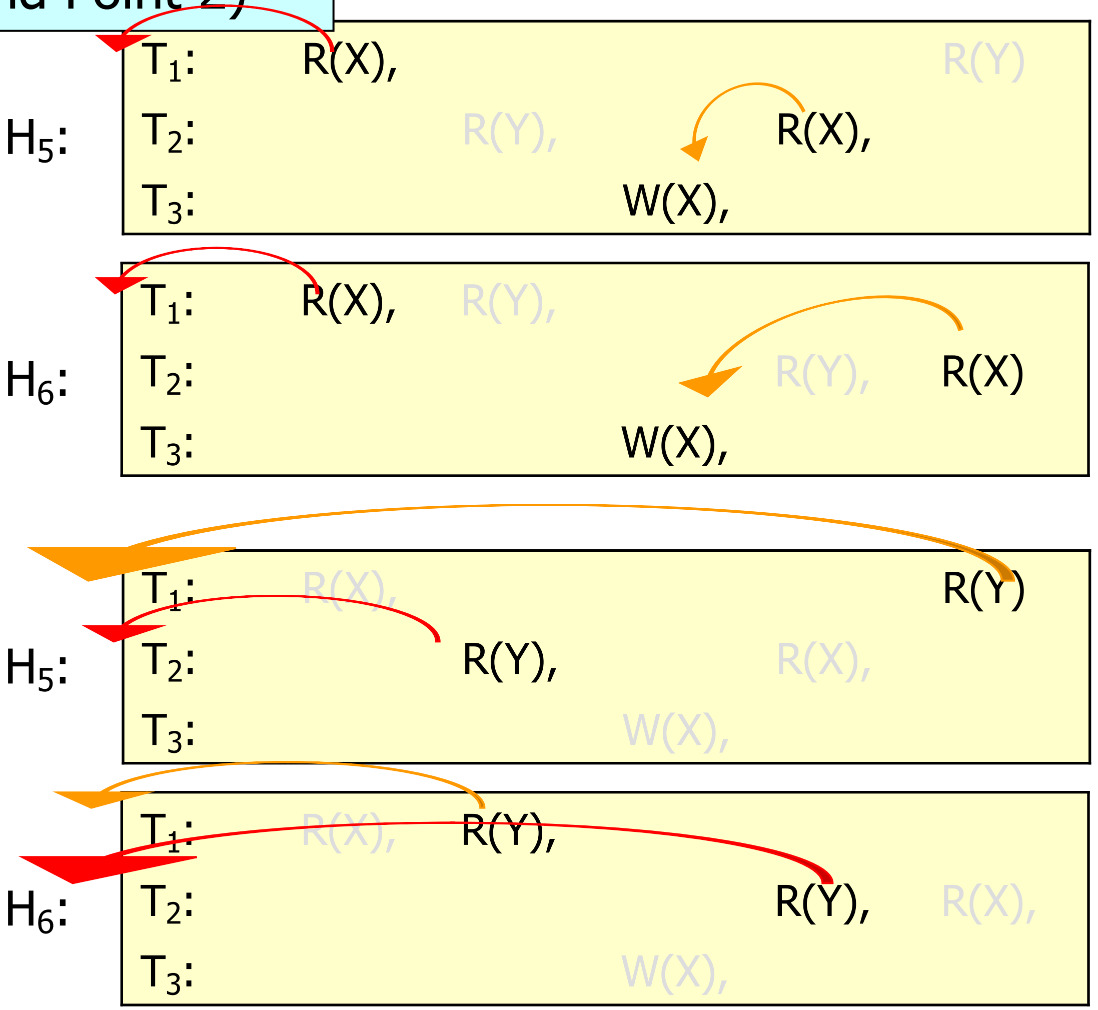
H_6:

T_1:	R(X),	R(Y),			
T_2:				R(Y),	R(X)
T_3:			W(X),		

6.2 View Serializability

Read (Point 1 and Point 2)

- E.g.



X

H5:

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

H6:

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

Y

H5:

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

H6:

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

6.2 View Serializability

Write (Point 3)

- E.g.

X

H5:	T_1:	R(X),				R(Y)
	T_2:		R(Y),		R(X),	
	T_3:			W(X),		

H6:	T_1:	R(X),	R(Y),			
	T_2:				R(Y),	R(X)
	T_3:			W(X),		

Y

H5:	T_1:	R(X),				R(Y)
	T_2:		R(Y),		R(X),	
	T_3:			W(X),		

H6:	T_1:	R(X),	R(Y),			
	T_2:				R(Y),	R(X)
	T_3:			W(X),		

6.2 View Serializability

- Answer
  - Yes, it is view serializable because it is view equivalent to

    $T_1T_3T_2 = R_1(X),R_1(Y),W_3(X),R_2(Y),R_2(X)$

6.2 View Serializability

- E.g.Consider the schedule H_7:

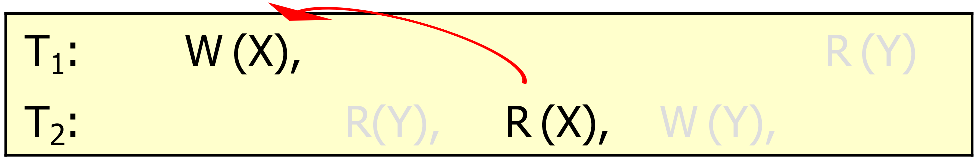
H_7:

T_1:	W (X),				R (Y)
T_2:		R(Y),	R (X),	W (Y),	

- Is it view serializable?

Read

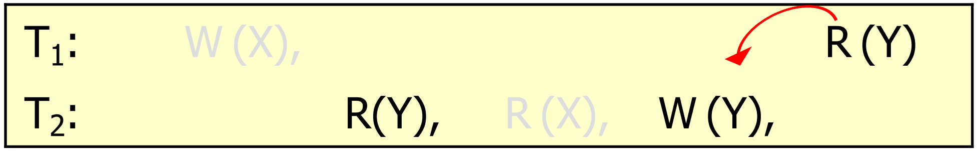
X



T_1:	W (X),				R (Y)
T_2:		R(Y),	R (X),	W (Y),	

T_2 reads from T_1

Y



T_1:	W (X),				R (Y)
T_2:		R(Y),	R (X),	W (Y),	

T_1 reads from T_2

H_7 is NOT view serializable because T_2 reads from T_1 and T_1 reads from T_2

Outline

1. Transaction
2. ACID Property
3. Schedule
4. Serial Schedule
5. Conflict Serializability
6. View Serializability
7. Strict Two-Phase Locking
8. A question

Lock-Based Protocols

When several transactions execute concurrently, the isolation property may no longer be preserved.
Isolation: user can understand a transaction without considering the effect of other transactions

A lock is a mechanism to control concurrent access to a data item.

Lock requests are made to concurrency-control manager. Transaction can proceed only after request is granted.

Lock-Based Protocols

- A lock is a mechanism to control concurrent access to a data item

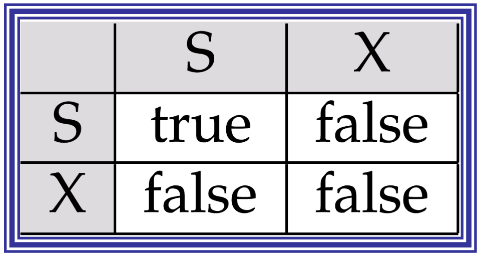
- Data items can be locked in two modes:

  1*. exclusive (X) mode*. Data item can be both read as well as written. X-lock is requested using **lock-X** instruction.

  2*. shared (S) mode*. Data item can only be read. S-lock is requested using **lock-S** instruction.

Lock-Based Protocols (Cont.)

Lock-compatibility matrix



	S	X
S	true	false
X	false	false

Any number of transactions can hold shared locks on an item, but if any transaction holds an exclusive on the item, no other transaction may hold any lock on the item.

If a lock cannot be granted, the requesting transaction is made to wait till all incompatible locks held by other transactions have been released. The lock is then granted.

Lock-Based Protocols (Cont.)

Example of a transaction performing locking:

```
$T_2$: lock-S(A);
    read (A);
    unlock(A);
    lock-S(B);
    read (B);
    unlock(B);
    display(A+B)
```

A **locking protocol** is a set of rules followed by all transactions while requesting and releasing locks. Locking protocols restrict the set of possible schedules.

Strict Two-Phase Locking Protocol

- Phase 1: Growing Phase
  - transaction may obtain locks
  - transaction may not release locks
- Phase 2: Shrinking Phase
  - transaction may release locks
  - transaction may not obtain locks
- The protocol assures serializability. It can be proved that the transactions can be serialized in the order of their **lock points** (i.e. the point where a transaction acquired its final lock).
- Disadv: Deadlock may occur

7. Strict Two-Phase Locking Protocol

- Two Kinds of Locks
  - Shared Lock - lock-S
    - Used when the transaction only read the data object
    - Allow read (for other transactions)
    - Not allow write (for other transactions)
  - Exclusive Lock - lock-X
    - Used when the transaction has a write operation on the data object
    - Not allow read (for other transactions)
    - Not allow write (for other transactions)
- E.g.1. T_1: R(A), R(B), W(B)

T_1: Lock-S(A) R(A), Lock-X(B) R(B), W(B) unlock(A) unlock(B)

- E.g.2. T2: R(B), R(A), W(A)

T_2: Lock-S(B) R(B), Lock-X(A) R(A), W(A) unlock(A) unlock(B)

7. Strict Two-Phase Locking Protocol

T_1: Lock-S(A) R(A), Lock-X(B) R(B), W(B) unlock(A) unlock(B)

T_2: Lock-S(B) R(B), Lock-X(A) R(A), W(A) unlock(A) unlock(B)

- The Schedule should be:

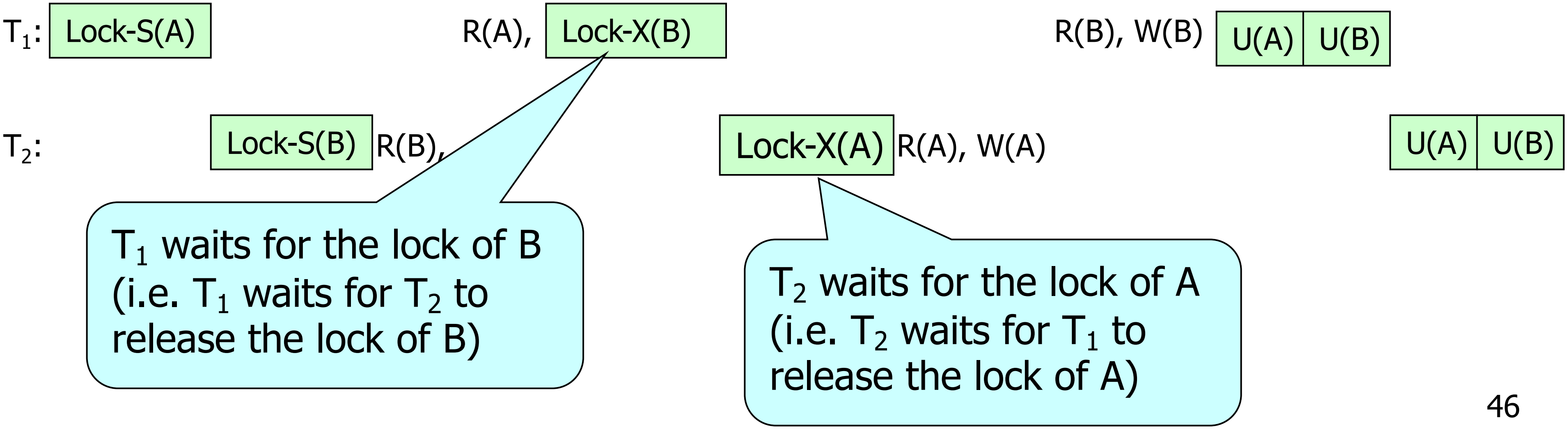
T_1:	Lock-S(A)	R(A),	Lock-X(B)	R(B), W(B)	U(A)	U(B)						
T_2:							Lock-S(B)	R(B),	Lock-X(A)	R(A), W(A)	U(A)	U(B)

7. Strict Two-Phase Locking Protocol

T_1: Lock-S(A) R(A), Lock-X(B) R(B), W(B) unlock(A) unlock(B)

T_2: Lock-S(B) R(B), Lock-X(A) R(A), W(A) unlock(A) unlock(B)

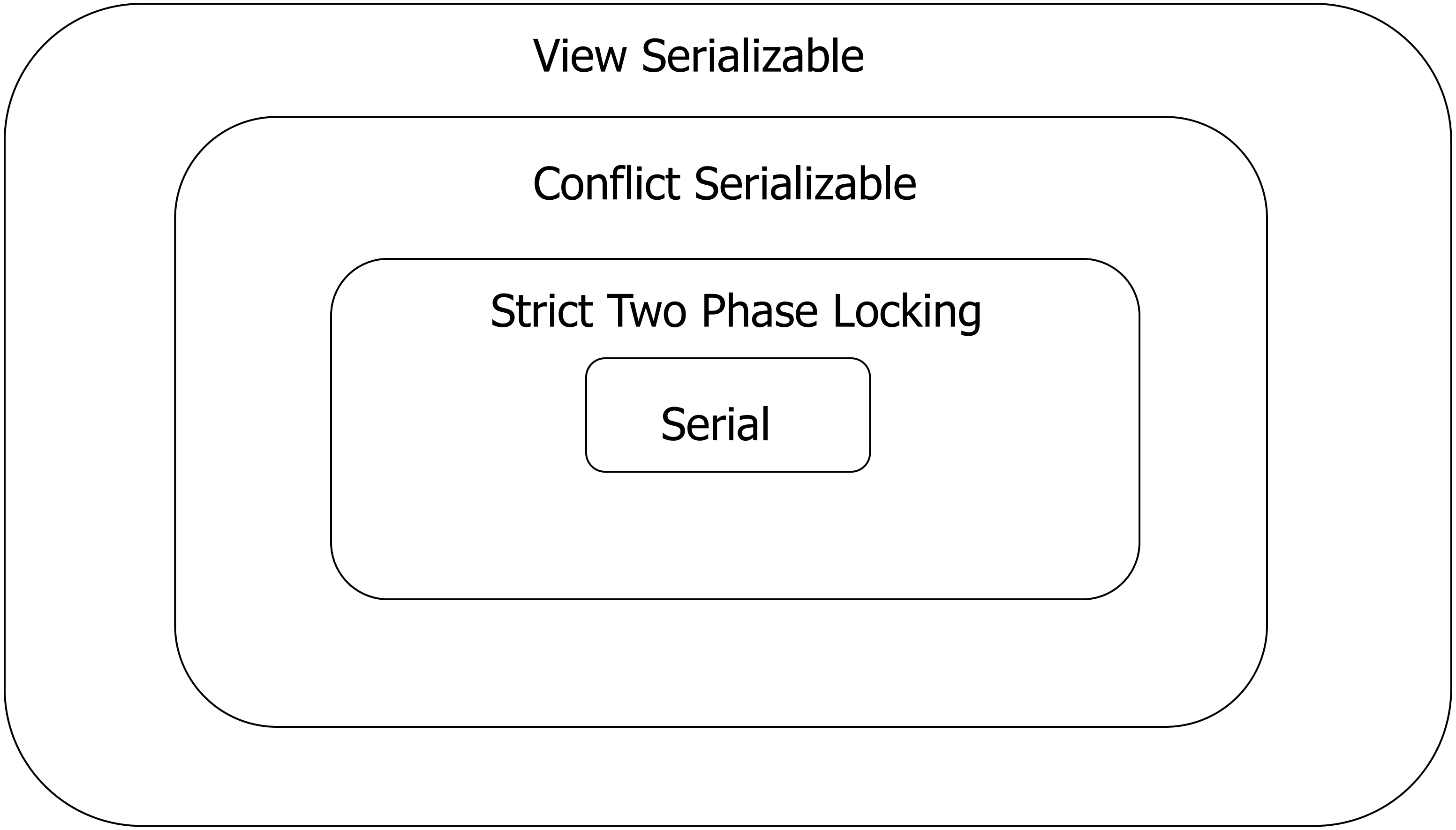
- But, there may be a deadlock!



T_1 waits for the lock of B (i.e. T_1 waits for T_2 to release the lock of B)

T_2 waits for the lock of A (i.e. T_2 waits for T_1 to release the lock of A)

7. Strict Two-Phase Locking Protocol



View Serializable

Conflict Serializable

Strict Two Phase Locking

Serial

Outline

1. Transaction
2. ACID Property
3. Schedule
4. Serial Schedule
5. Conflict Serializability
6. View Serializability
7. Strict Two-Phase Locking
8. A question

8 Questions

- Consider the following schedule:

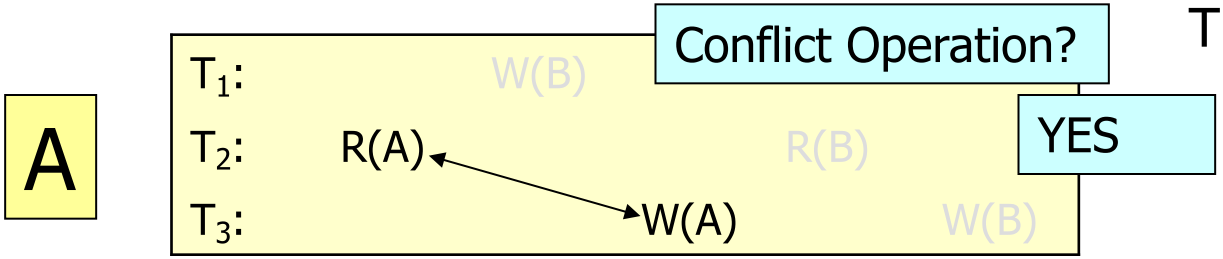
T_1:		W(B)			
T_2:	R(A)			R(B)	
T_3:			W(A)		W(B)

- (a) Draw the Precedence graph of the above schedule

8 Questions

T_1:		W(B)			
T_2:	R(A)			R(B)	
T_3:			W(A)		W(B)

- (a) Draw the Precedence graph of the above schedule



A

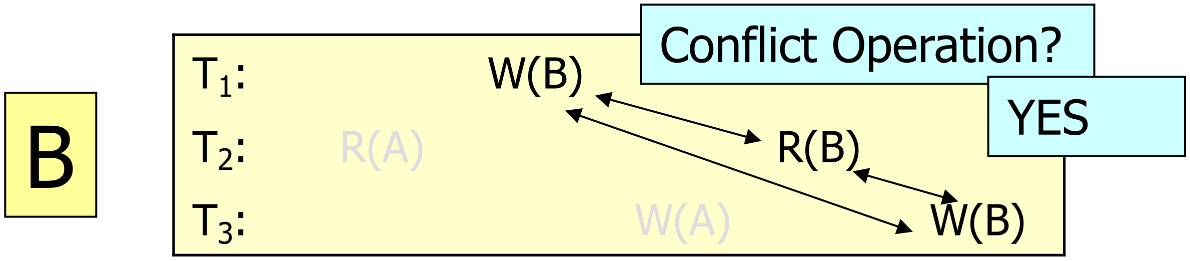
T_1: W(B)

T_2: R(A) R(B)

T_3: W(A) W(B)

Conflict Operation?

YES



B

T_1: W(B)

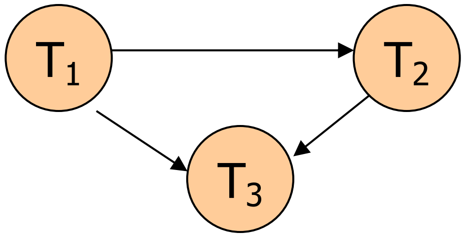
T_2: R(A) R(B)

T_3: W(A) W(B)

Conflict Operation?

YES

The Precedence graph is:

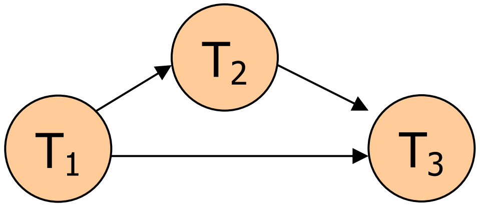


8 Questions

T_1:		W(B)			
T_2:	R(A)			R(B)	
T_3:			W(A)		W(B)

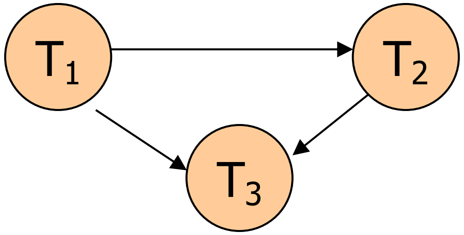
- (b) Is the schedule conflict serializable? Why? Please state the serialization order if it is conflict serializable.

Yes. This is because the Precedence graph does not contain any cycle.



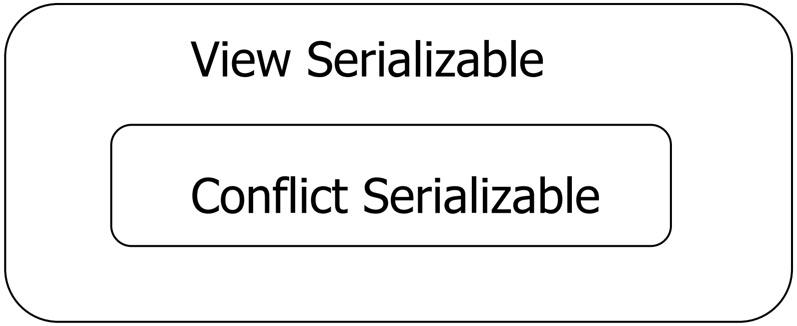
The serialization order is T_1, T_2 and T_3.

The Precedence graph is:



8 Questions

- (c) State the relation between the sets of conflict serializable schedules and view serializable schedules.



View Serializable

Conflict Serializable

The above set shows the relation between two sets of schedules.

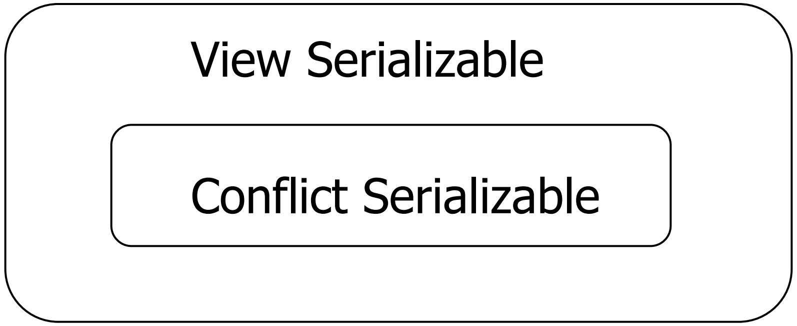
• If the schedule is conflict serializable, then it is also view serializable schedule.

8 Questions

- (d) From the answer in parts (b) and (c), can you conclude whether the schedule is view serializable? If your answer is “Yes”, please state whether the schedule is view serializable.

From (b), we know that the schedule is conflict serializable.

From (c),



View Serializable

Conflict Serializable

YES. We can conclude that the schedule is view serializable