Distributed Systems

CS425/ECE428

Instructor: Radhika Mittal

Acknowledgements for the materials: Indy Gupta and Nikita Borisov

Logistics

- Please reserve a slot for Midterm 2.
  - April 12-14.
- MP2 is due next Friday.
- I won't be able to hold my OH today.
  - Can entertain a few quick questions after class.

Thanks for the feedback!

- What people liked in general:
  - course content, lectures, MPs, TAs.

- What can be improved:
  - More real-world applications (coming up!)
  - Pace of lectures: too fast for some, too slow for some, appropriate for some…
  - Repetition in lectures
  - More practice materials / exam differed from homework….
    - Examples in class….
  - README for MP2

Agenda for today

- Transaction Processing and Concurrency Control
  - Chapter 16
    - Transaction semantics: ACID
    - Isolation and serial equivalence
    - Conflicting operations
    - Two-phase locking
    - Deadlocks
    - **Timestamped ordering**
- Distributed Transactions (if time)

Concurrency Control: Two approaches

- Pessimistic: assume the worst, prevent transactions from accessing the same object
  - E.g., Locking
- Optimistic: assume the best, allow transactions to write, but check later
  - E.g., Check at commit time

Concurrency Control: Two approaches

- Pessimistic: assume the worst, prevent transactions from accessing the same object
  - E.g., Locking

- Optimistic: assume the best, allow transactions to write, but check later
  - E.g., Check at commit time

Timestamped ordering: per-object state

- Committed value.
- Transaction id (timestamp) that wrote the committed value.
- Read timestamps (RTS): List of transaction ids (timestamps) that have read the committed value.
- Tentative writes (TW): List of tentative writes sorted by the corresponding transaction ids (timestamps).
    - Timestamped *versions* of the object.

Timestamped ordering: write rule

Transaction T_c requests a write operation on object D

if (Tc ≥ max. read timestamp on D

&& Tc > write timestamp on committed version of D)

Perform a tentative write on D:

If T_c already has an entry in the TW list for D, update it.

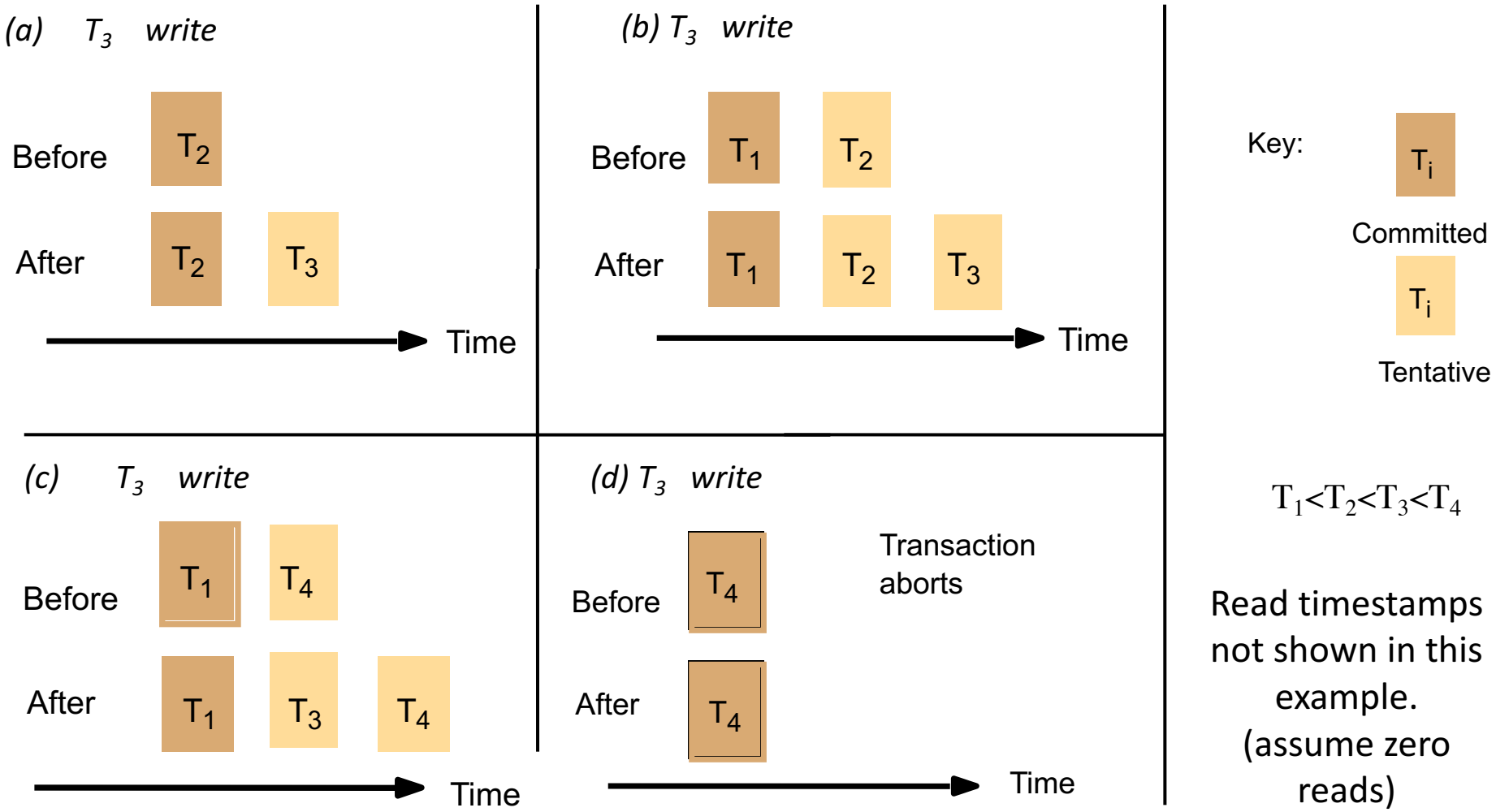
Else, add T_c and its write value to the TW list.

else

abort transaction T_c

//too late; a transaction with later timestamp has already read or written the object.

Timestamped ordering: write rule



Timestamped ordering: read rule

Transaction T_c requests a read operation on object D

if (T_c > write timestamp on committed version of D) {

D_s = version of D with the maximum write timestamp that is ≤ T_c

//search across the committed timestamp and the TW list for object D.

if (D_s is committed)

read D_s and add T_c to RTS list (if not already added)

else

if D_s was written by T_c, simply read D_s

else

wait until the transaction that wrote D_s is committed or aborted, and reapply the read rule.

// if the transaction is committed, T_c will read its value after the wait.

// if the transaction is aborted, T_c will read the value from an older transaction.

} else

abort transaction T_c

//too late; a transaction with later timestamp has already written the object.

Timestamped ordering: read rule

Transaction T_c requests a read operation on object D

if (T_c > write timestamp on committed version of D) {

D_s = version of D with the maximum write timestamp that is ≤ T_c

//search across the committed timestamp and the TW list for object D.

if (D_s is committed)

read D_s and add T_c to RTS list (if not already added)

else

if D_s was written by T_c, simply read D_s

else

wait until the transaction that wrote D_s is committed or aborted, and reapply the read rule. ***(variants exist in literature and practice; we will stick to this rule as per your textbook)***

// if the transaction is committed, T_c will read its value after the wait.

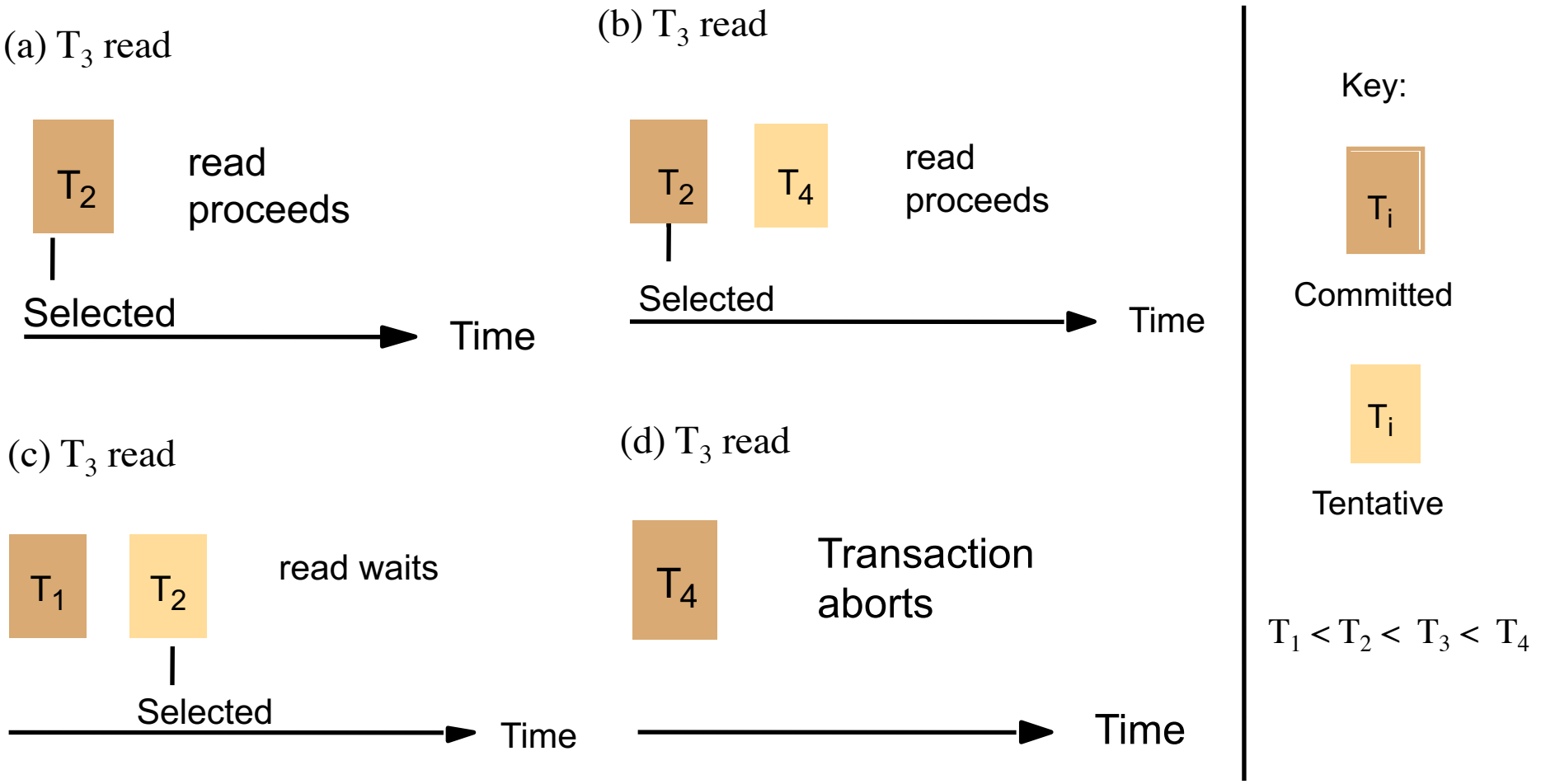
// if the transaction is aborted, T_c will read the value from an older transaction.

} else

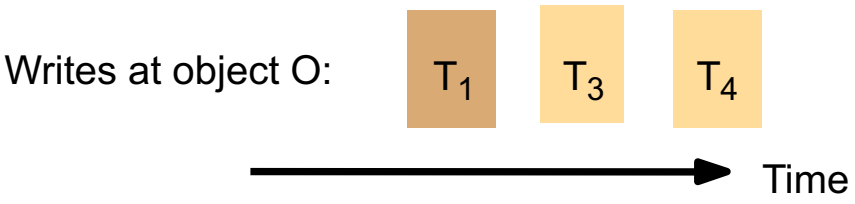
abort transaction T_c

//too late; a transaction with later timestamp has already written the object.

Timestamped ordering: read rule

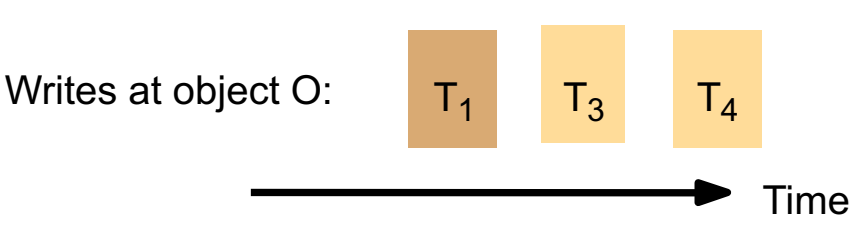


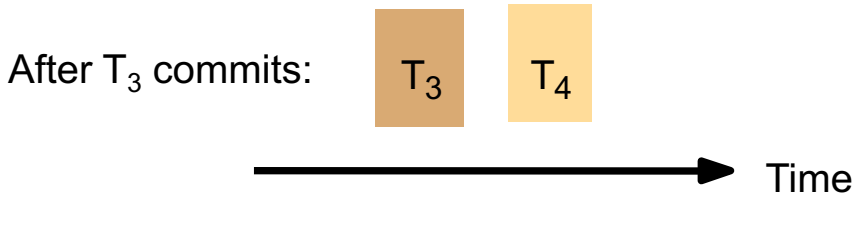
Timestamped ordering: committing



- Suppose T_4 is ready to commit.
- Must wait until T_3 commits or aborts.

- When a transaction is committed, the committed value of the object and associated timestamp are updated, and the corresponding write is removed from TW list.





Lost Update Example with Timestamped Ordering

Transaction T1	**Transaction T2**
x = getSeats(ABC123);	
	x = getSeats(ABC123);
if(x > 1)	if(x > 1)
x = x – 1;	
write(x, ABC123);	
	x = x – 1;
	write(x, ABC123);
commit	
	commit

ABC123: state
committed value = 10
committed timestamp = 0
RTS:
TW:

Lost Update Example with Timestamped Ordering

Transaction T1	**Transaction T2**
x = getSeats(ABC123);	
	x = getSeats(ABC123);
if(x > 1)	if(x > 1)
x = x – 1;	
write(x, ABC123);	
	x = x – 1;
	write(x, ABC123);
commit	
	commit

ABC123: state
committed value = 10
committed timestamp = 0
RTS: 1
TW:

Lost Update Example with Timestamped Ordering

Transaction T1	**Transaction T2**
x = getSeats(ABC123);	
	x = getSeats(ABC123);
if(x > 1)	if(x > 1)
x = x – 1;	
write(x, ABC123);	
	x = x – 1;
	write(x, ABC123);
commit	
	commit

ABC123: state
committed value = 10
committed timestamp = 0
RTS: 1
TW:

Lost Update Example with Timestamped Ordering

Transaction T1	**Transaction T2**
x = getSeats(ABC123);	
	x = getSeats(ABC123);
if(x > 1)	if(x > 1)
x = x – 1;	
write(x, ABC123);	
	x = x – 1;
	write(x, ABC123);
commit	
	commit

ABC123: state
committed value = 10
committed timestamp = 0
RTS: 1, 2
TW:

Lost Update Example with Timestamped Ordering

Transaction T1

x = getSeats(ABC123);

if(x > 1)

x = x – 1;

write(x, ABC123);

commit

Transaction T2

x = getSeats(ABC123);

if(x > 1)

x = x – 1;

write(x, ABC123);

commit

ABC123: state
committed value = 10
committed timestamp = 0
RTS: 1, 2
TW:

Abort!

Next Example with Timestamped Ordering

Transaction T1

x = getSeats(ABC123);
y = getSeats(ABC789);
write(x-5, ABC123);

write(y+5, ABC789);

commit

Transaction T2

x = getSeats(ABC123);
y = getSeats(ABC789);

print("Total:" x+y);

commit

ABC123: state
committed value = 10
committed timestamp = 0
RTS:
TW:

ABC789: state
committed value = 5
committed timestamp = 0
RTS:
TW:

Next Example with Timestamped Ordering

Transaction T1

x = getSeats(ABC123);
y = getSeats(ABC789);
write(x-5, ABC123);

write(y+5, ABC789);

commit

Transaction T2

x = getSeats(ABC123);
y = getSeats(ABC789);

print("Total:" x+y);

commit

ABC123: state
committed value = 10
committed timestamp = 0
RTS:
TW:

ABC789: state
committed value = 5
committed timestamp = 0
RTS:
TW:

Next Example with Timestamped Ordering

Transaction T1

x = getSeats(ABC123);
y = getSeats(ABC789);
write(x-5, ABC123);

write(y+5, ABC789);

commit

Transaction T2

x = getSeats(ABC123);
y = getSeats(ABC789);

print("Total:" x+y);

commit

ABC123: state
committed value = 10
committed timestamp = 0
RTS: 1
TW:

ABC789: state
committed value = 5
committed timestamp = 0
RTS:
TW:

Next Example with Timestamped Ordering

Transaction T1

x = getSeats(ABC123);
y = getSeats(ABC789);
write(x-5, ABC123);

write(y+5, ABC789);

commit

Transaction T2

x = getSeats(ABC123);
y = getSeats(ABC789);

print("Total:" x+y);

commit

ABC123: state
committed value = 10
committed timestamp = 0
RTS: 1
TW:

ABC789: state
committed value = 5
committed timestamp = 0
RTS:
TW:

Next Example with Timestamped Ordering

Transaction T1

x = getSeats(ABC123);
y = getSeats(ABC789);
write(x-5, ABC123);

write(y+5, ABC789);

commit

Transaction T2

x = getSeats(ABC123);
y = getSeats(ABC789);

print("Total:" x+y);

commit

ABC123: state
committed value = 10
committed timestamp = 0
RTS: 1
TW:

ABC789: state
committed value = 5
committed timestamp = 0
RTS: 1
TW:

Next Example with Timestamped Ordering

Transaction T1

x = getSeats(ABC123);
y = getSeats(ABC789);
write(x-5, ABC123);

write(y+5, ABC789);

commit

Transaction T2

x = getSeats(ABC123);
y = getSeats(ABC789);

print("Total:" x+y);

commit

ABC123: state
committed value = 10
committed timestamp = 0
RTS: 1
TW:

ABC789: state
committed value = 5
committed timestamp = 0
RTS: 1
TW:

Next Example with Timestamped Ordering

Transaction T1

x = getSeats(ABC123);
y = getSeats(ABC789);
write(x-5, ABC123);

write(y+5, ABC789);

commit

Transaction T2

x = getSeats(ABC123);
y = getSeats(ABC789);

print("Total:" x+y);

commit

ABC123: state
committed value = 10
committed timestamp = 0
RTS: 1
TW: (5, 1)

ABC789: state
committed value = 5
committed timestamp = 0
RTS: 1
TW:

Next Example with Timestamped Ordering

Transaction T1

x = getSeats(ABC123);
y = getSeats(ABC789);
write(x-5, ABC123);

write(y+5, ABC789);

commit

Transaction T2

x = getSeats(ABC123);
y = getSeats(ABC789);

print("Total:" x+y);

commit

ABC123: state
committed value = 10
committed timestamp = 0
RTS: 1
TW: (5, 1)

ABC789: state
committed value = 5
committed timestamp = 0
RTS: 1
TW:

Next Example with Timestamped Ordering

Transaction T1

x = getSeats(ABC123);
y = getSeats(ABC789);
write(x-5, ABC123);

write(y+5, ABC789);

commit

Transaction T2

x = getSeats(ABC123); wait
y = getSeats(ABC789);

print("Total:" x+y);

commit

ABC123: state
committed value = 10
committed timestamp = 0
RTS: 1
TW: (5, 1)

ABC789: state
committed value = 5
committed timestamp = 0
RTS: 1
TW:

Next Example with Timestamped Ordering

Transaction T1

x = getSeats(ABC123);
y = getSeats(ABC789);
write(x-5, ABC123);

write(y+5, ABC789);

commit

Transaction T2

x = getSeats(ABC123); wait
y = getSeats(ABC789);

print("Total:" x+y);

commit

ABC123: state
committed value = 10
committed timestamp = 0
RTS: 1
TW: (5, 1)

ABC789: state
committed value = 5
committed timestamp = 0
RTS: 1
TW:

Next Example with Timestamped Ordering

Transaction T1

x = getSeats(ABC123);
y = getSeats(ABC789);
write(x-5, ABC123);

write(y+5, ABC789);

commit

Transaction T2

x = getSeats(ABC123); wait
y = getSeats(ABC789);

print("Total:" x+y);

commit

ABC123: state
committed value = 10
committed timestamp = 0
RTS: 1
TW: (5, 1)

ABC789: state
committed value = 5
committed timestamp = 0
RTS: 1
TW: (10, 1)

Next Example with Timestamped Ordering

Transaction T1

x = getSeats(ABC123);
y = getSeats(ABC789);
write(x-5, ABC123);

write(y+5, ABC789);

commit

Transaction T2

x = getSeats(ABC123); wait
y = getSeats(ABC789);

print("Total:" x+y);

commit

ABC123: state
committed value = 10
committed timestamp = 0
RTS: 1
TW: (5, 1)

ABC789: state
committed value = 5
committed timestamp = 0
RTS: 1
TW: (10, 1)

Next Example with Timestamped Ordering

Transaction T1

x = getSeats(ABC123);
y = getSeats(ABC789);
write(x-5, ABC123);

write(y+5, ABC789);

commit

Transaction T2

x = getSeats(ABC123); wait
y = getSeats(ABC789);

print("Total:" x+y);

commit

ABC123: state
committed value = ~~10~~ 5
committed timestamp = ~~0~~ 1
RTS: 1
TW: ~~(5, 1)~~

ABC789: state
committed value = ~~5~~ 10
committed timestamp = ~~0~~ 1
RTS: 1
TW: ~~(10, 1)~~

Next Example with Timestamped Ordering

Transaction T1

x = getSeats(ABC123);
y = getSeats(ABC789);
write(x-5, ABC123);

write(y+5, ABC789);

commit

Transaction T2

x = getSeats(ABC123); wait
y = getSeats(ABC789);

print("Total:" x+y);

commit

ABC123: state
committed value = ~~10~~ 5
committed timestamp = ~~0~~ 1
RTS: 1
TW: ~~(5, 1)~~

ABC789: state
committed value = ~~5~~ 10
committed timestamp = ~~0~~ 1
RTS: 1
TW: ~~(10, 1)~~

T2 then proceeds after T1 commits

Concurrency Control: Summary

- *How to prevent transactions from affecting one another?*
- Goal: increase concurrency and transaction throughput while maintaining correctness (ACID).
- Target *serial equivalence.*
- Two approaches:
  - Pessimistic concurrency control: locking based.
    - read-write locks with two-phase locking and deadlock detection.
  - Optimistic concurrency control: abort if too late.
    - timestamped ordering.