CSC 742
Database Management Systems

Topic 14: Concurrency Control
– Time Ordering

Timestamp Ordering

- Order transactions based on their timestamps
  - this order determines the equivalent serial schedule
- Options for timestamps
  - Transaction start time
  - Sequence number assigned to new transactions increasingly.
  - Essential requirement: Have the same order as the transaction start times.

Timestamp Ordering

- Define timestamps of data items
  - Read-TS(X): the largest timestamp among all the timestamps of transactions that have successfully read X
  - Write-TS(X): the largest timestamp among all the timestamps of transactions that have successfully write X

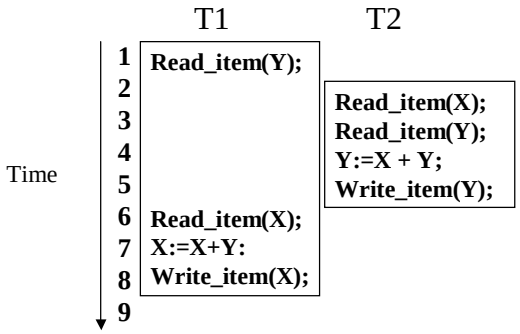
Basic Timestamp Ordering

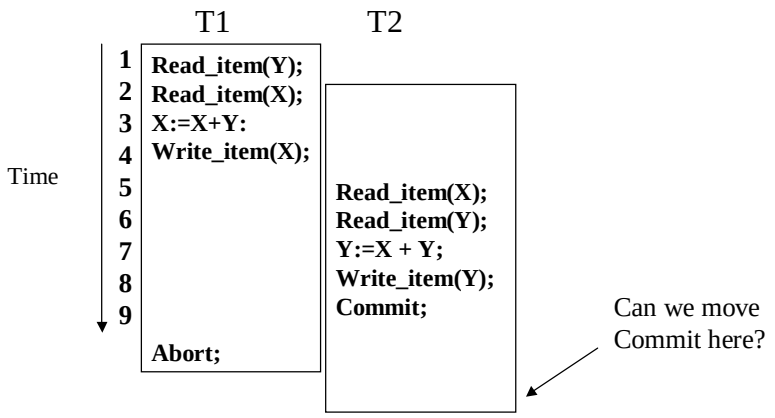
- When T tries to write(X)
  - if Read_TS(X) > TS(T) or Write_TS(S) > TS(T)
    - Intuition: X has been read or written by a "later" transaction
    - Abort T
  - else
    - Execute and set write-TS(X) = TS(T)

Basic Timestamp Ordering (Cont'd)

- When T tries to read(X)
  - if Write_TS(X) > TS(S)
    - X was written by a "later" transaction
    - Abort T
  - else
    - Execute and update read-TS(X)
- Intuition:
  - Order conflicting operations in the same order as the transaction timestamps.
  - No deadlock!

Read_TS(X)=0
Write_TS(X)=0
Read_TS(Y)=0
Write_TS(Y)=0

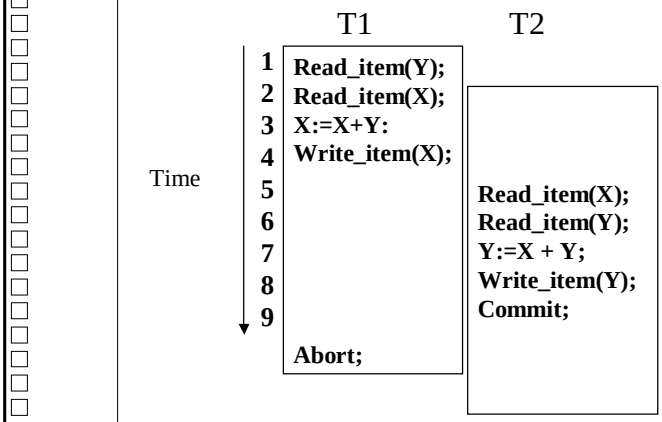




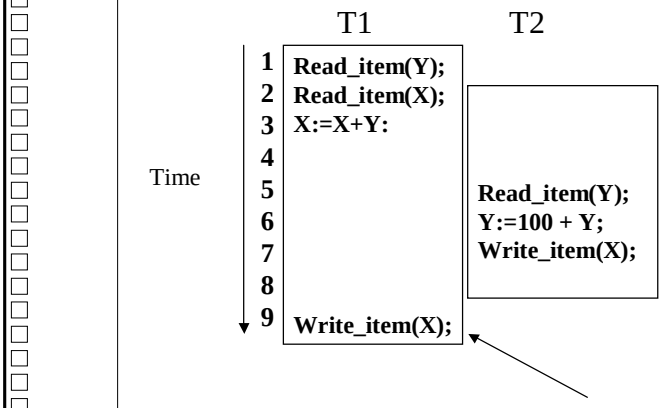
Is this allowed by Basic TO?
Is this recoverable?

Strict Time Ordering

- In addition to the Basic TO rules
  - Delay read(X) or write(X) until the transaction T' that wrote(X) has committed or aborted.
    - Write_TS(X)=TS(T')
  - Equivalent to using locks along with Basic TO.



How will these be changed with Strict TO?



Is this allowed by Basic (or Strict) TO?
What will happen with Basic (or Strict) TO?
Other options?

Thomas's Write Rule

- When T tries to write(X)
  - If read_TS(X)>TS(T)
    - X has been read by a later transaction
    - Abort T
  - Else if write_TS(X)>TS(T)
    - X has been written by a later transaction
    - Ignore write(X)
  - Else
    - Execute write(X) and update write-TS(X)
- Does not guarantee conflict serializability

Multiversion Timestamp Ordering: 1

- Several versions X1, …, Xk of data item X
- Several transactions may read a version
- Only one transaction can write a version
- Save read and write timestamps for each version
  - Read_TS(Xi): The largest of the stamps of the transactions that have read Xi
  - Write_TS(X): The largest of the stamps of the transactions that have read Xi

Multiversion TO: 2

- When T tries to write X
  - if the last write version preceding T has been read by a later transaction, abort T
  - else create a new version with read_TS(X)=write_TS(X)=TS(T)
- When T tries to read X
  - find last write version preceding T
  - update its read-TS

Rollbacks may cascade

Which version of X to use?

- TS(T)=10
- T wants to read(X)
  - Read_TS(X1)=6, Write_TS(X1)=5
  - Read_TS(X2)=9, Write_TS(X2)=7
  - Read_TS(X3)=14, Write_TS(X3)=11
- T wants to write(X)
  - Read_TS(X1)=6, Write_TS(X1)=5
  - Read_TS(X2)=12, Write_TS(X2)=7
  - Read_TS(X3)=14, Write_TS(X3)=11