# COMP207
# Database Development

Lecture 11

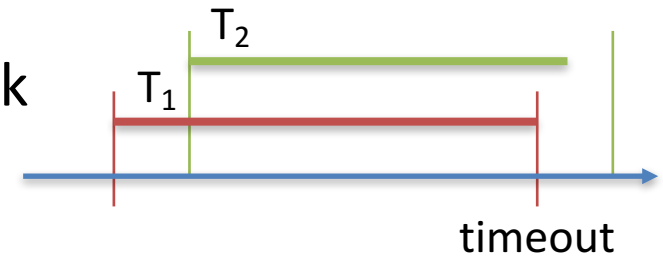
Transaction Management:
Timestamp-based Scheduling &
Concluding Remarks

# Deadlock prevention

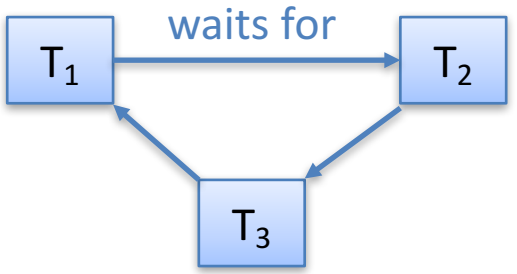
- Two approaches for deadlock prevention:
  - **Detect deadlocks & fix them**
  - **Enforce deadlock-free schedules**

# Deadlock Detection: Approaches

- **Timeouts**
  - Assume a transaction is in a deadlock if it exceeds a given time limit




- **Wait-for graphs**
  - Nodes: transactions
  - Edge from $T_1$ to $T_2$ if $T_1$ waits for $T_2$ to release a lock
  - Deadlocks correspond to cycles




- **Timestamp-based**

# Timestamps for Deadlock Detection

- Each transaction T is assigned a unique integer **TS(T)** upon **arrival** (the **timestamp of T**), such that later arrival means higher number
- **Timestamps for deadlock detection do not change** even after a restart!
- Two schemes, each based on some times doing rollback when a transaction $T_1$ requests a lock hold by another transaction $T_2$:
  - **Wait-Die** – $\mathbf{T_1}$ is rolled back, if it is younger than $\mathbf{T_2}$
  - **Wound-Wait** – $\mathbf{T_2}$ is rolled back, if it is younger than $\mathbf{T_1}$

# Why Wound-Wait Works

Eventually, any finite number of transactions finishes under Wound-Wait

- At all times, the oldest transaction can move
- Hence, eventually it finishes and there is one less transaction left and we are still doing Wound-Wait!

- Wait-Die is similar, but we look at the oldest transaction or the transaction it (recursively) waits for

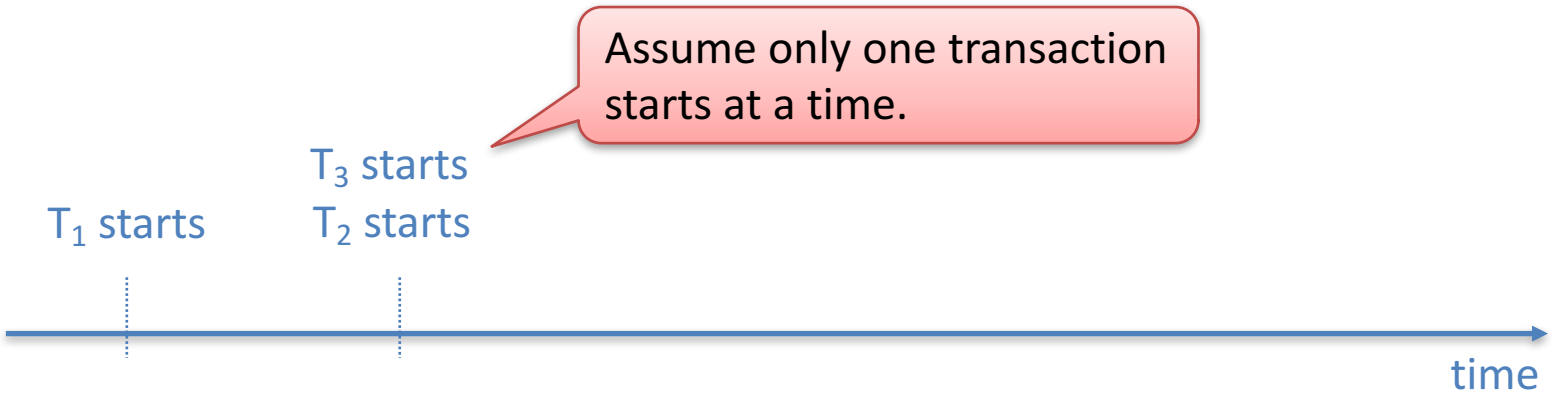
# Deadlock prevention

- Two approaches for deadlock prevention:
  - **Detect deadlocks & fix them**
  - **Enforce deadlock-free schedules**

Uses timestamps too!

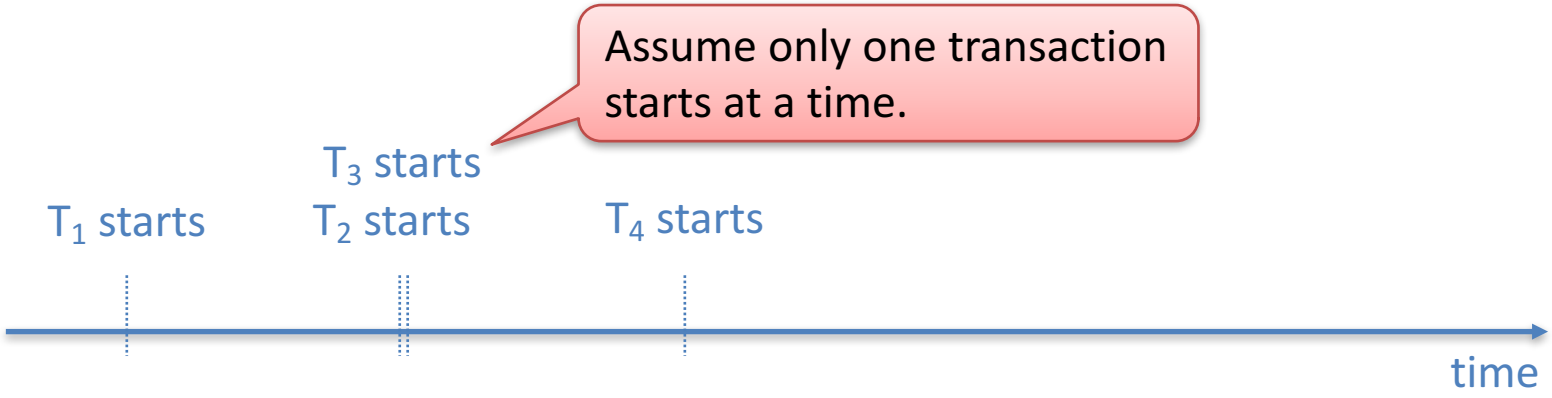
# Basic Idea

- Schedule transactions so that the effect is the same as **executing each transaction instantaneously** when it is started.



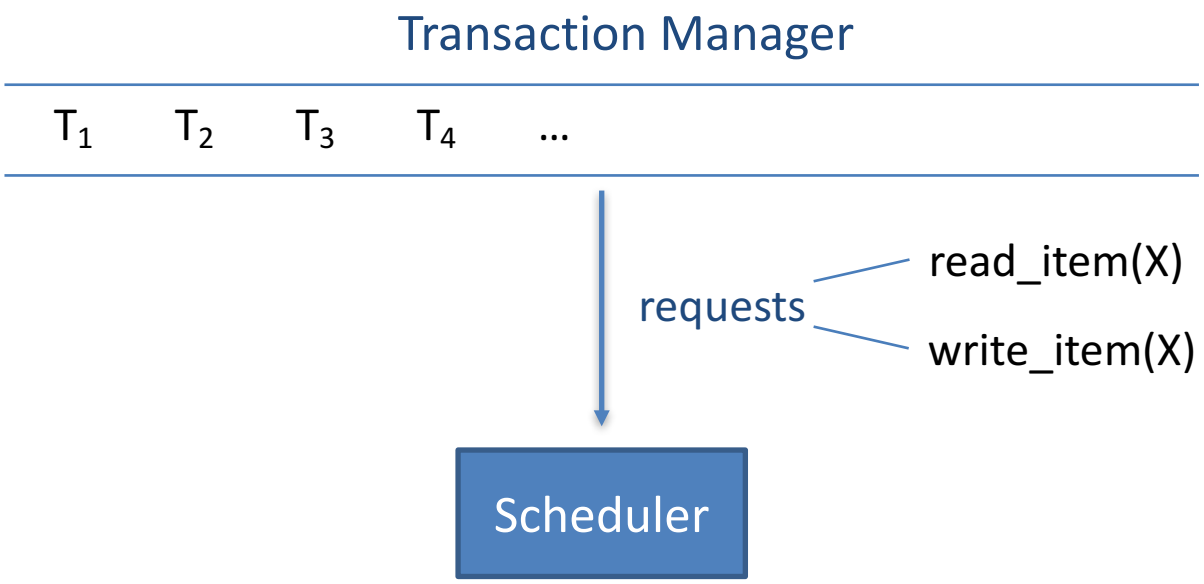
# Basic Idea

- Schedule transactions so that the effect is the same as **executing each transaction instantaneously** when it is started.




- Equivalent to serial schedule that has all transactions in the order of their start time.

# Timestamp-Based Schedulers




**Possible actions of the scheduler:**

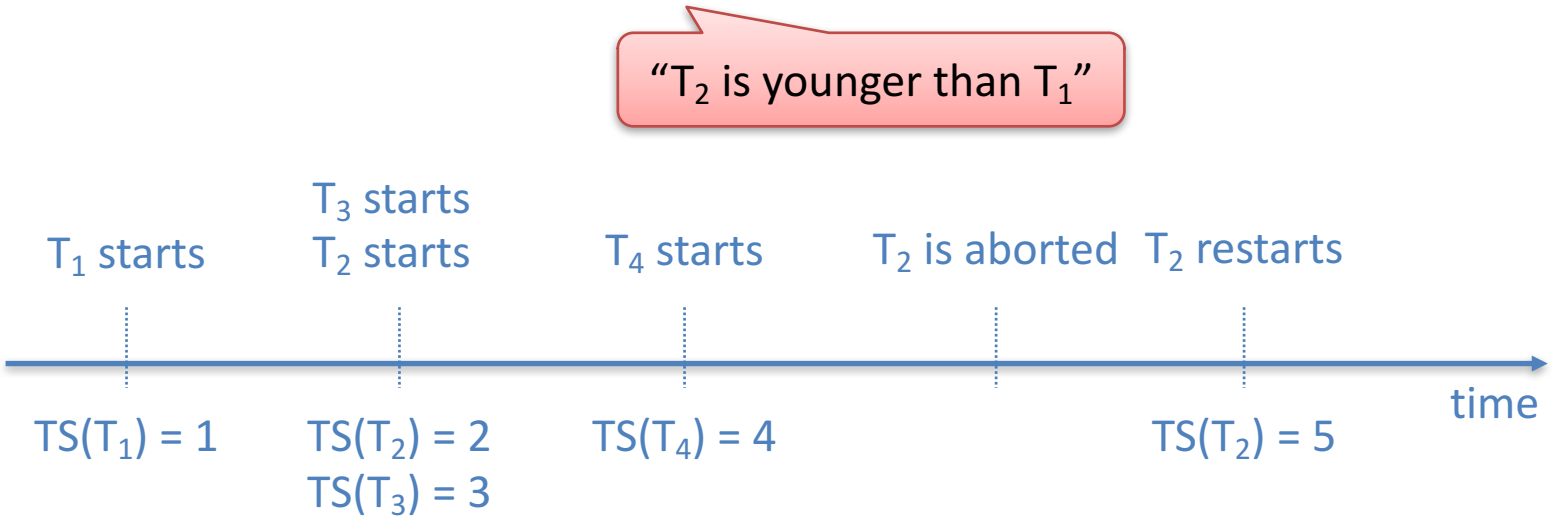
Grant request

Abort transaction

Delay transaction

# Timestamps

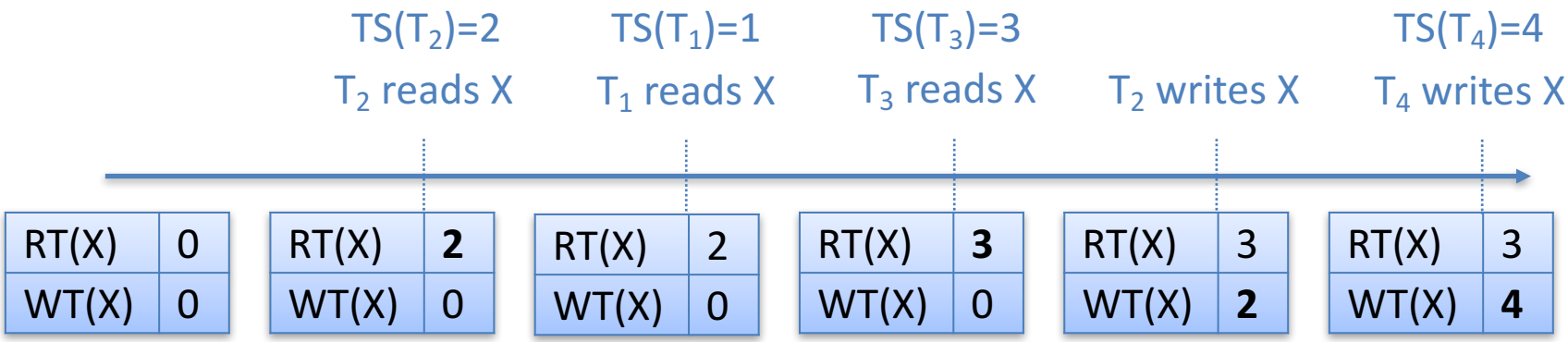
- Each transaction **T** is assigned a unique integer **TS(T)** when it starts (the **timestamp of T**).
- If $T_1$ started earlier than $T_2$, we require **$TS(T_1) < TS(T_2)$**




- We assign a **new timestamp even after a restart**!

# Additional Bookkeeping

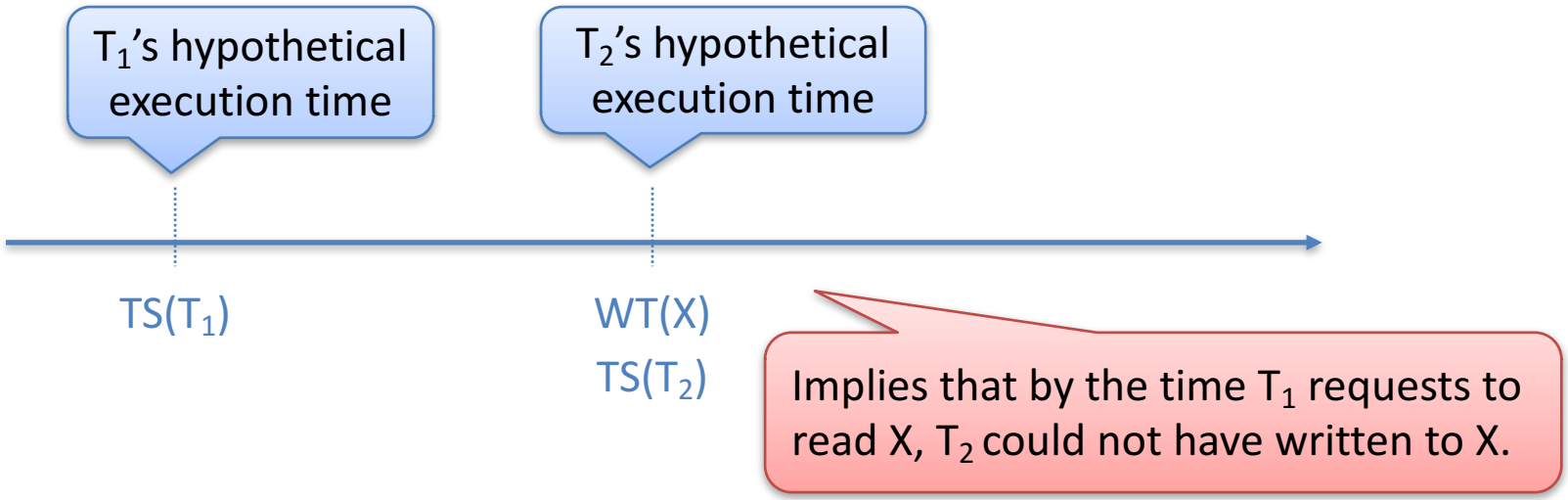
- For each database item X, maintain:
  - **Read Time of X: RT(X)**
    Timestamp of youngest transaction that read X
  - **Write Time of X: WT(X)**
    Timestamp of youngest transaction that wrote X



# Read Requests

If $T_1$ requests to **read** X:

- **Abort & restart $T_1$** if WT(X) > TS($T_1$)

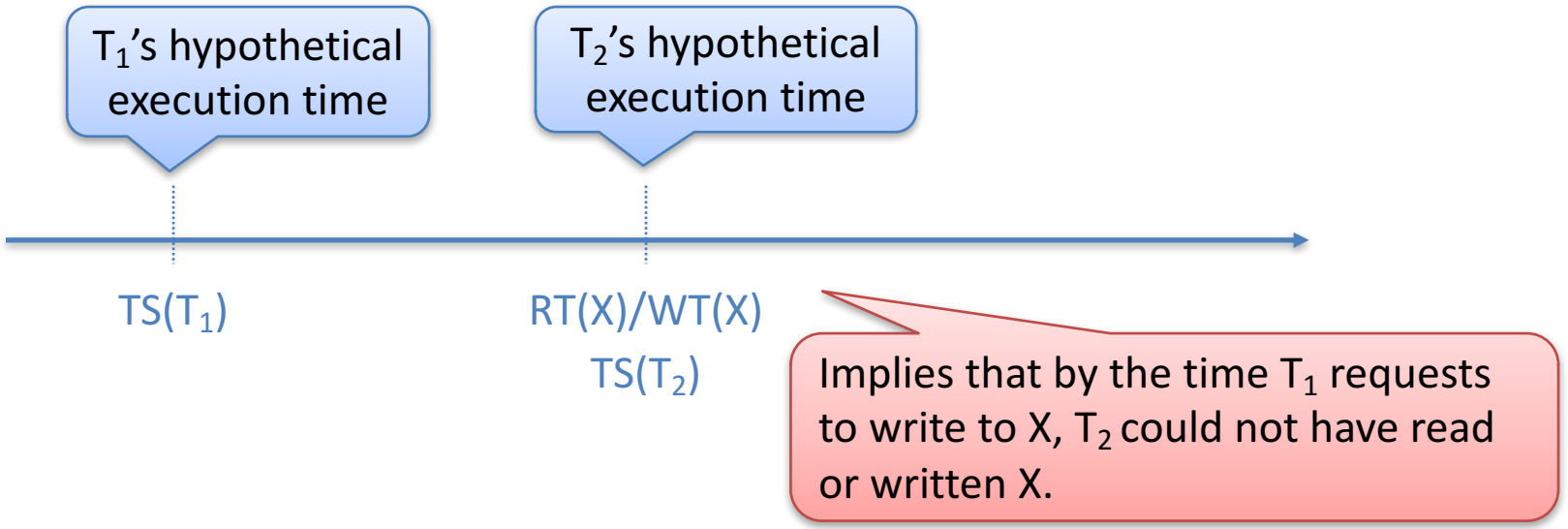



- **Grant request** otherwise

# Write Requests

If $T_1$ requests to **write** X:

- **Abort & restart $T_1$** if RT(X) > TS($T_1$) or WT(X) > TS($T_1$)



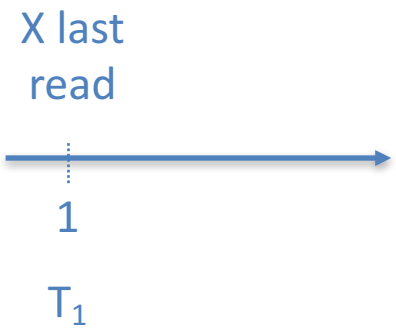

- **Grant request** otherwise

# Example 1

| | | | X | | Y | |
|---|---|---|---|---|---|---|
| **Time** | **$T_1$ (TS = 1)** | **$T_2$ (TS = 2)** | **RT** | **WT** | **RT** | **WT** |
| 0 | | | 0 | 0 | 0 | 0 |
| 1 | | | | | | |
| 2 | | | | | | |
| 3 | | | | | | |
| 4 | | | | | | |
| 5 | | | | | | |
| 6 | | | | | | |
| 7 | | | | | | |
| 8 | | | | | | |
| 9 | | | | | | |
| 10 | | | | | | |
| 11 | | | | | | |

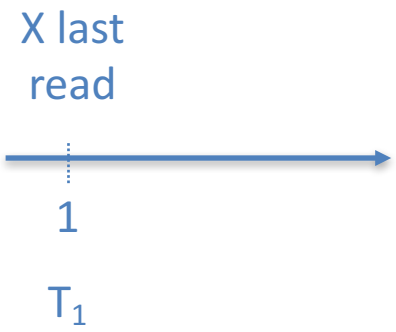
# Example 1

| | | | X | | Y | |
|---|---|---|---|---|---|---|
| **Time** | **$T_1$ (TS = 1)** | **$T_2$ (TS = 2)** | **RT** | **WT** | **RT** | **WT** |
| 0 | | | 0 | 0 | 0 | 0 |
| 1 | read_item(X) | | **1** | 0 | 0 | 0 |
| 2 | | | | | | |
| 3 | | | | | | |
| 4 | | | | | | |
| 5 | | | | | | |
| 6 | | | | | | |
| 7 | | | | | | |
| 8 | | | | | | |
| 9 | | | | | | |
| 10 | | | | | | |
| 11 | | | | | | |



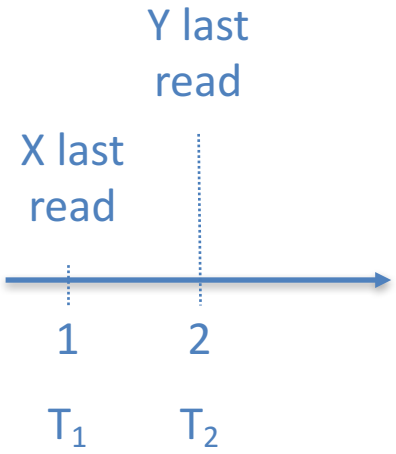
# Example 1

| | | | X | | Y | |
|---|---|---|---|---|---|---|
| **Time** | **$T_1$ (TS = 1)** | **$T_2$ (TS = 2)** | **RT** | **WT** | **RT** | **WT** |
| 0 | | | 0 | 0 | 0 | 0 |
| 1 | read_item(X) | | **1** | 0 | 0 | 0 |
| 2 | X := X + 100 | | 1 | 0 | 0 | 0 |
| 3 | | | | | | |
| 4 | | | | | | |
| 5 | | | | | | |
| 6 | | | | | | |
| 7 | | | | | | |
| 8 | | | | | | |
| 9 | | | | | | |
| 10 | | | | | | |
| 11 | | | | | | |



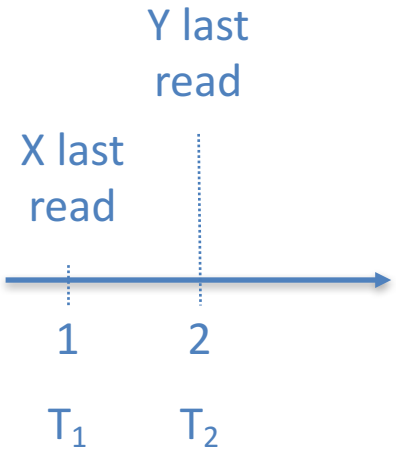
# Example 1

| | | | X | | Y | |
|---|---|---|---|---|---|---|
| **Time** | **$T_1$ (TS = 1)** | **$T_2$ (TS = 2)** | **RT** | **WT** | **RT** | **WT** |
| 0 | | | 0 | 0 | 0 | 0 |
| 1 | read_item(X) | | **1** | 0 | 0 | 0 |
| 2 | X := X + 100 | | 1 | 0 | 0 | 0 |
| 3 | | read_item(Y) | 1 | 0 | **2** | 0 |
| 4 | | | | | | |
| 5 | | | | | | |
| 6 | | | | | | |
| 7 | | | | | | |
| 8 | | | | | | |
| 9 | | | | | | |
| 10 | | | | | | |
| 11 | | | | | | |



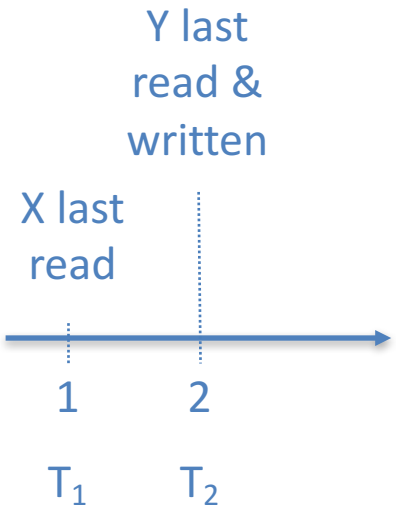
# Example 1

| | | | X | | Y | |
|---|---|---|---|---|---|---|
| **Time** | **$T_1$ (TS = 1)** | **$T_2$ (TS = 2)** | **RT** | **WT** | **RT** | **WT** |
| 0 | | | 0 | 0 | 0 | 0 |
| 1 | read_item(X) | | **1** | 0 | 0 | 0 |
| 2 | X := X + 100 | | 1 | 0 | 0 | 0 |
| 3 | | read_item(Y) | 1 | 0 | **2** | 0 |
| 4 | | Y := Y * 2 | 1 | 0 | 2 | 0 |
| 5 | | | | | | |
| 6 | | | | | | |
| 7 | | | | | | |
| 8 | | | | | | |
| 9 | | | | | | |
| 10 | | | | | | |
| 11 | | | | | | |



# Example 1

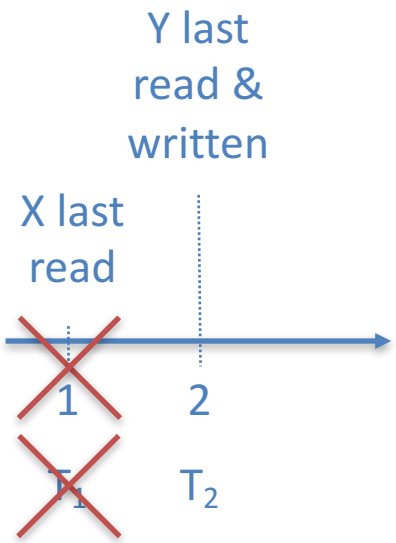
| | | | X | | Y | |
|---|---|---|---|---|---|---|
| **Time** | **$T_1$ (TS = 1)** | **$T_2$ (TS = 2)** | **RT** | **WT** | **RT** | **WT** |
| 0 | | | 0 | 0 | 0 | 0 |
| 1 | read_item(X) | | **1** | 0 | 0 | 0 |
| 2 | X := X + 100 | | 1 | 0 | 0 | 0 |
| 3 | | read_item(Y) | 1 | 0 | **2** | 0 |
| 4 | | Y := Y * 2 | 1 | 0 | 2 | 0 |
| 5 | | write_item(Y) | 1 | 0 | 2 | **2** |
| 6 | | | | | | |
| 7 | | | | | | |
| 8 | | | | | | |
| 9 | | | | | | |
| 10 | | | | | | |
| 11 | | | | | | |



# Example 1

| | | | X | | Y | |
|---|---|---|---|---|---|---|
| **Time** | **$T_1$ (TS = 1)** | **$T_2$ (TS = 2)** | **RT** | **WT** | **RT** | **WT** |
| 0 | | | 0 | 0 | 0 | 0 |
| 1 | read_item(X) | | **1** | 0 | 0 | 0 |
| 2 | X := X + 100 | | 1 | 0 | 0 | 0 |
| 3 | | read_item(Y) | 1 | 0 | **2** | 0 |
| 4 | | Y := Y * 2 | 1 | 0 | 2 | 0 |
| 5 | | write_item(Y) | 1 | 0 | 2 | **2** |
| 6 | read_item(Y) | | | | | |
| 7 | | | | | | |
| 8 | | | | | | |
| 9 | | | | | | |
| 10 | | | | | | |
| 11 | | | | | | |

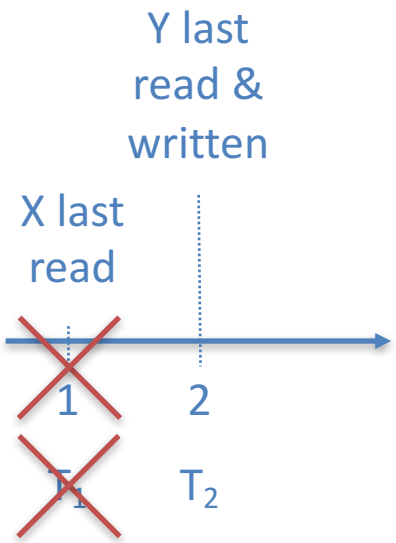
Must abort & restart $T_1$



# Example 1

New timestamp

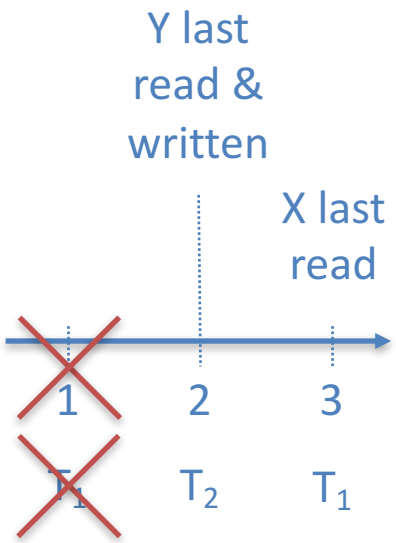
| | | | X | | Y | |
|---|---|---|---|---|---|---|
| **Time** | **$T_1$ (TS = 3)** | **$T_2$ (TS = 2)** | **RT** | **WT** | **RT** | **WT** |
| 0 | | | 0 | 0 | 0 | 0 |
| 1 | read_item(X) | | **1** | 0 | 0 | 0 |
| 2 | X := X + 100 | | 1 | 0 | 0 | 0 |
| 3 | | read_item(Y) | 1 | 0 | **2** | 0 |
| 4 | | Y := Y * 2 | 1 | 0 | 2 | 0 |
| 5 | | write_item(Y) | 1 | 0 | 2 | **2** |
| 6 | ~~read_item(Y)~~ | | **0** | 0 | 2 | 2 |
| 7 | | | | | | |
| 8 | | | | | | |
| 9 | | | | | | |
| 10 | | | | | | |
| 11 | | | | | | |



# Example 1

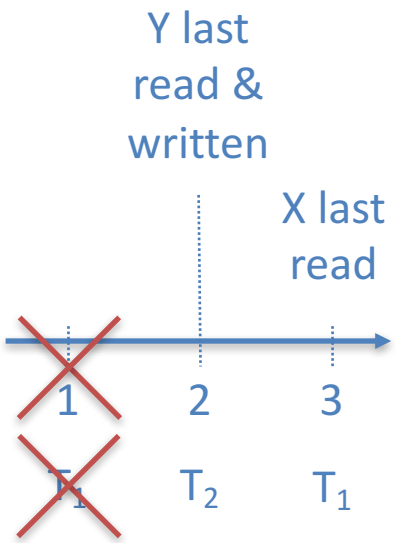
New timestamp

| | | | X | | Y | |
|---|---|---|---|---|---|---|
| **Time** | **$T_1$ (TS = 3)** | **$T_2$ (TS = 2)** | **RT** | **WT** | **RT** | **WT** |
| 0 | | | 0 | 0 | 0 | 0 |
| 1 | read_item(X) | | **1** | 0 | 0 | 0 |
| 2 | X := X + 100 | | 1 | 0 | 0 | 0 |
| 3 | | read_item(Y) | 1 | 0 | **2** | 0 |
| 4 | | Y := Y * 2 | 1 | 0 | 2 | 0 |
| 5 | | write_item(Y) | 1 | 0 | 2 | **2** |
| 6 | ~~read_item(Y)~~ | | **0** | 0 | 2 | 2 |
| 7 | read_item(X) | | **3** | 0 | 2 | 2 |
| 8 | | | | | | |
| 9 | | | | | | |
| 10 | | | | | | |
| 11 | | | | | | |



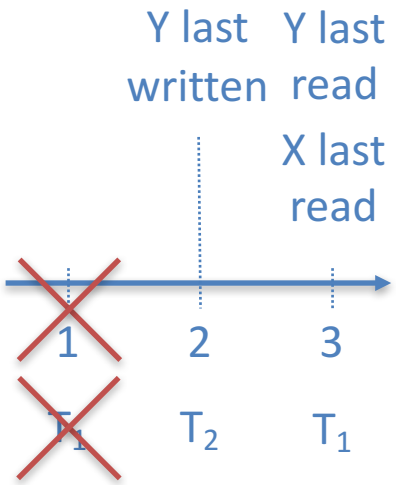
# Example 1

| | | | X | | Y | |
|---|---|---|---|---|---|---|
| **Time** | **$T_1$ (TS = 3)** | **$T_2$ (TS = 2)** | **RT** | **WT** | **RT** | **WT** |
| 0 | | | 0 | 0 | 0 | 0 |
| 1 | read_item(X) | | **1** | 0 | 0 | 0 |
| 2 | X := X + 100 | | 1 | 0 | 0 | 0 |
| 3 | | read_item(Y) | 1 | 0 | **2** | 0 |
| 4 | | Y := Y * 2 | 1 | 0 | 2 | 0 |
| 5 | | write_item(Y) | 1 | 0 | 2 | **2** |
| 6 | ~~read_item(Y)~~ | | **0** | 0 | 2 | 2 |
| 7 | read_item(X) | | **3** | 0 | 2 | 2 |
| 8 | X := X + 100 | | 3 | 0 | 2 | 2 |
| 9 | | | | | | |
| 10 | | | | | | |
| 11 | | | | | | |



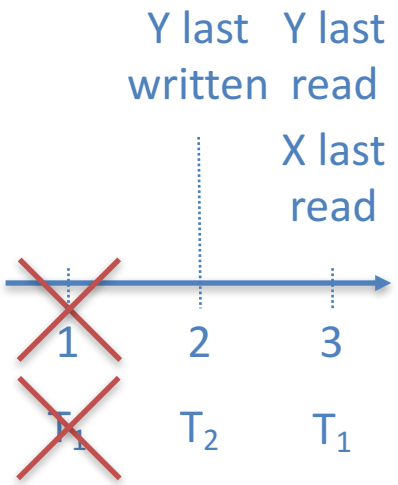
# Example 1

| | | | X | | Y | |
|---|---|---|---|---|---|---|
| **Time** | **$T_1$ (TS = 3)** | **$T_2$ (TS = 2)** | **RT** | **WT** | **RT** | **WT** |
| 0 | | | 0 | 0 | 0 | 0 |
| 1 | read_item(X) | | **1** | 0 | 0 | 0 |
| 2 | X := X + 100 | | 1 | 0 | 0 | 0 |
| 3 | | read_item(Y) | 1 | 0 | **2** | 0 |
| 4 | | Y := Y * 2 | 1 | 0 | 2 | 0 |
| 5 | | write_item(Y) | 1 | 0 | 2 | **2** |
| 6 | ~~read_item(Y)~~ | | **0** | 0 | 2 | 2 |
| 7 | read_item(X) | | **3** | 0 | 2 | 2 |
| 8 | X := X + 100 | | 3 | 0 | 2 | 2 |
| 9 | read_item(Y) | | 3 | 0 | **3** | 2 |
| 10 | | | | | | |
| 11 | | | | | | |



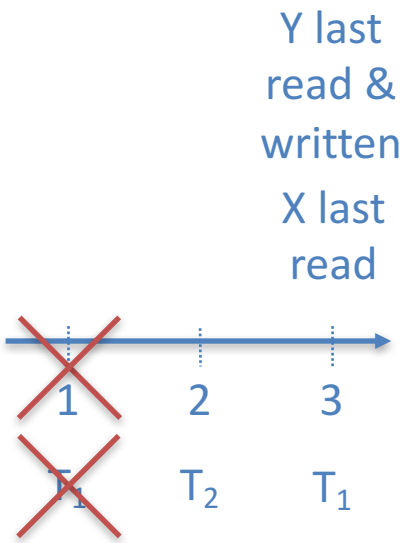
# Example 1

| | | | X | | Y | |
|---|---|---|---|---|---|---|
| **Time** | **$T_1$ (TS = 3)** | **$T_2$ (TS = 2)** | **RT** | **WT** | **RT** | **WT** |
| 0 | | | 0 | 0 | 0 | 0 |
| 1 | read_item(X) | | **1** | 0 | 0 | 0 |
| 2 | X := X + 100 | | 1 | 0 | 0 | 0 |
| 3 | | read_item(Y) | 1 | 0 | **2** | 0 |
| 4 | | Y := Y * 2 | 1 | 0 | 2 | 0 |
| 5 | | write_item(Y) | 1 | 0 | 2 | **2** |
| 6 | ~~read_item(Y)~~ | | **0** | 0 | 2 | 2 |
| 7 | read_item(X) | | **3** | 0 | 2 | 2 |
| 8 | X := X + 100 | | 3 | 0 | 2 | 2 |
| 9 | read_item(Y) | | 3 | 0 | **3** | 2 |
| 10 | Y := Y * 3 | | 3 | 0 | 3 | 2 |
| 11 | | | | | | |



# Example 1

| | | | X | | Y | |
|---|---|---|---|---|---|---|
| **Time** | **$T_1$ (TS = 3)** | **$T_2$ (TS = 2)** | **RT** | **WT** | **RT** | **WT** |
| 0 | | | 0 | 0 | 0 | 0 |
| 1 | read_item(X) | | **1** | 0 | 0 | 0 |
| 2 | X := X + 100 | | 1 | 0 | 0 | 0 |
| 3 | | read_item(Y) | 1 | 0 | **2** | 0 |
| 4 | | Y := Y * 2 | 1 | 0 | 2 | 0 |
| 5 | | write_item(Y) | 1 | 0 | 2 | **2** |
| 6 | ~~read_item(Y)~~ | | **0** | 0 | 2 | 2 |
| 7 | read_item(X) | | **3** | 0 | 2 | 2 |
| 8 | X := X + 100 | | 3 | 0 | 2 | 2 |
| 9 | read_item(Y) | | 3 | 0 | **3** | 2 |
| 10 | Y := Y * 3 | | 3 | 0 | 3 | 2 |
| 11 | write_item(Y) | | 3 | 0 | 3 | **3** |



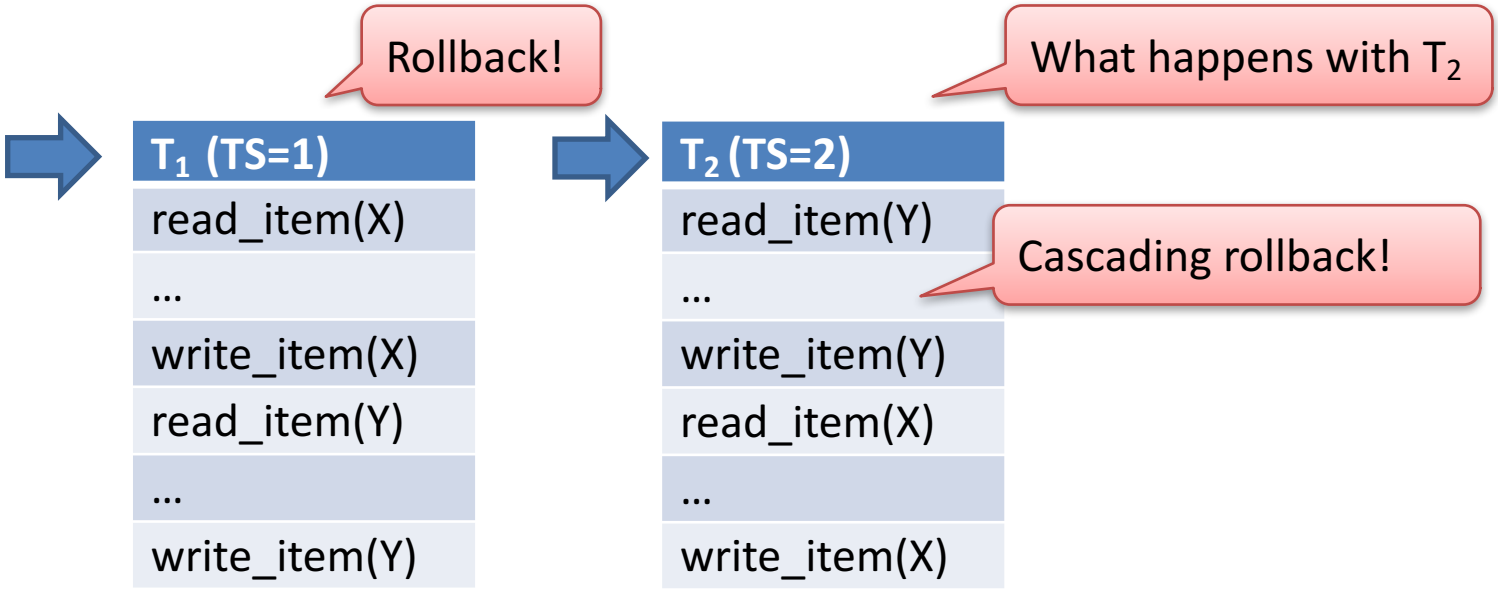
# Example 2

| $T_1$ |
| --- |
| read_item(X) |
| … |
| write_item(X) |
| read_item(Y) |
| … |
| write_item(Y) |

| $T_2$ |
| --- |
| read_item(Y) |
| … |
| write_item(Y) |
| read_item(X) |
| … |
| write_item(X) |

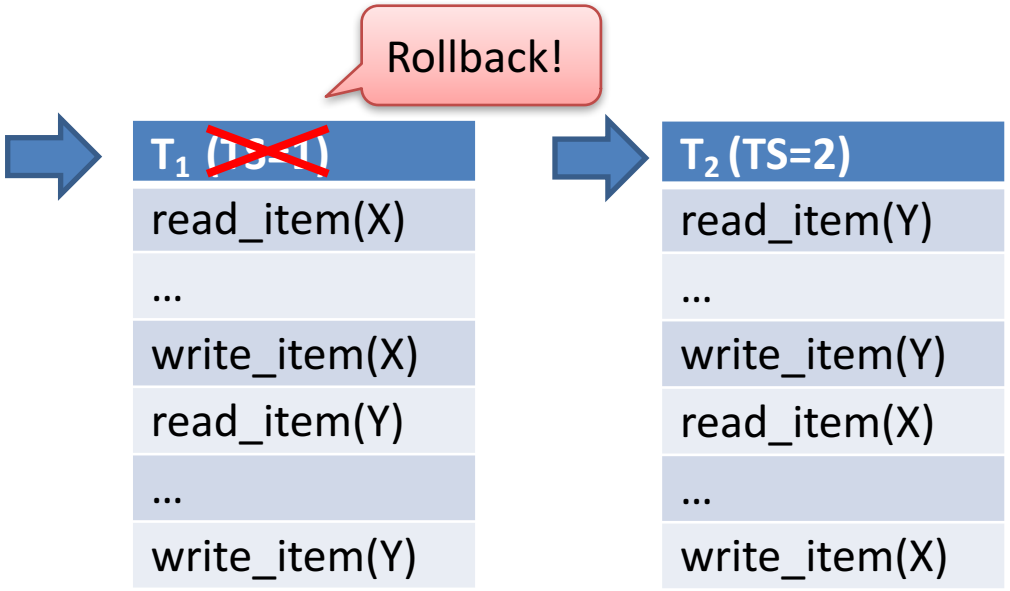
- Assume a timestamp-based scheduler
  - $T_1$ starts first
  - Lines 1-3 of $T_1$ are executed first, then lines 1-3 of $T_2$
- Which operations could be executed next?

# Example 2



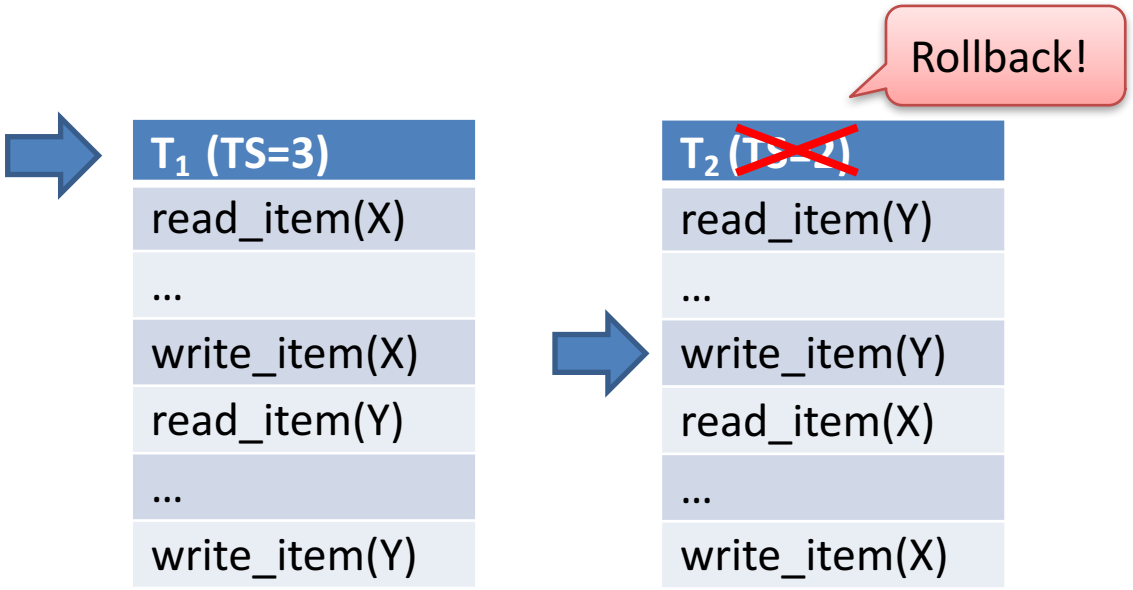

- Assume a timestamp-based scheduler
  - $T_1$ starts first
  - Lines 1-3 of $T_1$ are executed first, then lines 1-3 of $T_2$
- Which operations could be executed next?

# Example 2



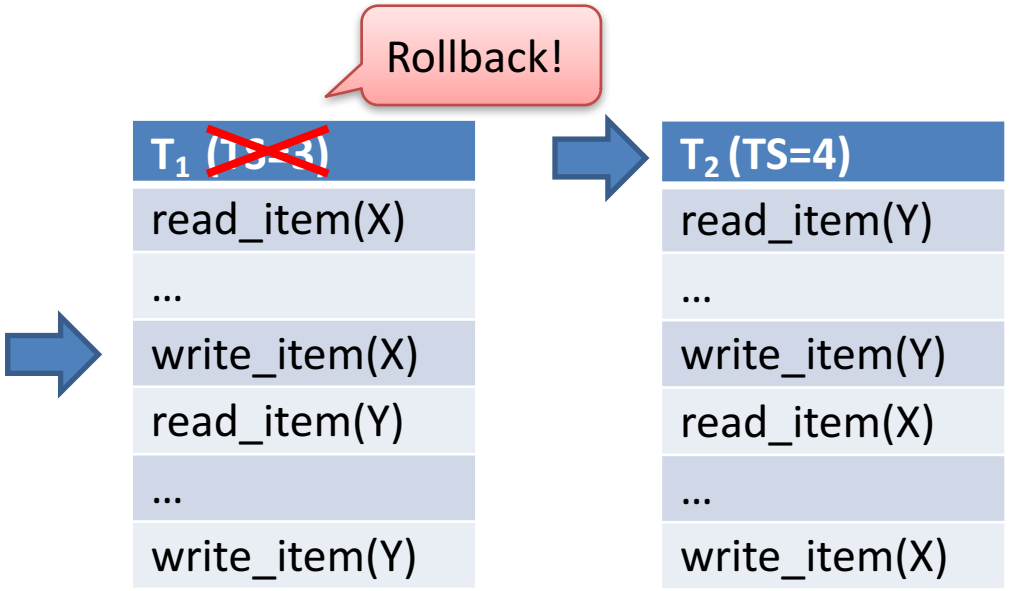

- Assume a timestamp-based scheduler
  - $T_1$ starts first
  - Lines 1-3 of $T_1$ are executed first, then lines 1-3 of $T_2$
- Which operations could be executed next?

# Example 2




- Assume a timestamp-based scheduler
  - $T_1$ starts first
  - Lines 1-3 of $T_1$ are executed first, then lines 1-3 of $T_2$
- Which operations could be executed next?

# Example 2




- Assume a timestamp-based scheduler
  - $T_1$ starts first
  - Lines 1-3 of $T_1$ are executed first, then lines 1-3 of $T_2$
- Which operations could be executed next?

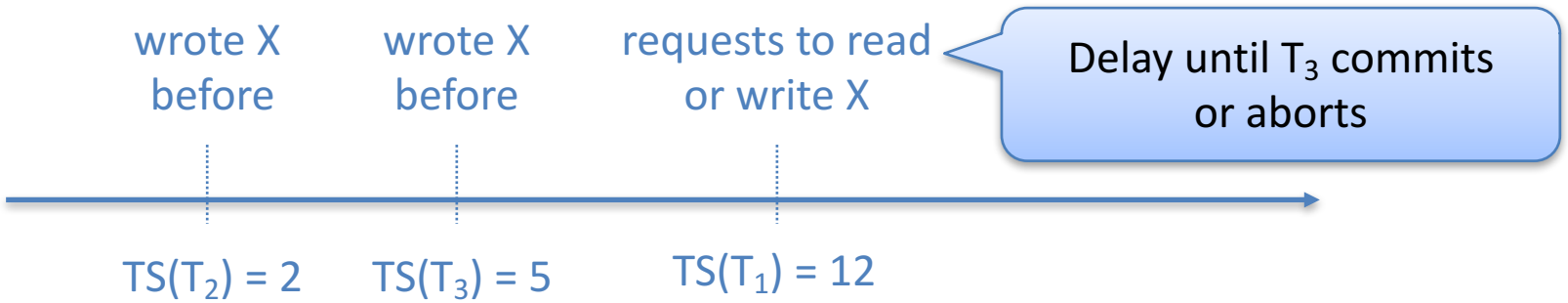
# Timestamp-based Scheduling

- Nice properties:
  - Enforces conflict-serialisable schedules
  - Deadlocks don't occur
- Bad properties:
  - **Cascading rollbacks**
  - **Starvation can occur** (cyclic aborts & restarts of transactions)
- Starvation can be prevented using appropriate techniques (not in this module)

# Ensuring Strictness

- Schedules enforced by timestamp-based schedulers are not strict.
- Additional condition to enforce a **strict schedule**:

> *Delay read or write requests* until the youngest transaction who wrote X before has committed or aborted.



# Locking or Timestamping?

# Timestamping vs Locking

- Space usage roughly the same
  - Timestamping: manage timestamps & read/write times
  - Locking: manage locks
- Optimistic vs conservative
  - **Timestamping is optimistic**: “unserialisable behaviour will not occur, or occurs rarely and can then be fixed”
  - **Locking is conservative**: “unserialisable behaviour will occur unless we take precautions”
- Both have issues
  - Locking may cause **deadlocks** (but can be fixed)
  - Timestamping may cause **starvation** (but can be fixed)

# Timestamping or Locking?

Why might this be the case?

- Timestamping might be preferable if there is little interaction between transactions, e.g.,
  - Transactions are read-only (SELECT-FROM-WHERE)
  - Transactions access distinct database items
- Some DBMSs use a combination of locking and timestamping
  - Locking for transactions that read & write
  - Timestamping for read-only transactions