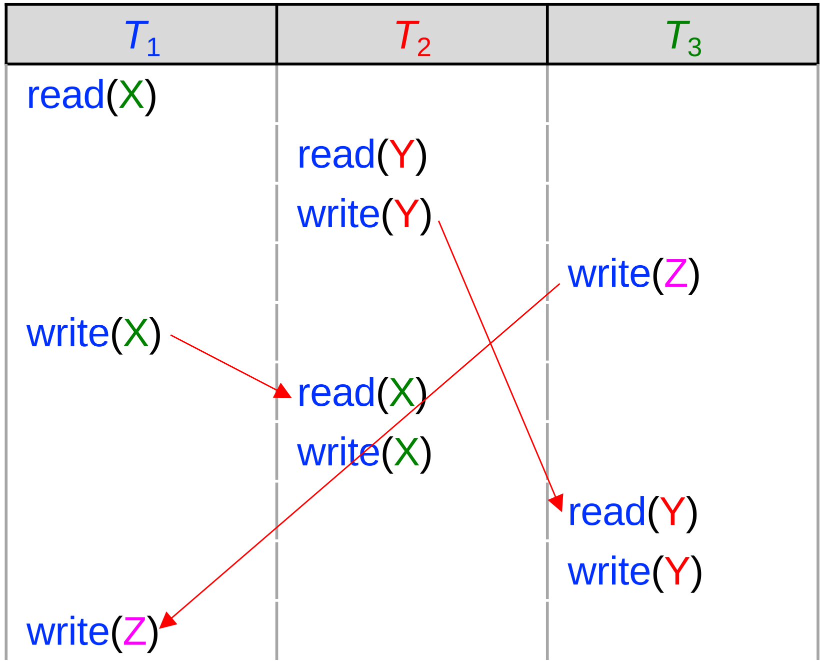
# COMP 3311 DATABASE MANAGEMENT SYSTEMS

## LECTURE 21 EXERCISES CONCURRENCY CONTROL: TIMESTAMP-BASED PROTOCOLS

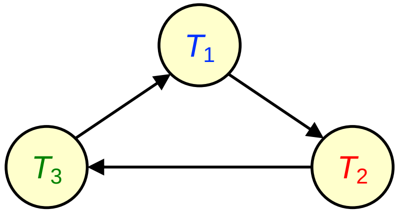
# EXERCISE 1

Recall that this schedule is not serializable because there is a cycle $T_1 T_2 T_3 T_1$. Therefore, the schedule will fail under any protocol that aims at conflict serializability.



| $T_1$ | $T_2$ | $T_3$ |
|---|---|---|
| read(X) | | |
| | read(Y) | |
| | write(Y) | |
| | | write(Z) |
| write(X) | | |
| | read(X) | |
| | write(X) | |
| | | read(Y) |
| | | write(Y) |
| write(Z) | | |

Precedence Graph



☞ **The timestamps determine the serializability order.**

# EXERCISE 1 (cont'd)

Use the single-version, timestamp-ordering protocol, to complete the following non-serializable schedule assuming the timestamps 1, 2, and 3 for transactions $T_1$, $T_2$, and $T_3$, respectively. Show where the protocol will fail. Assume initial R/W timestamp of all items is 0.

**Read**
If TS($T_i$) < WTS($Q$) **rollback**
If TS($T_i$) ≥ WTS($Q$)
RTS($Q$) = max(TS($T_i$), RTS($Q$))

**Write**
If TS($T_i$) < RTS($Q$) **rollback**
If TS($T_i$) < WTS($Q$) **rollback**
Otherwise WTS($Q$) = TS($T_i$)

| | | | | | |
|---|---|---|---|---|---|
| RTS(X)=2 | WTS(X)=2 | RTS(Y)=3 | WTS(Y)=3 | RTS(Z)=0 | WTS(Z)=3 |

| $T_1$ [TS=1] | $T_2$ [TS=2] | $T_3$ [TS=3] |
|---|---|---|
| read(X) ✓ TS($T_1$)=1 ≥ WTS(X)=0; **set** RTS(X)=1 | | |
| | read(Y) ✓ TS($T_2$)=2 ≥ WTS(Y)=0; **set** RTS(Y)=2 | |
| | write(Y) ✓ TS($T_2$)=2 ≥ RTS(Y)=2 & ≥ WTS(Y)=0; **set** WTS(Y)=2 | |
| | | write(Z) ✓ TS($T_3$)=3 ≥ RTS(Z)=0 & ≥ WTS(Z)=0; **set** WTS(Z)=3 |
| write(X) ✓ TS($T_1$)=1 ≥ RTS(X)=1 & ≥ WTS(X)=0; **set** WTS(X)=1 | | |
| | read(X) ✓ TS($T_2$)=2 ≥ WTS(X)=1; **set** RTS(X)=2 | |
| | write(X) ✓ TS($T_2$)=2 ≥ RTS(X)=2 & ≥ WTS(X)=1; **set** WTS(X)=2 | |
| | | read(Y) ✓ TS($T_3$)=3 ≥ WTS(Y)=2; **set** RTS(Y)=3 |
| | | write(Y) ✓ TS($T_3$)=3 ≥ RTS(Y)=3 & ≥ WTS(Y)=2; **set** WTS(Y)=3 |
| write(Z) TS($T_1$)=1 < WTS(Z)=3 ⇒ **rollback** | | |

☞ **The timestamps determine the serializability order.**

# EXERCISE 1 (cont'd)

Use the single-version, timestamp-ordering protocol, to complete the following non-serializable schedule assuming the timestamps 1, 2, and 3 for transactions $T_1$, $T_2$, and $T_3$, respectively. Show where the protocol will fail. Assume initial R/W timestamp of all items is 0.

**Read**
If TS($T_i$) < WTS($Q$) **rollback**
If TS($T_i$) ≥ WTS($Q$)
RTS($Q$) = max(TS($T_i$), RTS($Q$))

**Write**
If TS($T_i$) < RTS($Q$) **rollback**
If TS($T_i$) < WTS($Q$) **ignore**
Otherwise WTS($Q$) = TS($T_i$)

| | | | | | |
|---|---|---|---|---|---|
| RTS(X)=2 | WTS(X)=2 | RTS(Y)=3 | WTS(Y)=3 | RTS(Z)=0 | WTS(Z)=3 |

| $T_1$ [TS=1] | $T_2$ [TS=2] | $T_3$ [TS=3] |
|---|---|---|
| read(X) ✓ TS($T_1$)=1 ≥ WTS(X)=0; **set** RTS(X)=1 | | |
| | read(Y) ✓ TS($T_2$)=2 ≥ WTS(Y)=0; **set** RTS(Y)=2 | |
| | write(Y) ✓ TS($T_2$)=2 ≥ RTS(Y)=2 & ≥ WTS(Y)=0; **set** WTS(Y)=2 | |
| | | write(Z) ✓ TS($T_3$)=3 ≥ RTS(Z)=0 & ≥ WTS(Z)=0; **set** WTS(Z)=3 |
| write(X) ✓ TS($T_1$)=1 ≥ RTS(X)=1 & ≥ WTS(X)=0; **set** WTS(X)=1 | | |
| | read(X) ✓ TS($T_2$)=2 ≥ WTS(X)=1; **set** RTS(X)=2 | |
| | write(X) ✓ TS($T_2$)=2 ≥ RTS(X)=2 & ≥ WTS(X)=1; **set** WTS(X)=2 | |
| | | read(Y) ✓ TS($T_3$)=3 ≥ WTS(Y)=2; **set** RTS(Y)=3 |
| | | write(Y) ✓ TS($T_3$)=3 ≥ RTS(Y)=3 & ≥ WTS(Y)=2; **set** WTS(Y)=3 |
| write(Z) TS($T_1$)=1 < WTS(Z)=3 ⇒ **ignore** | | |

$Q_k$ is the version of $Q$ whose write timestamp is the largest write timestamp *less than or equal to* TS($T_i$).

# EXERCISE 2

Use the multi-version, timestamp-ordering protocol to complete the schedule of Exercise 1 assuming the timestamps 1, 2, and 3 for transactions $T_1$, $T_2$, and $T_3$, respectively. Assume initial R/W timestamp of $X_0$, $Y_0$ and $Z_0$ is 0. ☞ Timestamps are used to label versions.

| | |
|---|---|
| RTS($X_0$)=1 | WTS($X_0$)=0 |
| RTS($X_1$)=2 | WTS($X_1$)=1 |
| RTS($X_2$)=2 | WTS($X_2$)=2 |

| | |
|---|---|
| RTS($Y_0$)=2 | WTS($Y_0$)=0 |
| RTS($Y_1$)=3 | WTS($Y_1$)=2 |
| RTS($Y_2$)=3 | WTS($Y_2$)=3 |

| | |
|---|---|
| RTS($Z_0$)=0 | WTS($Z_0$)=0 |
| RTS($Z_1$)=3 | WTS($Z_1$)=3 |

**Read**

**Reads always succeed**

set RTS($Q_k$) = max(TS($T_i$), RTS($Q_k$))

**Write**

If TS($T_i$) < RTS($Q_k$) **rollback**

If TS($T_i$) = WTS($Q_k$) **overwrite contents**

If TS($T_i$) > WTS($Q_k$) **create new version** set R/WTS($Q'$)=TS($T_i$)

| $T_1$ [TS=1] | $T_2$ [TS=2] | $T_3$ [TS=3] |
|---|---|---|
| read($X_0$) ✓ **set** RTS($X_0$)=1 | | |
| | read($Y_0$) ✓ **set** RTS($Y_0$)=2 | |
| | write($Y_1$) ✓ TS($T_2$)=2 > WTS($Y_0$)=0; **create** $Y_1$; **set** R/WTS($Y_1$)=2 | |
| | | write($Z_1$) ✓ TS($T_3$)=3 > WTS($Z_0$)=0; **create** $Z_1$; **set** R/WTS($Z_1$)=3 |
| write($X_1$) ✓ TS($T_1$)=1 > WTS($X_0$)=0; **create** $X_1$; **set** R/WTS($X_1$)=1 | | |
| | read($X_1$) ✓ **set** RTS($X_1$)=2 | |
| | write($X_2$) ✓ TS($T_2$)=2 > WTS($X_1$)=1; **create** $X_2$; **set** R/WTS($X_2$)=2 | |
| | | read($Y_1$) ✓ **set** RTS($Y_1$)=3 |
| | | write($Y_2$) ✓ TS($T_3$)=3 > WTS($Y_1$)=2; **create** $Y_2$; **set** R/WTS($Y_2$)=3 |
| write(Z) TS($T_1$)=1 > WTS($Z_0$)=0; **create** $Z_2$; **set** R/WTS($Z_2$)=1 | | |

⟹ No transaction will ever read this value so it can be deleted!

$Q_k$ is the version of $Q$ whose write timestamp is the largest write timestamp *less than or equal to* TS($T_i$).

# EXERCISE 2 (CONT'D)

Multi-version timestamp-ordering protocol assuming the timestamps 2, 1 and 3 for transactions $T_1$, $T_2$ and $T_3$, respectively.

**Read**

**Reads always succeed**

set RTS($Q_k$) = max(TS($T_i$), RTS($Q_k$))

**Write**

If TS($T_i$) < RTS($Q_k$) **rollback**

If TS($T_i$) = WTS($Q_k$)
**overwrite contents**

If TS($T_i$) > WTS($Q_k$)
**create new version**
set R/WTS($Q'$)=TS($T_i$)

| | | | | | |
|---|---|---|---|---|---|
| RTS($X_0$)=2 | WTS($X_0$)=0 | RTS($Y_0$)=1 | WTS($Y_0$)=0 | RTS($Z_0$)=0 | WTS($Z_0$)=0 |
| RTS($X_1$)=2 | WTS($X_1$)=2 | RTS($Y_1$)=1 | WTS($Y_1$)=1 | RTS($Z_1$)=3 | WTS($Z_1$)=3 |

| $T_1$ [TS=2] | $T_2$ [TS=1] | $T_3$ [TS=3] |
|---|---|---|
| read($X_0$) ✓ **set** RTS($X_0$)=2 | | |
| | read($Y_0$) ✓ **set** RTS($Y_0$)=1 | |
| | write($Y_1$) ✓ TS($T_2$)=1 > WTS($Y_0$)=0; **create** $Y_1$; **set** R/WTS($Y_1$)=1 | |
| | | write($Z_1$) ✓ TS($T_3$)=3 > WTS($Z_0$)=0; **create** $Z_1$; **set** R/WTS($Z_1$)=3 |
| write($X_1$) ✓ TS($T_1$)=2 > WTS($X_0$)=0; **create** $X_1$; **set** R/WTS($X_1$)=2 | | |
| | read($X_1$) ✓ RTS($X_1$)=2 | |
| | write(X) TS($T_2$)=1 < RTS($X_0$)=2 ⟹ **rollback** | |
| | | read(Y) |
| | | write(Y) |
| write(Z) | | |

Any other timestamp ordering of the transactions, as in this example, will lead to a rollback.

# EXERCISE 3

The following schedule is conflict serializable.

a) What is the equivalent serial schedule?

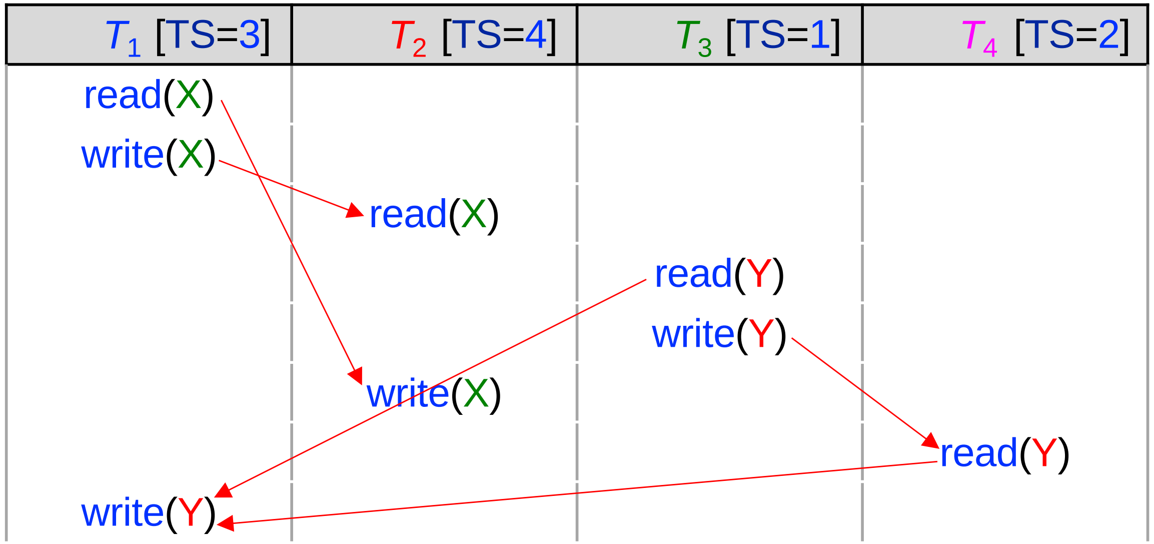
b) Assign appropriate timestamps to the transactions $T_1$, $T_2$, $T_3$ and $T_4$ so that the schedule is conflict serializable according to the single version, timestamp-ordering protocol. Assume initial R/W timestamp of all items is 0.

| $T_1$ | $T_2$ | $T_3$ | $T_4$ |
|---|---|---|---|
| read(X) | | | |
| write(X) | | | |
| | read(X) | | |
| | | read(Y) | |
| | | write(Y) | |
| | write(X) | | |
| | | | read(Y) |
| write(Y) | | | |

☞ The timestamps determine the serializability order.

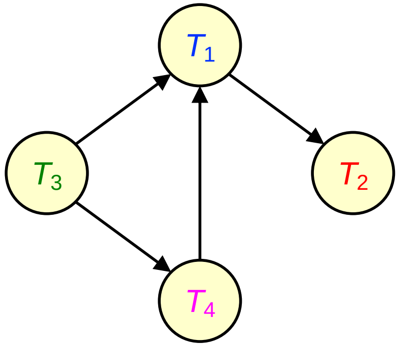
# EXERCISE 3 (CONT'D)

a) What is the equivalent serial schedule?



| $T_1$ [TS=3] | $T_2$ [TS=4] | $T_3$ [TS=1] | $T_4$ [TS=2] |
|---|---|---|---|
| read(X) | | | |
| write(X) | | | |
| | read(X) | | |
| | | read(Y) | |
| | | write(Y) | |
| | write(X) | | |
| | | | read(Y) |
| write(Y) | | | |

## Precedence Graph



The equivalent serial schedule is $T_3$ $T_4$ $T_1$ $T_2$

b) Assign appropriate timestamps to the transactions $T_1$, $T_2$, $T_3$ and $T_4$ so that the schedule is conflict serializable according to the single version, timestamp-ordering protocol. Assume initial R/W timestamp of all items is 0.

☞ The timestamps determine the serializability order.

# EXERCISE 3 (CONT'D)

We note that, for this schedule, any other order of timestamps will fail according to the single version timestamp-ordering protocol as shown below for the order $T_1$, $T_2$, $T_3$, $T_4$.

**Read**
If TS($T_i$) < WTS($Q$) **rollback**
If TS($T_i$) ≥ WTS($Q$)
RTS($Q$) = max(TS($T_i$), RTS($Q$))

**Write**
If TS($T_i$) < RTS($Q$) **rollback**
If TS($T_i$) < WTS($Q$) **ignore**
Otherwise WTS($Q$) = TS($T_i$)

| $T_1$ [TS=1] | $T_2$ [TS=2] | $T_3$ [TS=3] | $T_4$ [TS=4] |
|---|---|---|---|
| read(X) ✓ TS($T_1$)=1 ≥ WTS(X)=0; set RTS(X)=1 | | | |
| write(X) ✓ TS($T_1$)=1 ≥ RTS(X)=1 & ≥ WTS(X)=0; **set** WTS(X)=1 | | | |
| | read(X) ✓ TS($T_2$)=2 ≥ WTS(X)=1; **set** RTS(X)=2 | | |
| | | read(Y) ✓ TS($T_3$)=3 ≥ WTS(Y)=0; **set** RTS(Y)=3 | |
| | | write(Y) ✓ TS($T_3$)=3 ≥ RTS(Y)=3 & ≥ WTS(Y)=0; **set** WTS(Y)=3 | |
| | write(X) ✓ TS($T_2$)=2 ≥ RTS(X)=2 & ≥ WTS(X)=1; **set** WTS(X)=2 | | |
| | | | read(Y) ✓ TS($T_4$)=4 ≥ WTS(Y)=3; **set** RTS(Y)=4 |
| write(Y) TS($T_1$)=1 < RTS(Y)=4 ⇒ **rollback** | | | |

$Q_k$ is the version of $Q$ whose write timestamp is the largest write timestamp *less than or equal to* TS($T_i$).

# EXERCISE 4

Use the multi version, timestamp-ordering protocol, to complete the conflict serializable schedule of **Exercise 3** assuming the timestamps 1, 2, 3, and 4 for transactions $T_1$, $T_2$, $T_3$ and $T_4$, respectively. Show where the protocol will fail. Assume initial R/W timestamp of all items is 0.

| | |
|---|---|
| RTS($X_0$)=1 | WTS($X_0$)=0 |
| RTS($X_1$)=2 | WTS($X_1$)=1 |
| RTS($X_2$)=2 | WTS($X_2$)=2 |

| | |
|---|---|
| RTS($Y_0$)=3 | WTS($Y_0$)=0 |
| RTS($Y_1$)=4 | WTS($Y_1$)=3 |

**Read**

**Reads always succeed**

set RTS($Q_k$) =
max(TS($T_i$), RTS($Q_k$))

**Write**

If TS($T_i$) < RTS($Q_k$) **rollback**

If TS($T_i$) = WTS($Q_k$)
**overwrite contents**

If TS($T_i$) > WTS($Q_k$)
**create new version**
set R/WTS($Q'$)=TS($T_i$)

| $T_1$ [TS=1] | $T_2$ [TS=2] | $T_3$ [TS=3] | $T_4$ [TS=4] |
|---|---|---|---|
| read($X_0$) ✓ **set** RTS($X_0$)=1 | | | |
| write($X_1$) ✓ TS($T_1$)=1 ≥ RTS($X_0$)=1 & > WTS($X_0$)=0; **create** $X_1$; **set** R/WTS($X_1$)=1 | | | |
| | read($X_1$) ✓ **set** RTS($X_1$)=2 | | |
| | | read($Y_0$) ✓ **set** RTS($Y_0$)=3 | |
| | | write($Y_1$) ✓ TS($T_3$)=3 ≥ RTS($Y_0$)=3 & > WTS($Y_0$)=0; **create** $Y_1$; **set** R/WTS($Y_1$)=3 | |
| | write($X_2$) ✓ TS($T_2$)=2 ≥ RTS($X_1$)=2 & > WTS($X_1$)=1; **create** $X_2$; **set** R/WTS($X_2$)=2 | | |
| | | | read($Y_1$) ✓ **set** RTS($Y_1$)=4 |
| write(Y) TS($T_1$)=1 < RTS($Y_0$)=3 ⇒ **rollback** | | | |

☞ The timestamps determine the serializability order.

# EXERCISE 4 (cont'd)

## Why rollback of write(Y) by $T_1$ rather than ignore?

The equivalent serial schedule, according to the timestamps, is $T_1$ $T_2$ $T_3$ $T_4$. Since $T_3$ comes after $T_1$ in the serial schedule, $T_3$ should have read the value of Y written by $T_1$ rather than the value that it read.

☞ **The schedule is not serializable.**

| | |
|---|---|
| RTS($X_0$)=1 | WTS($X_0$)=0 |
| RTS($X_1$)=2 | WTS($X_1$)=1 |
| RTS($X_2$)=2 | WTS($X_2$)=2 |

| | |
|---|---|
| RTS($Y_0$)=3 | WTS($Y_0$)=0 |
| RTS($Y_1$)=4 | WTS($Y_1$)=3 |

| $T_1$ [TS=1] | $T_2$ [TS=2] | $T_3$ [TS=3] | $T_4$ [TS=4] |
|---|---|---|---|
| read($X_0$) | | | |
| write($X_1$) | | | **Which transactions need to be rolled back?** |
| | read($X_1$) | | |
| | | read($Y_0$) | |
| | | write($Y_1$)<br>**⇒ rollback** | |
| | write($X_2$)<br>**⇒ rollback** | | |
| | | | read($Y_1$)<br>**⇒ rollback** |
| write(Y) **⇒ rollback** | | | |

**Read**

**Reads always succeed**

set RTS($Q_k$) = max(TS($T_i$), RTS($Q_k$))

**Write**

If TS($T_i$) < RTS($Q_k$) **rollback**

If TS($T_i$) = WTS($Q_k$)
**overwrite contents**

If TS($T_i$) > WTS($Q_k$)
**create new version**
set R/WTS($Q'$)=TS($T_i$)

**All transactions need to be rolled back!**