Timestamp based schedules

Overview of this video

In the last video, we saw how we could add deadlock-prevention into our Strict 2PL system, by among other approaches, use time stamps

This video, we try to make schedules that ensure no deadlocks directly, by using time-stamps!

Deadlock prevention

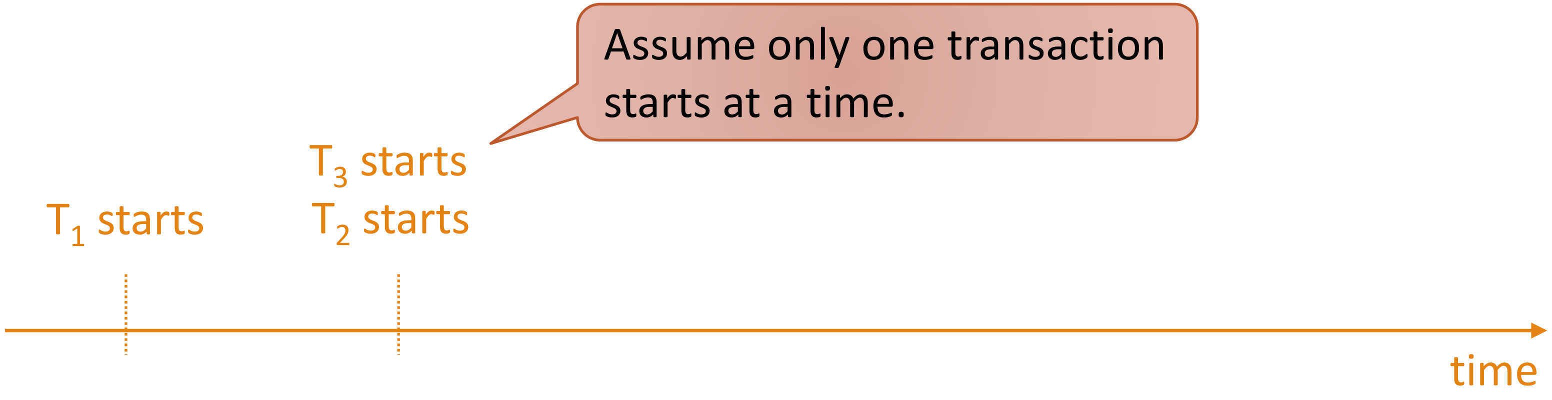
Two approaches for deadlock prevention:

- **Detect deadlocks & fix them** ✓
- **Enforce deadlock-free schedules**

Uses timestamps too!

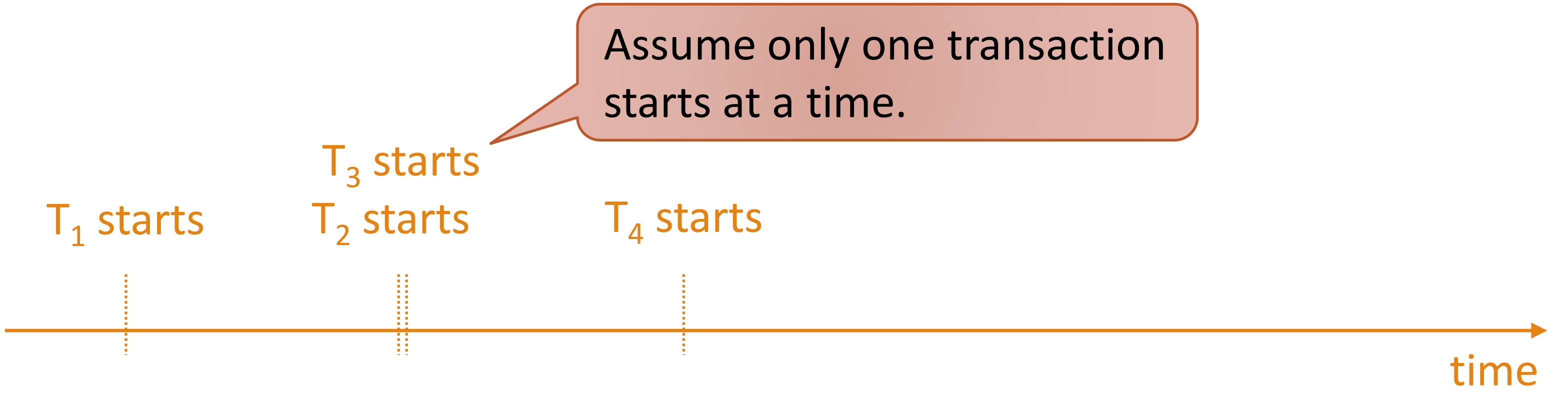
Basic Idea

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



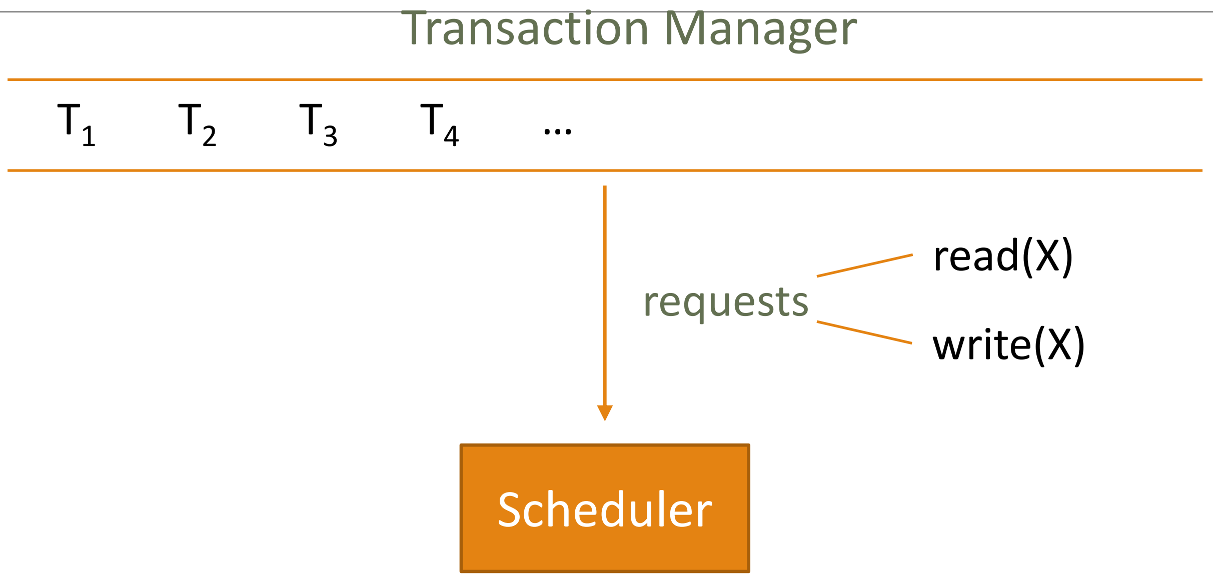
Basic Idea

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

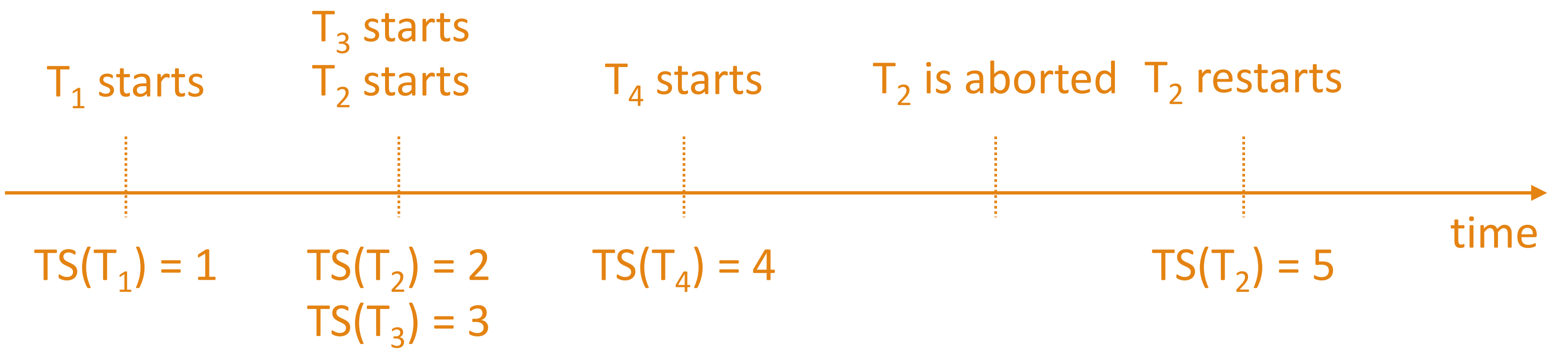
Delay transaction

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 $\mathbf{TS(T_1) < TS(T_2)}$

"T_2 is younger than T_1"



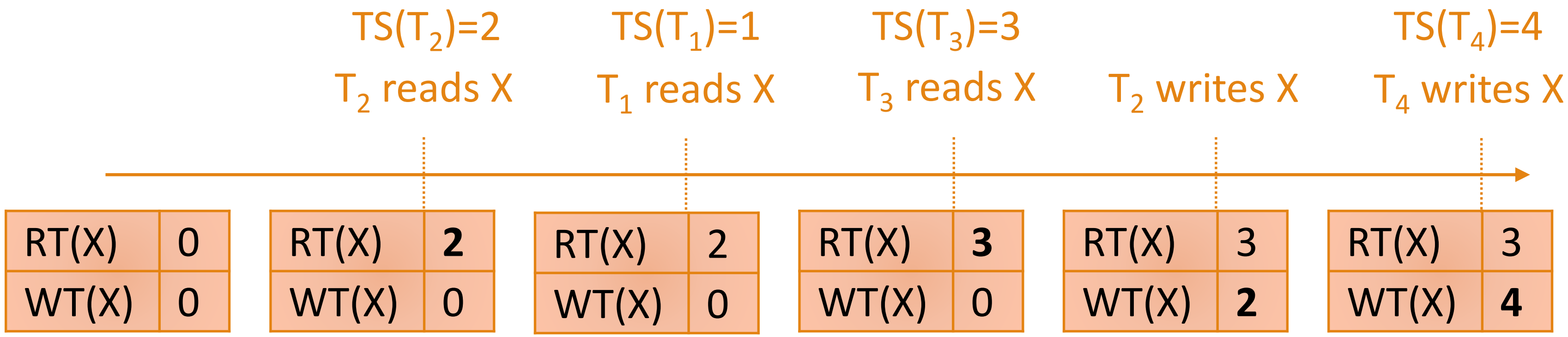
We assign a **new timestamp even after a restart**!

Recall: this is different from detection! There, restarts did not change timestamps!

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



$TS(T_2)=2$
T_2 reads X

$TS(T_1)=1$
T_1 reads X

$TS(T_3)=3$
T_3 reads X

T_2 writes X

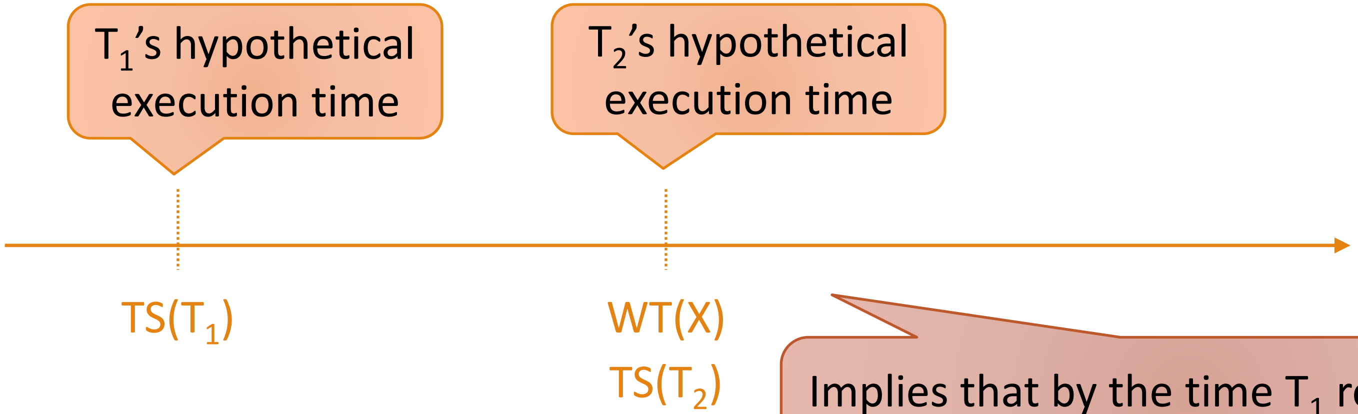
$TS(T_4)=4$
T_4 writes X

RT(X)	0	RT(X)	**2**	RT(X)	2	RT(X)	**3**	RT(X)	3	RT(X)	3
WT(X)	0	WT(X)	0	WT(X)	0	WT(X)	0	WT(X)	**2**	WT(X)	**4**

Read Requests

If T_1 requests to **read** X:

Abort & restart T_1 if WT(X) > TS(T_1)



T_1's hypothetical execution time
T_2's hypothetical execution time
TS(T_1)
WT(X)
TS(T_2)

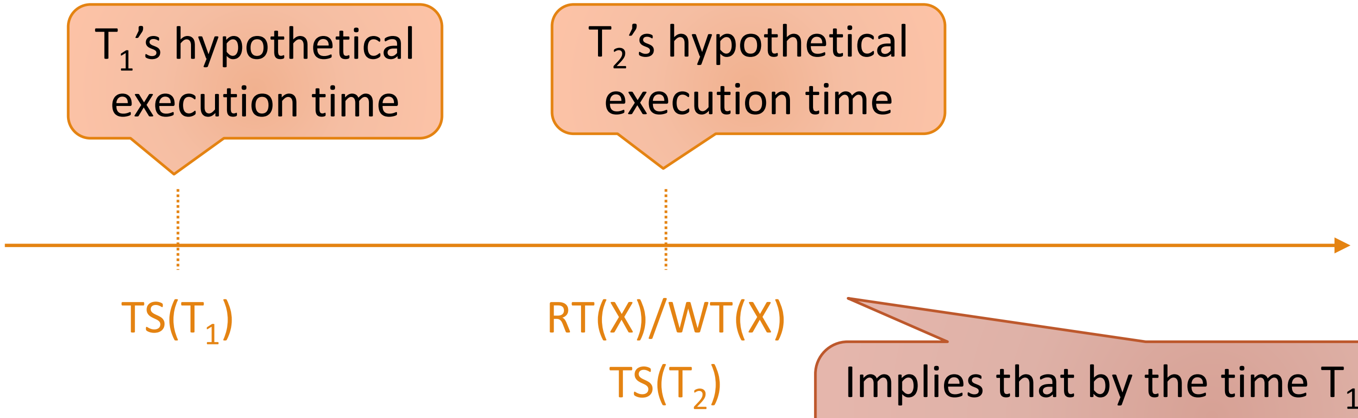
Implies that by the time T_1 requests to read X, T_2 could not have written to X.

Grant request otherwise

Write Requests

If T_1 requests to **write** X:

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


T_1's hypothetical execution time
T_2's hypothetical execution time
$TS(T_1)$
$RT(X)/WT(X)$
$TS(T_2)$

Implies that by the time T_1 requests to write to X, T_2 could not have read or written X.

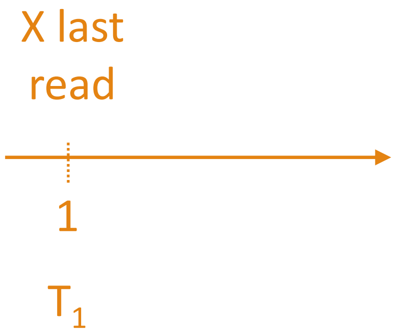
Grant request otherwise

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						

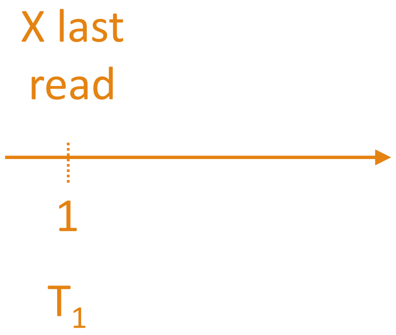
Example 1

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



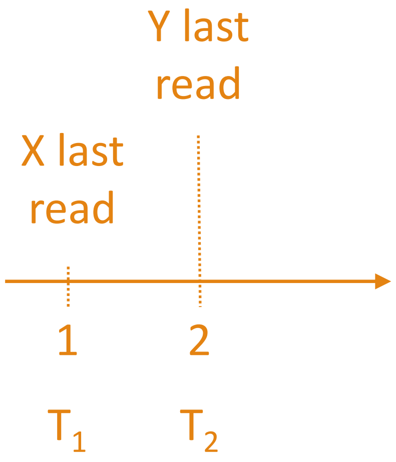
Example 1

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



Y last read

X last read

1

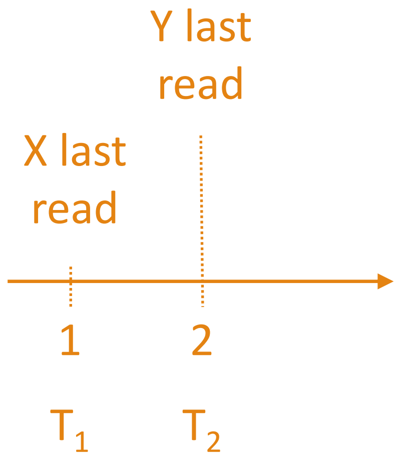
2

T_1

T_2

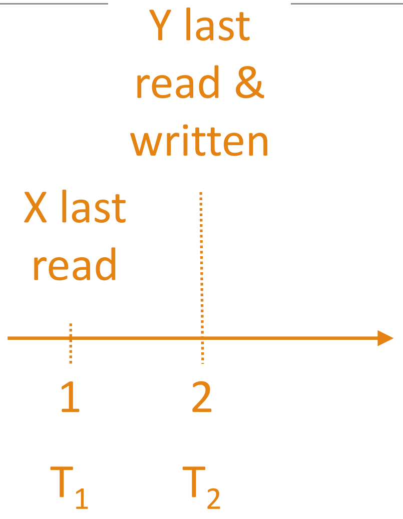
Example 1

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



Y last read & written

X last read

1

2

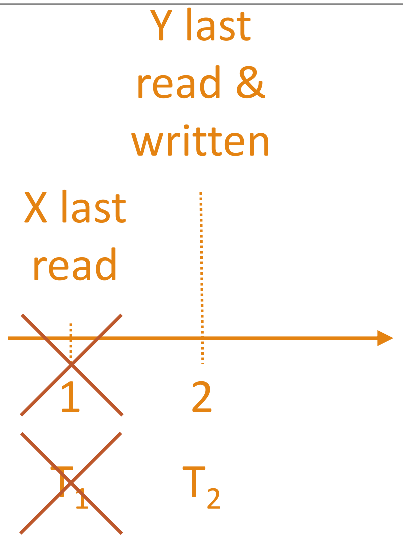
T_1

T_2

Example 1

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

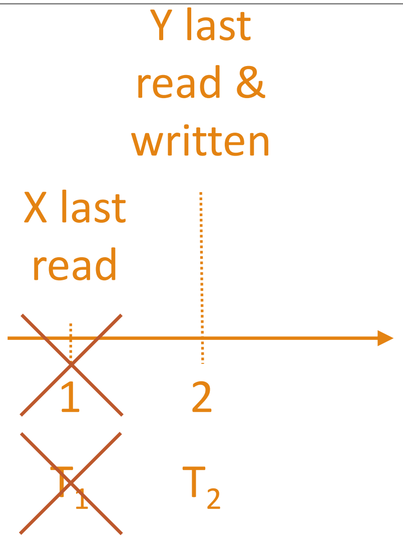
Must abort & restart T_1



Exam

New timestamp

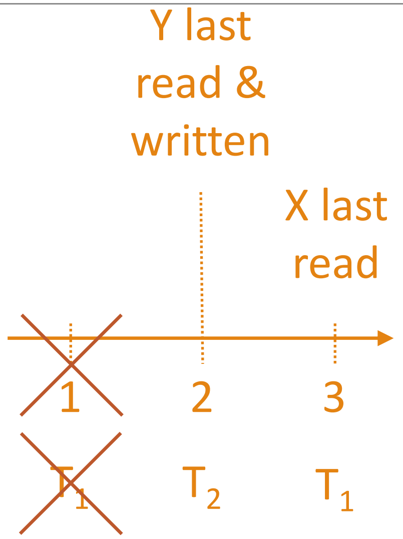
			X		Y	
Time	**T_1 (TS = 3)**	**T_2 (TS = 2)**	**RT**	**WT**	**RT**	**WT**
0			0	0	0	0
1	read(X)		**1**	0	0	0
2	X := X + 100		1	0	0	0
3		read(Y)	1	0	**2**	0
4		Y := Y * 2	1	0	2	0
5		write(Y)	1	0	2	**2**
6	~~read(Y)~~		**0**	0	2	2
7						
8						
9						
10						
11						



Exam

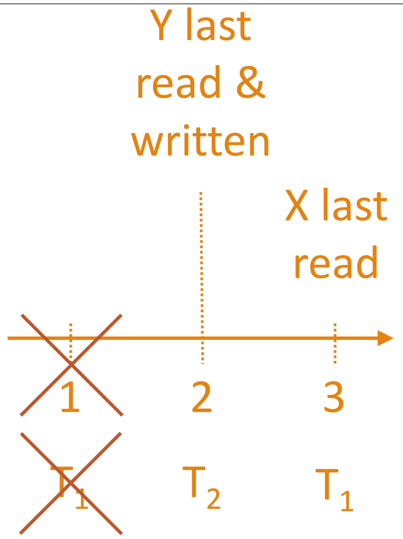
New timestamp

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



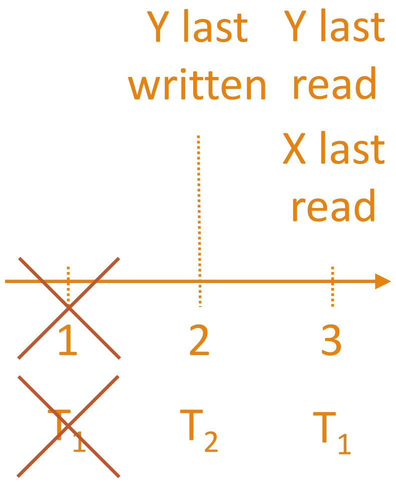
Example 1

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



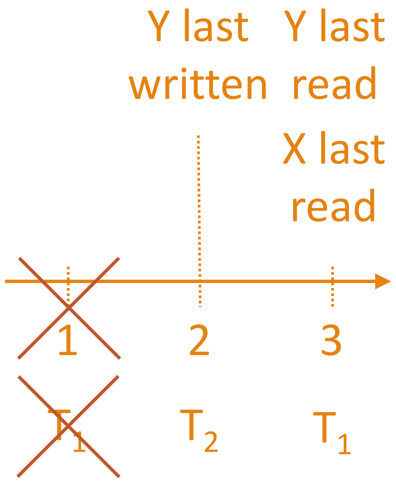
Example 1

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

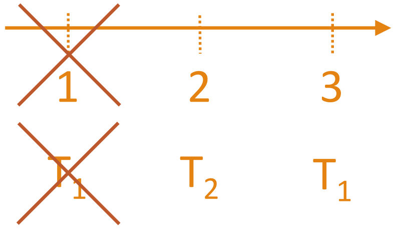


Example 1

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

Y last read & written

X last read



Example 2

T_1
read(X)
…
write(X)
read(Y)
…
write(Y)

T_2
read(Y)
…
write(Y)
read(X)
…
write(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

Rollback!

T_1 (TS=1)
read(X)
…
write(X)
read(Y)
…
write(Y)

What happens with T_2

T_2 (TS=2)
read(Y)
…
write(Y)
read(X)
…
write(X)

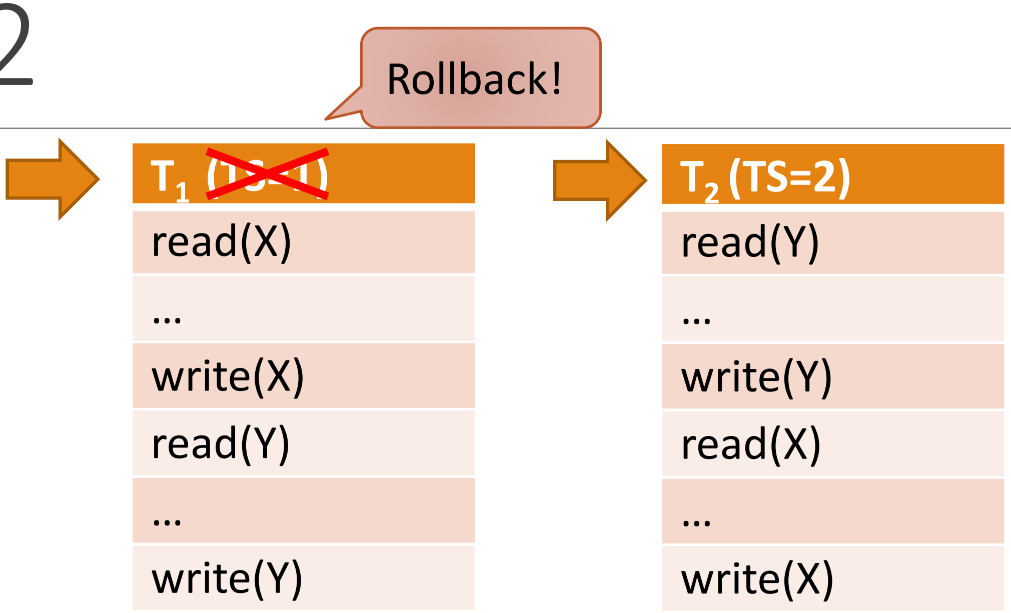
Cascading rollback!

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



Rollback!

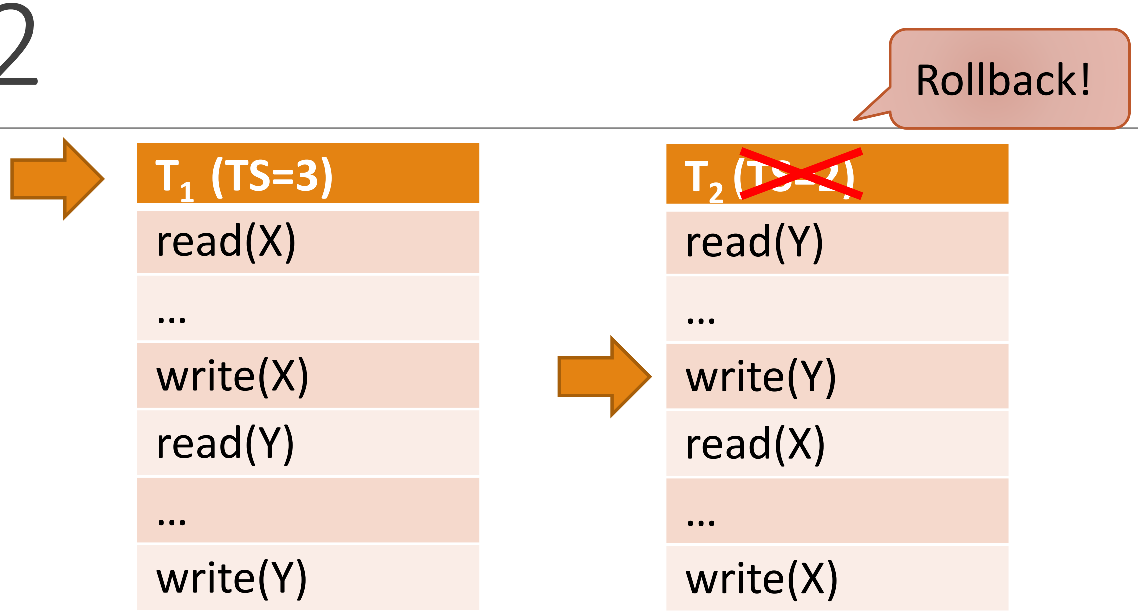
T_1 ~~(TS=1)~~	T_2 (TS=2)
read(X)	read(Y)
…	…
write(X)	write(Y)
read(Y)	read(X)
…	…
write(Y)	write(X)

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



Rollback!

T_1 (TS=3)
read(X)
…
write(X)
read(Y)
…
write(Y)

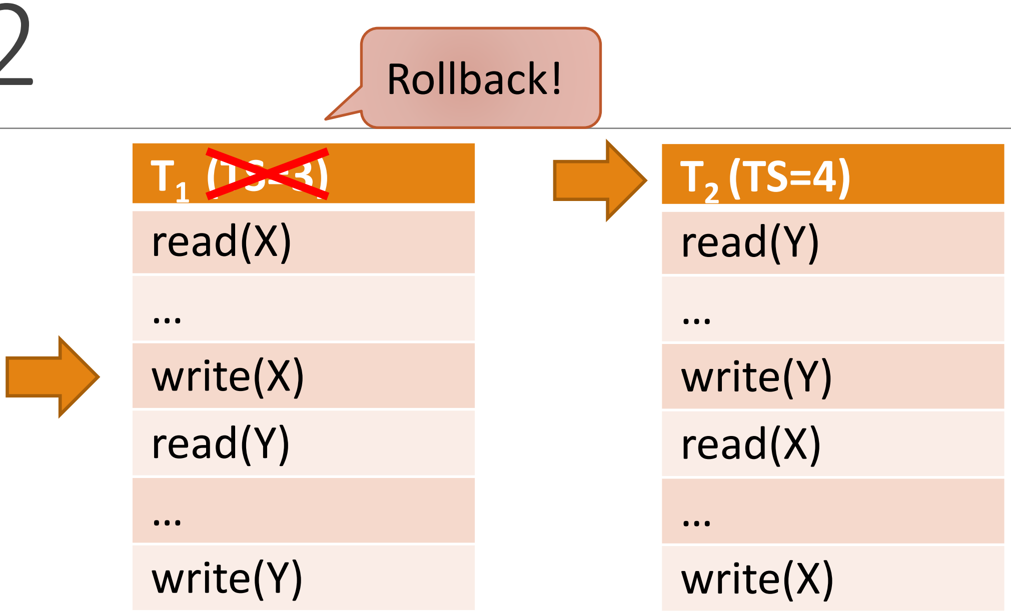
T_2 ~~(TS=2)~~
read(Y)
…
write(Y)
read(X)
…
write(X)

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



Rollback!

T_1 ~~(TS=3)~~	T_2 (TS=4)
read(X)	read(Y)
…	…
write(X)	write(Y)
read(Y)	read(X)
…	…
write(Y)	write(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?

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)

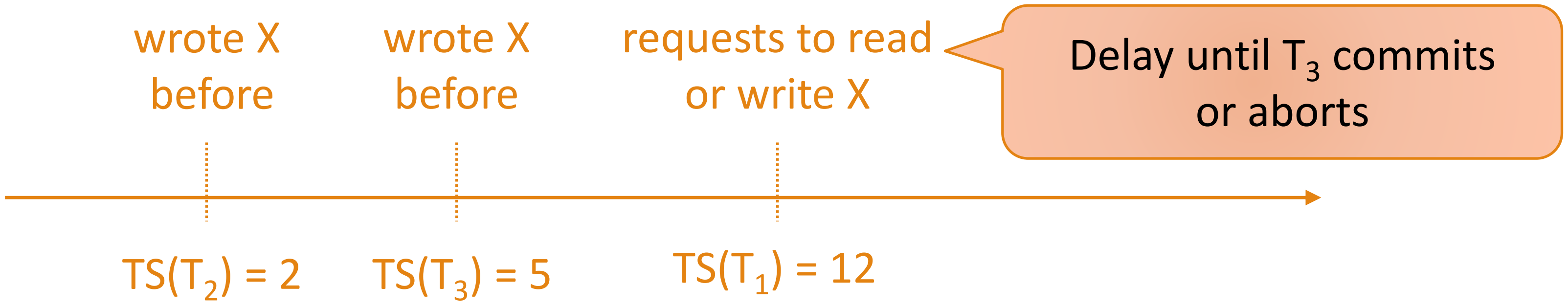
Starvation can be prevented using appropriate techniques (not in this module)

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.



wrote X before

wrote X before

requests to read or write X

Delay until T_3 commits or aborts

$TS(T_2) = 2$

$TS(T_3) = 5$

$TS(T_1) = 12$

Multiversion concurrency control

Many DBMS are implementing a variant of time-stamp based approaches, called multiversion concurrency control

The idea is the same as time-stamp based approaches to scheduling, but you have multiple versions of the database!

- Meaning that write operations do not overwrite each other, but instead $w_i(X)$ creates makes a new version of X at time $TS(T_i)$
- Whenever you read, you read the latest version before your timestamp

This means that a transaction only need to restart if it tries to write AND the read timestamp is later than your timestamp

Or, written out in full, the rules are:

- For writes: **Abort & restart T_1** if $RT(X) > TS(T_1)$ and **otherwise, grant request**
- For reads: **Always granted**

There is also a strict variant, where you delay reads until the transaction you read from commits

Example 1 - MVCC

			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						
12						
13						

Timestamp	**X**	**Y**
0	10	10

Example 1 - MVCC

			X		Y	
Time	**T$_1$** (TS = 1)	**T$_2$** (TS = 2)	**RT**	**WT**	**RT**	**WT**
0			0	0	0	0
1	read(X)		**1**	0	0	0
2						
3						
4						
5						
6						
7						
8						
9						
10						
11						
12						
13						

Timestamp	**X**	**Y**
0	10	10

Example 1 - MVCC

			X		Y	
Time	**T_1** (TS = 1)	**T_2** (TS = 2)	**RT**	**WT**	**RT**	**WT**
0			0	0	0	0
1	read(X)		**1**	0	0	0
2	X := X + 100		1	0	0	0
3						
4						
5						
6						
7						
8						
9						
10						
11						
12						
13						

Timestamp	**X**	**Y**
0	10	10

Example 1 - MVCC

			X		Y	
Time	**T$_1$** (TS = 1)	**T$_2$** (TS = 2)	**RT**	**WT**	**RT**	**WT**
0			0	0	0	0
1	read(X)		**1**	0	0	0
2	X := X + 100		1	0	0	0
3		read(Y)	1	0	**2**	0
4						
5						
6						
7						
8						
9						
10						
11						
12						
13						

Timestamp	**X**	**Y**
0	10	10

Example 1 - MVCC

			X		Y	
Time	**T_1** (TS = 1)	**T_2** (TS = 2)	**RT**	**WT**	**RT**	**WT**
0			0	0	0	0
1	read(X)		**1**	0	0	0
2	X := X + 100		1	0	0	0
3		read(Y)	1	0	**2**	0
4		Y := Y * 2	1	0	2	0
5						
6						
7						
8						
9						
10						
11						
12						
13						

Timestamp	**X**	**Y**
0	10	10

Example 1 - MVCC

			X		Y	
Time	**T_1** (TS = 1)	**T_2** (TS = 2)	**RT**	**WT**	**RT**	**WT**
0			0	0	0	0
1	read(X)		**1**	0	0	0
2	X := X + 100		1	0	0	0
3		read(Y)	1	0	**2**	0
4		Y := Y * 2	1	0	2	0
5		write(Y)	1	0	2	**2**
6						
7						
8						
9						
10						
11						
12						
13						

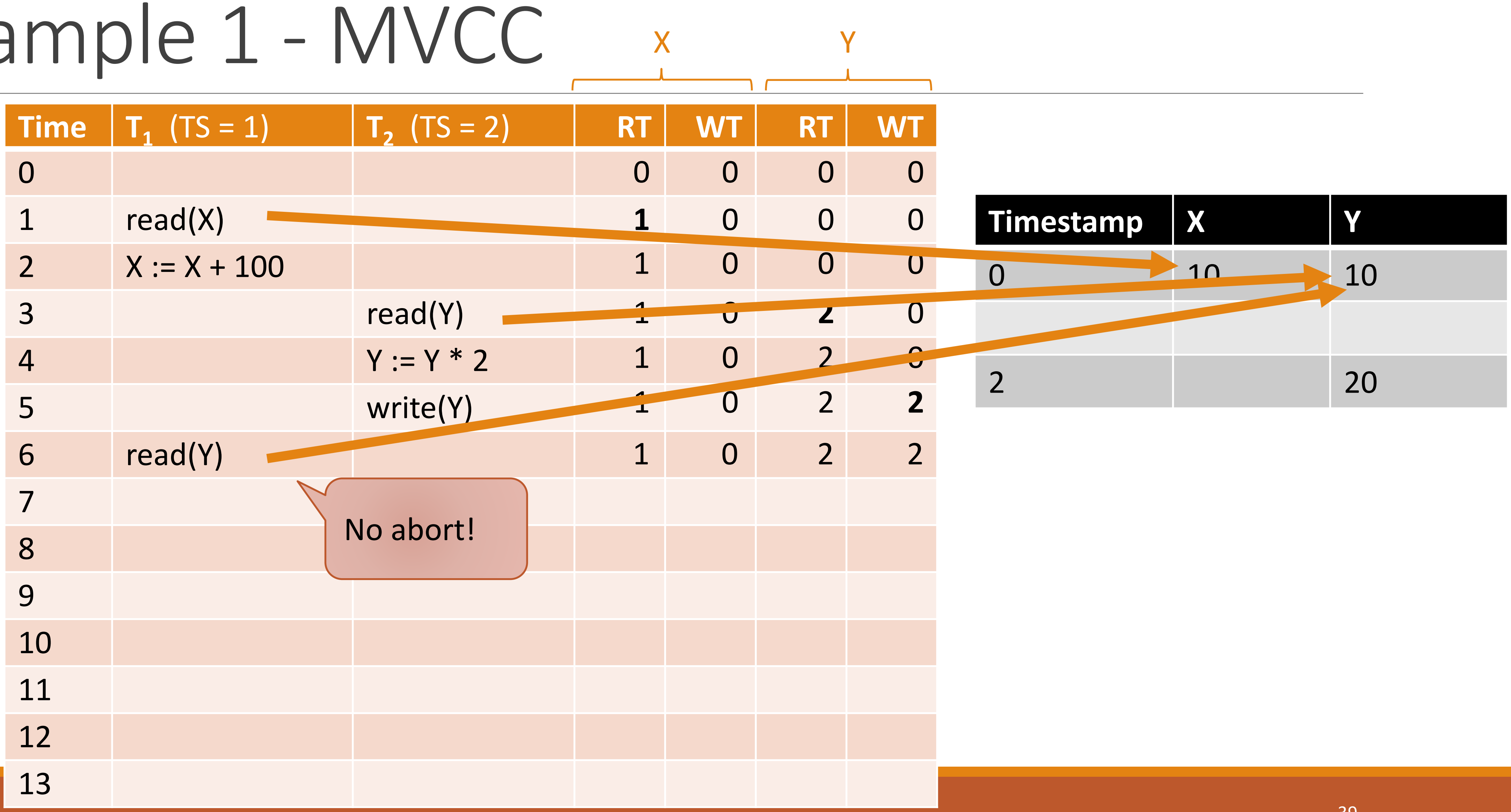
Timestamp	**X**	**Y**
0	10	10

Example 1 - MVCC

			X		Y	
Time	**T_1 (TS = 1)**	**T_2 (TS = 2)**	**RT**	**WT**	**RT**	**WT**
0			0	0	0	0
1	read(X)		**1**	0	0	0
2	X := X + 100		1	0	0	0
3		read(Y)	1	0	**2**	0
4		Y := Y * 2	1	0	2	0
5		write(Y)	1	0	2	**2**
6						
7						
8						
9						
10						
11						
12						
13						

Timestamp	**X**	**Y**
0	10	10
2		20

Example 1 - MVCC

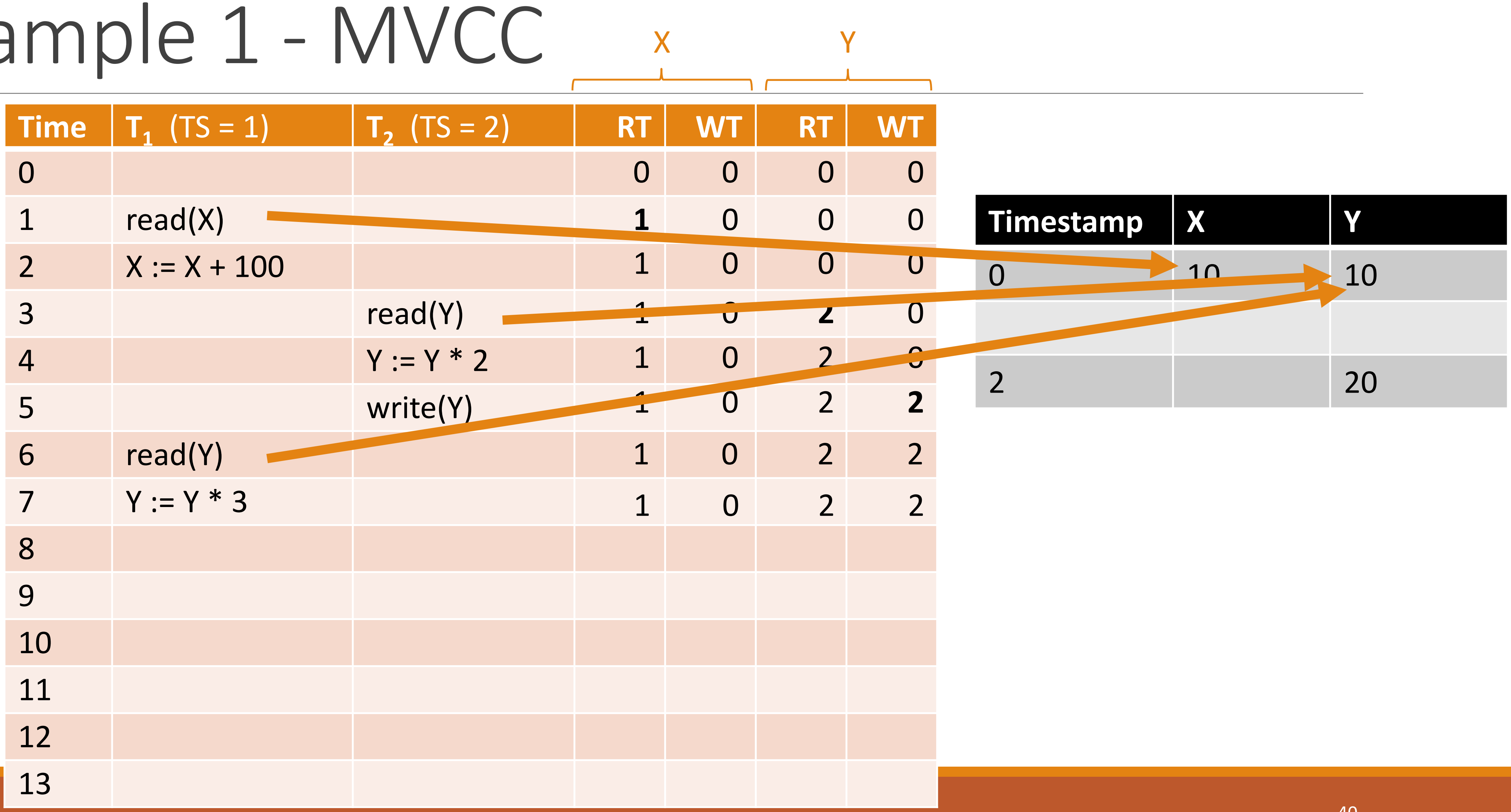


			X		Y	
Time	**T_1 (TS = 1)**	**T_2 (TS = 2)**	**RT**	**WT**	**RT**	**WT**
0			0	0	0	0
1	read(X)		**1**	0	0	0
2	X := X + 100		1	0	0	0
3		read(Y)	1	0	**2**	0
4		Y := Y * 2	1	0	2	0
5		write(Y)	1	0	2	**2**
6	read(Y)		1	0	2	2
7						
8						
9						
10						
11						
12						
13						

No abort!

Timestamp	**X**	**Y**
0	10	10
2		20

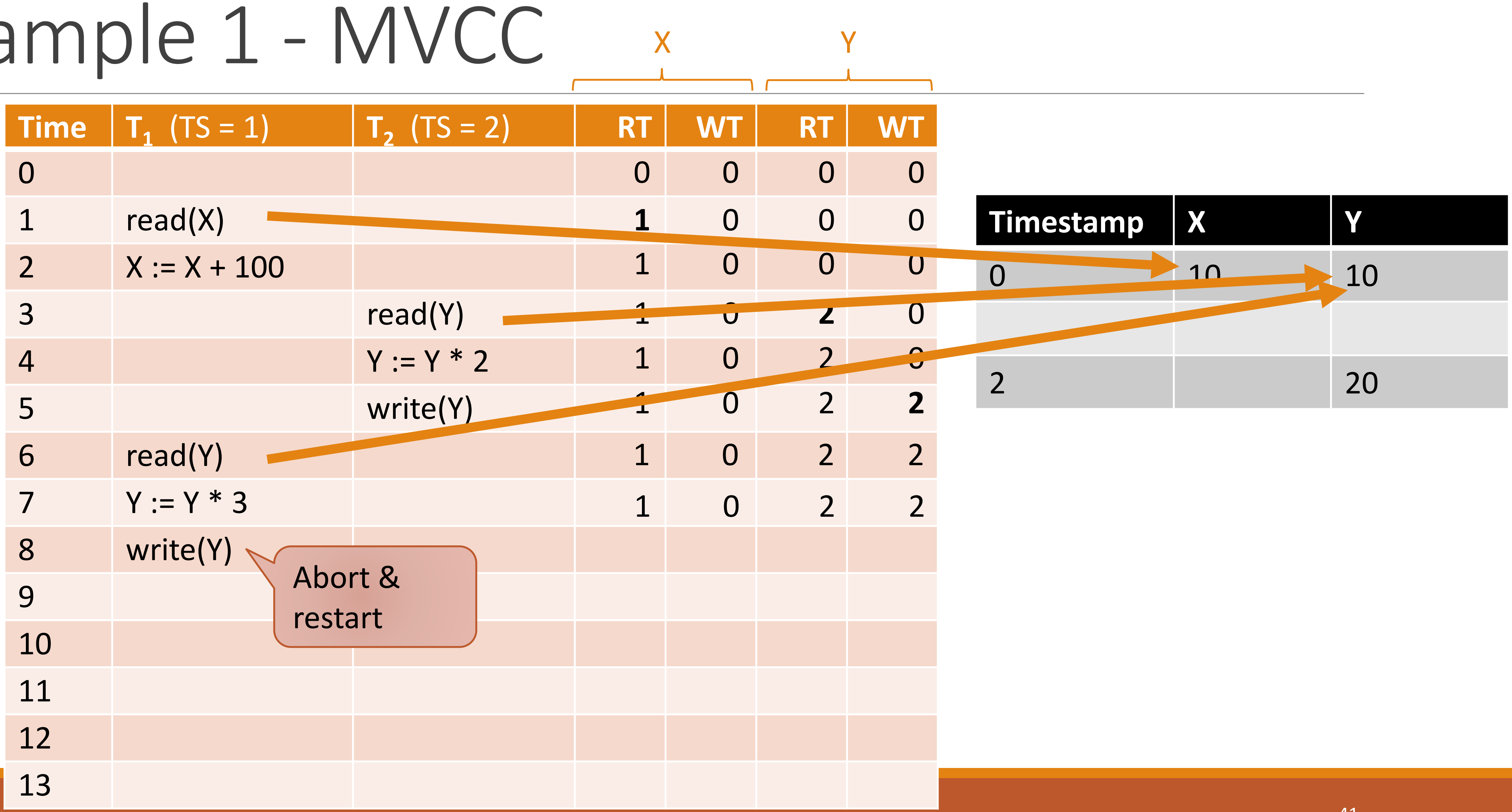
Example 1 - MVCC



			X		Y	
Time	**T$_1$** (TS = 1)	**T$_2$** (TS = 2)	**RT**	**WT**	**RT**	**WT**
0			0	0	0	0
1	read(X)		**1**	0	0	0
2	X := X + 100		1	0	0	0
3		read(Y)	1	0	**2**	0
4		Y := Y * 2	1	0	2	0
5		write(Y)	1	0	2	**2**
6	read(Y)		1	0	2	2
7	Y := Y * 3		1	0	2	2
8						
9						
10						
11						
12						
13						

Timestamp	**X**	**Y**
0	10	10
2		20

Example 1 - MVCC



			X		Y	
Time	**T_1** (TS = 1)	**T_2** (TS = 2)	**RT**	**WT**	**RT**	**WT**
0			0	0	0	0
1	read(X)		**1**	0	0	0
2	X := X + 100		1	0	0	0
3		read(Y)	1	0	**2**	0
4		Y := Y * 2	1	0	2	0
5		write(Y)	1	0	2	**2**
6	read(Y)		1	0	2	2
7	Y := Y * 3		1	0	2	2
8	write(Y)					
9						
10						
11						
12						
13						

Abort & restart

Timestamp	**X**	**Y**
0	10	10
2		20

Exam MVCC

New timestamp

			X		Y	
Time	**T_1 (TS = 3)**	**T_2 (TS = 2)**	**RT**	**WT**	**RT**	**WT**
0			0	0	0	0
1	read(X)		**1**	0	0	0
2	X := X + 100		1	0	0	0
3		read(Y)	1	0	**2**	0
4		Y := Y * 2	1	0	2	0
5		write(Y)	1	0	2	**2**
6	read(Y)		1	0	2	2
7	Y := Y * 3		1	0	2	2
8	~~write(Y)~~		0	0	2	2
9						
10						
11						
12						
13						

Timestamp	**X**	**Y**
0	10	10
2		20

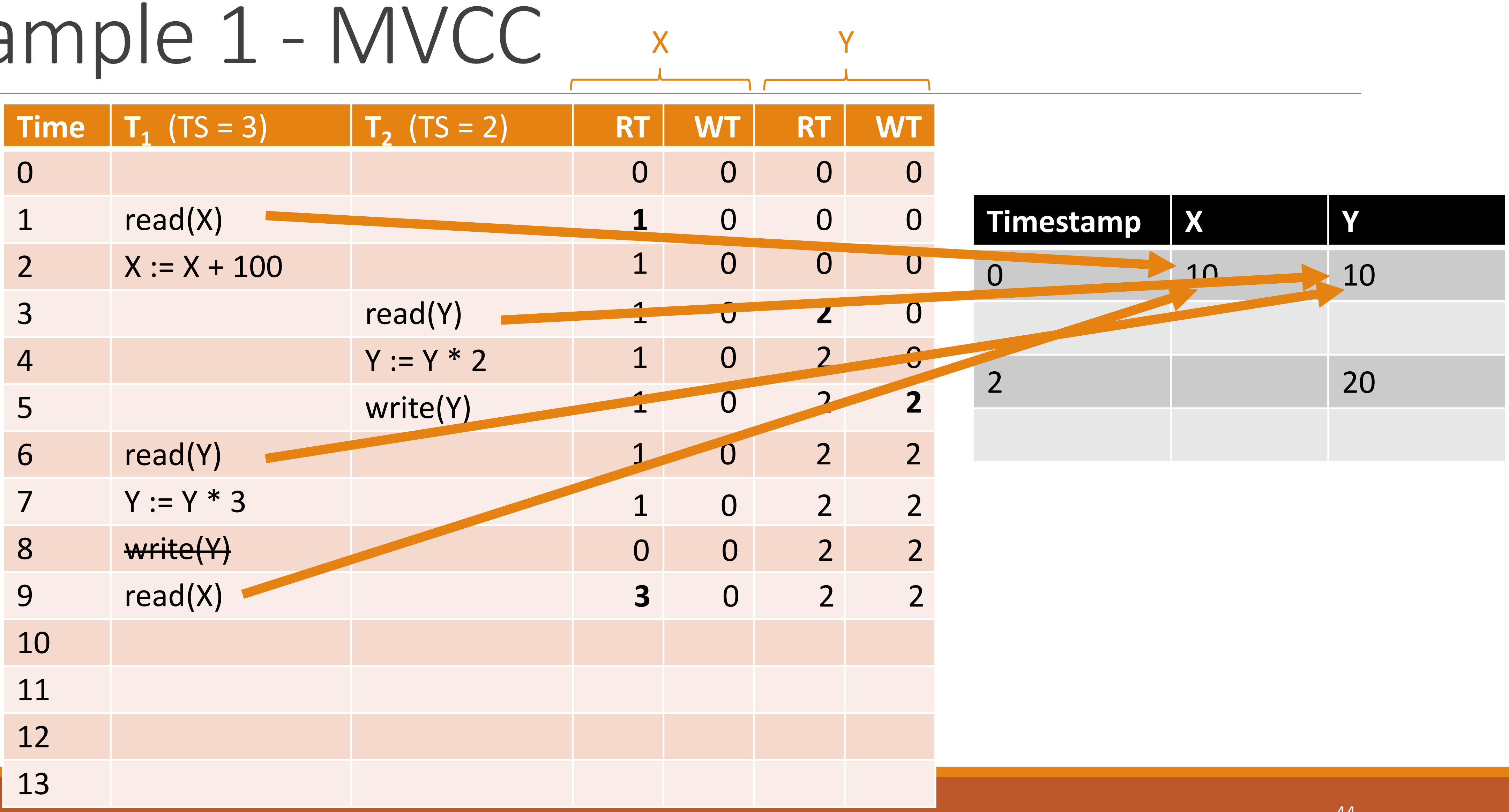
Exam MVCC

New timestamp

Time	T_1 (TS = 3)	T_2 (TS = 2)	X		Y	
			RT	WT	RT	WT
0			0	0	0	0
1	read(X)		**1**	0	0	0
2	X := X + 100		1	0	0	0
3		read(Y)	1	0	**2**	0
4		Y := Y * 2	1	0	2	0
5		write(Y)	1	0	2	**2**
6	read(Y)		1	0	2	2
7	Y := Y * 3		1	0	2	2
8	~~write(Y)~~		**0**	0	2	2
9						
10						
11						
12						
13						

Timestamp	X	Y
0	10	10
2		20

Example 1 - MVCC



			X		Y	
Time	**T$_1$** (TS = 3)	**T$_2$** (TS = 2)	**RT**	**WT**	**RT**	**WT**
0			0	0	0	0
1	read(X)		**1**	0	0	0
2	X := X + 100		1	0	0	0
3		read(Y)	1	0	**2**	0
4		Y := Y * 2	1	0	2	0
5		write(Y)	1	0	2	**2**
6	read(Y)		1	0	2	2
7	Y := Y * 3		1	0	2	2
8	~~write(Y)~~		0	0	2	2
9	read(X)		**3**	0	2	2
10						
11						
12						
13						

Timestamp	**X**	**Y**
0	10	10
2		20

Example 1 - MVCC

			X		Y	
Time	**T_1 (TS = 3)**	**T_2 (TS = 2)**	**RT**	**WT**	**RT**	**WT**
0			0	0	0	0
1	read(X)		**1**	0	0	0
2	X := X + 100		1	0	0	0
3		read(Y)	1	0	**2**	0
4		Y := Y * 2	1	0	2	0
5		write(Y)	1	0	2	**2**
6	read(Y)		1	0	2	2
7	Y := Y * 3		1	0	2	2
8	~~write(Y)~~		0	0	2	2
9	read(X)		**3**	0	2	2
10	X := X + 100		3	0	2	2
11						
12						
13						

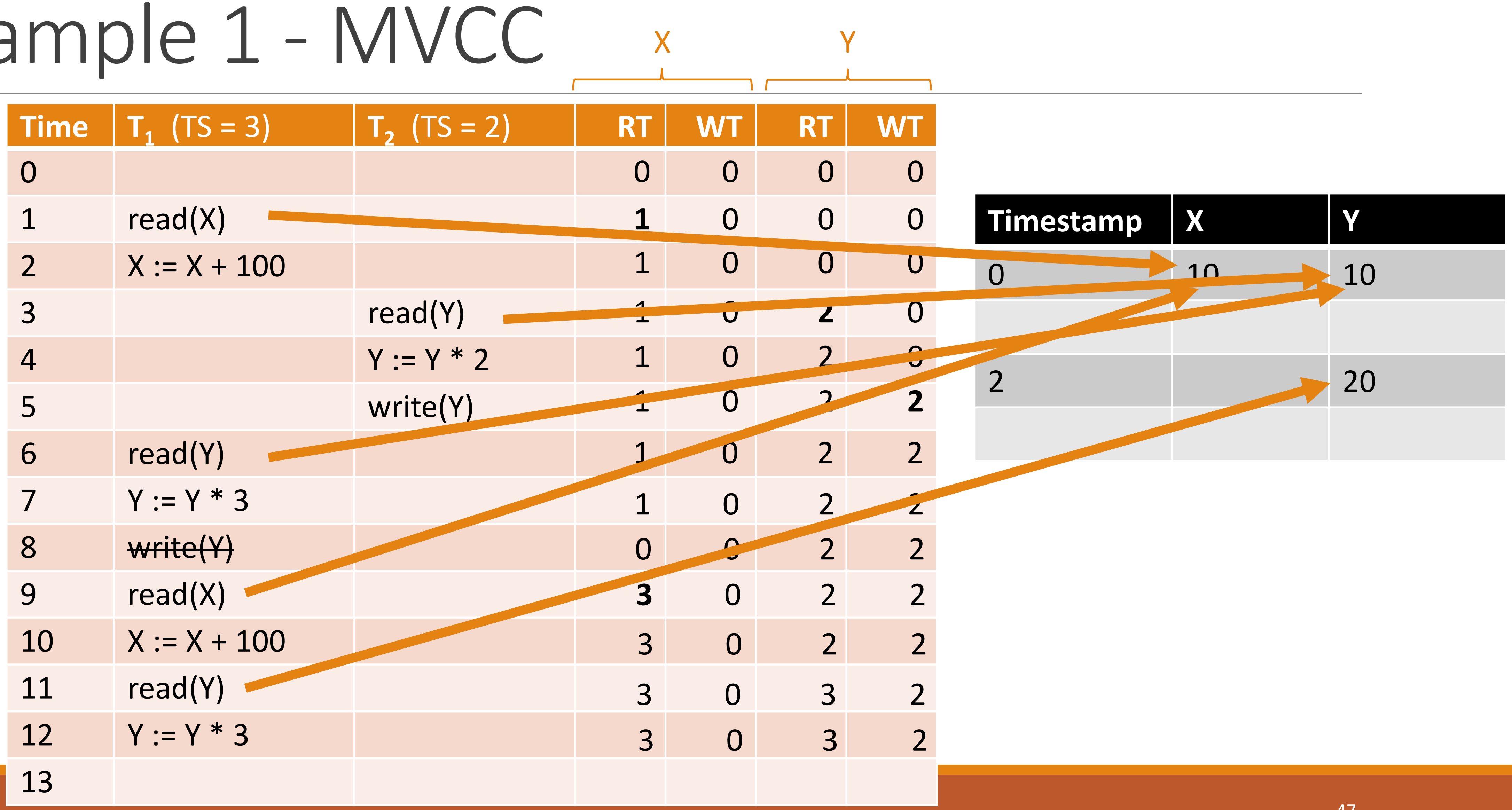
Timestamp	X	Y
0	10	10
2		20

Example 1 - MVCC

			X		Y	
Time	**T_1 (TS = 3)**	**T_2 (TS = 2)**	**RT**	**WT**	**RT**	**WT**
0			0	0	0	0
1	read(X)		**1**	0	0	0
2	X := X + 100		1	0	0	0
3		read(Y)	1	0	**2**	0
4		Y := Y * 2	1	0	2	0
5		write(Y)	1	0	2	**2**
6	read(Y)		1	0	2	2
7	Y := Y * 3		1	0	2	2
8	~~write(Y)~~		0	0	2	2
9	read(X)		**3**	0	2	2
10	X := X + 100		3	0	2	2
11	read(Y)		3	0	3	2
12						
13						

Timestamp	**X**	**Y**
0	10	10
2		20

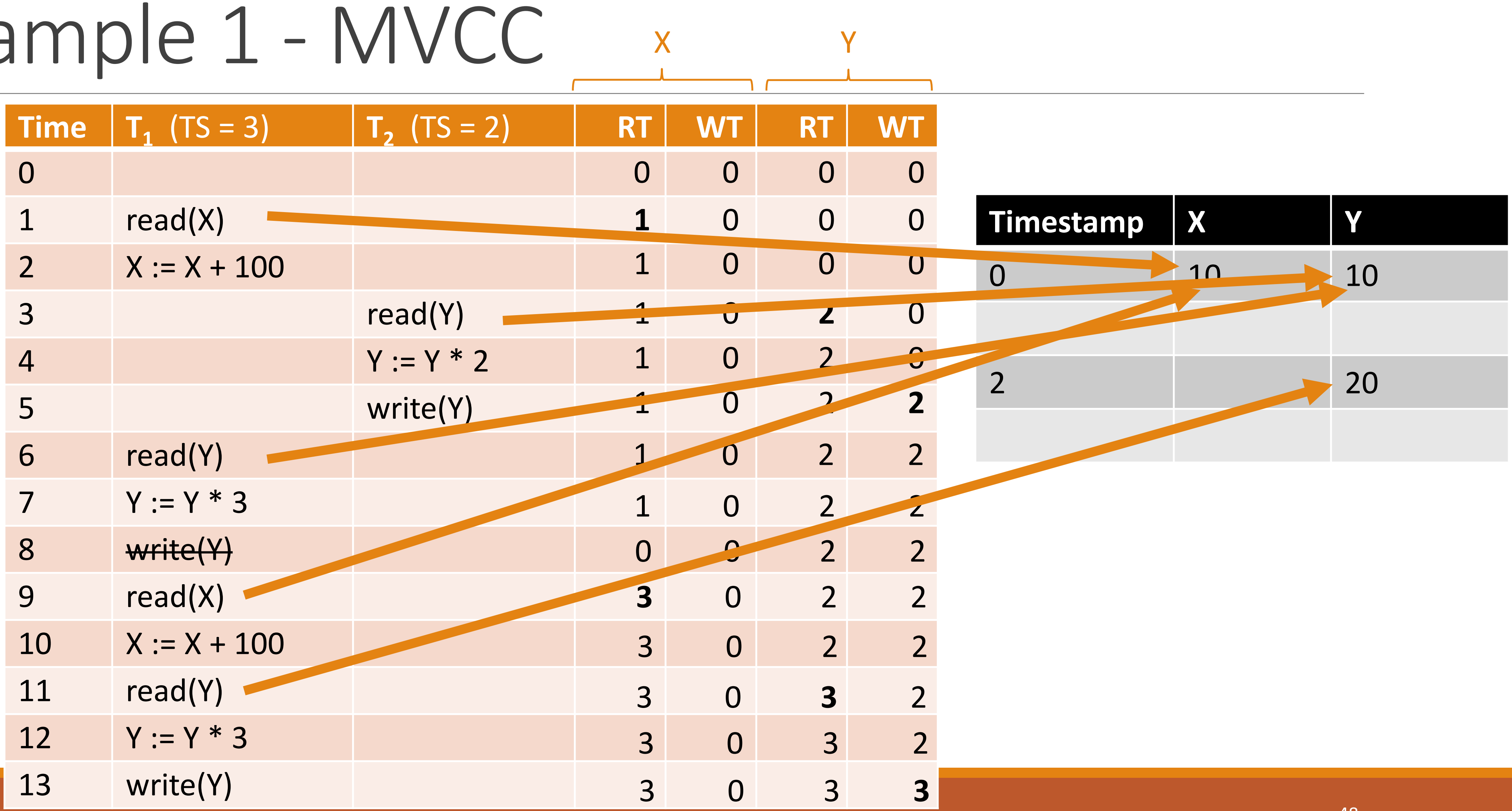
Example 1 - MVCC



			X		Y	
Time	**T_1 (TS = 3)**	**T_2 (TS = 2)**	**RT**	**WT**	**RT**	**WT**
0			0	0	0	0
1	read(X)		**1**	0	0	0
2	X := X + 100		1	0	0	0
3		read(Y)	1	0	**2**	0
4		Y := Y * 2	1	0	2	0
5		write(Y)	1	0	2	**2**
6	read(Y)		1	0	2	2
7	Y := Y * 3		1	0	2	2
8	~~write(Y)~~		0	0	2	2
9	read(X)		**3**	0	2	2
10	X := X + 100		3	0	2	2
11	read(Y)		3	0	3	2
12	Y := Y * 3		3	0	3	2
13						

Timestamp	**X**	**Y**
0	10	10
2		20

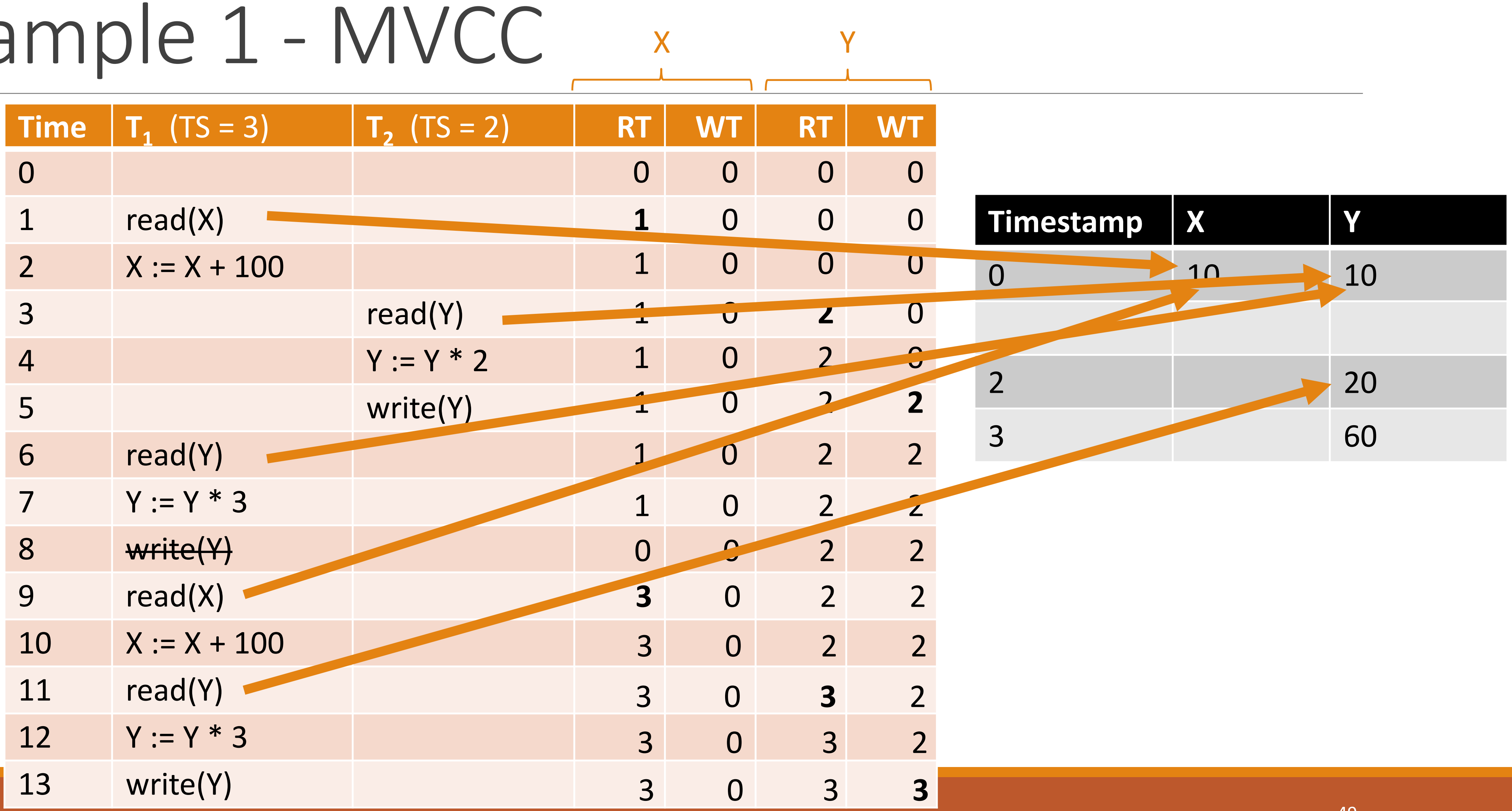
Example 1 - MVCC



			X		Y	
Time	T_1 (TS = 3)	T_2 (TS = 2)	RT	WT	RT	WT
0			0	0	0	0
1	read(X)		**1**	0	0	0
2	X := X + 100		1	0	0	0
3		read(Y)	1	0	**2**	0
4		Y := Y * 2	1	0	2	0
5		write(Y)	1	0	2	**2**
6	read(Y)		1	0	2	2
7	Y := Y * 3		1	0	2	2
8	~~write(Y)~~		0	0	2	2
9	read(X)		**3**	0	2	2
10	X := X + 100		3	0	2	2
11	read(Y)		3	0	**3**	2
12	Y := Y * 3		3	0	3	2
13	write(Y)		3	0	3	**3**

Timestamp	X	Y
0	10	10
2		20

Example 1 - MVCC



			X		Y	
Time	T_1 (TS = 3)	T_2 (TS = 2)	RT	WT	RT	WT
0			0	0	0	0
1	read(X)		**1**	0	0	0
2	X := X + 100		1	0	0	0
3		read(Y)	1	0	**2**	0
4		Y := Y * 2	1	0	2	0
5		write(Y)	1	0	2	**2**
6	read(Y)		1	0	2	2
7	Y := Y * 3		1	0	2	2
8	~~write(Y)~~		0	0	2	2
9	read(X)		**3**	0	2	2
10	X := X + 100		3	0	2	2
11	read(Y)		3	0	**3**	2
12	Y := Y * 3		3	0	3	2
13	write(Y)		3	0	3	**3**

Timestamp	X	Y
0	10	10
2		20
3		60

Summary

One can use an alternate technique (instead of Strict 2PL) to ensure all the ACID properties and avoid deadlocks automatically: The idea is to just act as if the transaction was executed at its start time and restart any transaction that attempts to create a paradox with that!

- This only gives you conflict-serializability, but strict can be ensured by, on each read or write request (that would not be a paradox), wait until the transaction that last wrote to that item has committed or aborted

One can do the idea in two ways:

- "Normal" database with 1 version of each variable
- Multiversion concurrency control, where each write is saved with its own timestamp!