CS186 Discussion #9

(More Concurrency)

Michelle Nguyen

Midterm Review Session

Saturday, 11/7, 1-3pm in 100 GPB

Lock Granularity

sid	points	grade
Bob	43	C
Joe	99	C
Alice	87	C
Suzy	50	C
Ted	73	C
Tim	12	C

T1: `UPDATE students SET grade='A' WHERE points >= 70;`

T2: `UPDATE students SET grade='F' WHERE points < 70;`

Lock Granularity

T1(X)

sid	points	grade
Bob	43	C
Joe	99	C
Alice	87	C
Suzy	50	C
Ted	73	C
Tim	12	C

T1: `UPDATE students SET grade='A' WHERE points >= 70;`

T2: `UPDATE students SET grade='F' WHERE points < 70;`

Lock Granularity

T1(X)

sid	points	grade
Bob	43	C
Joe	99	A
Alice	87	A
Suzy	50	C
Ted	73	A
Tim	12	C

T1: `UPDATE students SET grade='A' WHERE points >= 70;`

T2: `UPDATE students SET grade='F' WHERE points < 70;`

Lock Granularity

T2(X)

sid	points	grade
Bob	43	C
Joe	99	A
Alice	87	A
Suzy	50	C
Ted	73	A
Tim	12	C

T1: `UPDATE students SET grade='A' WHERE points >= 70;`

T2: `UPDATE students SET grade='F' WHERE points < 70;`

Lock Granularity

T2(X)

sid	points	grade
Bob	43	F
Joe	99	A
Alice	87	A
Suzy	50	F
Ted	73	A
Tim	12	F

T1: `UPDATE students SET grade='A' WHERE points >= 70;`

T2: `UPDATE students SET grade='F' WHERE points < 70;`

Lock Hierarchy

- Each Xact starts at root of hierarchy
- Gets locks in top-down order
- Releases locks in bottom-up order

$$\begin{tikzcd}
\text{Database} \arrow[d] \\
\text{Tables} \arrow[d] \\
\text{Pages} \arrow[d] \\
\text{Tuples}
\end{tikzcd}$$

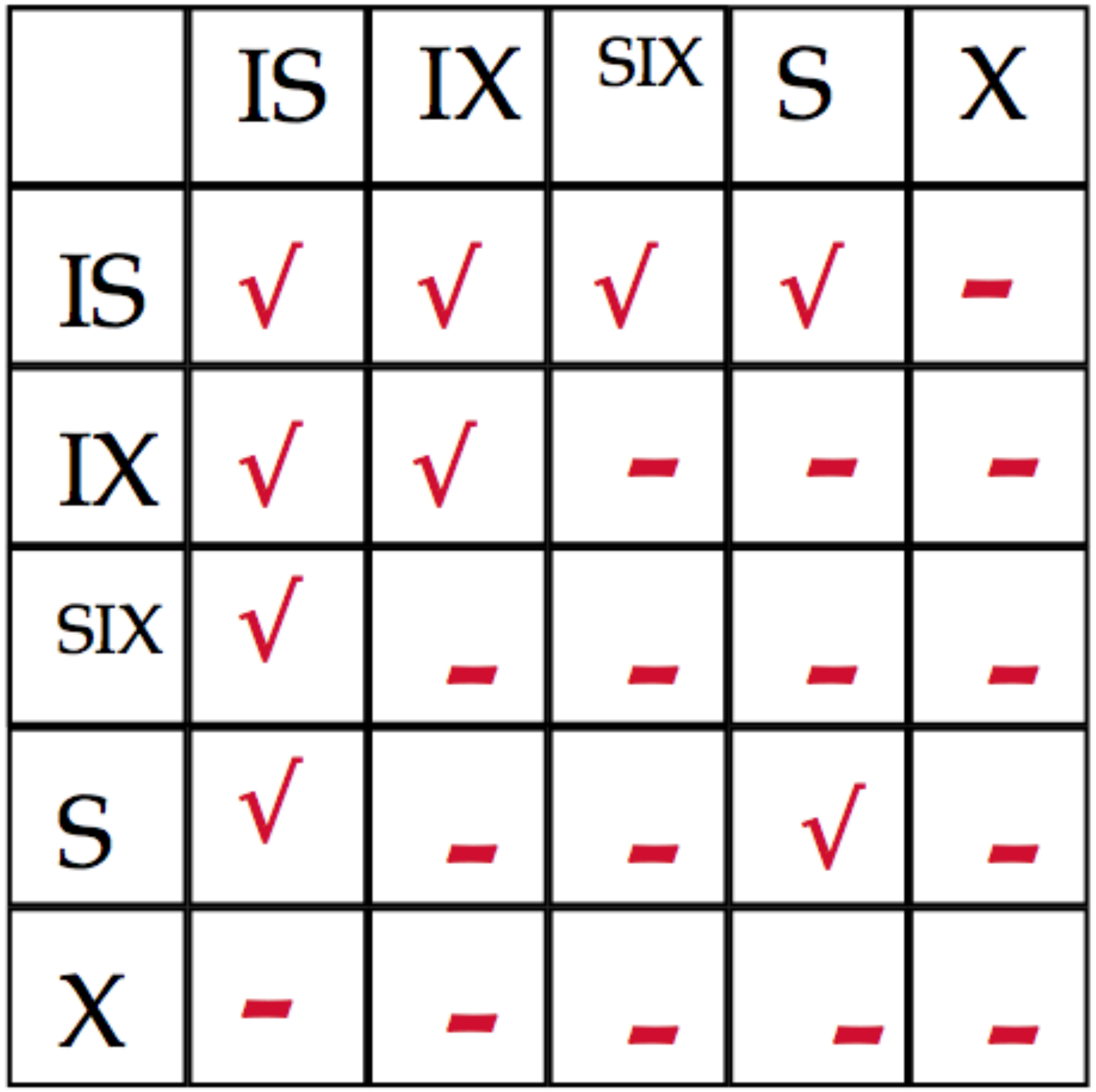
Locks

- IS: intent to get S lock(s) at finer granularity
- IX: intent to get X lock(s) at finer granularity
- SIX: shared lock, with intent to get X lock(s) at finer granularity

Lock Compatibility Matrix

	IS	IX	SIX	S	X
IS					
IX					
SIX					
S				√	–
X				–	–

Lock Compatibility Matrix



	IS	IX	SIX	S	X
IS	√	√	√	√	–
IX	√	√	–	–	–
SIX	√	–	–	–	–
S	√	–	–	√	–
X	–	–	–	–	–

Lock Granularity

T1(IX)

T2(IX)

sid	points	grade	
Bob	43	F	T2(X)
Joe	99	A	T1(X)
Alice	87	A	T1(X)
Suzy	50	F	T2(X)
Ted	73	A	T1(X)
Tim	12	F	T2(X)

Worksheet - Lock Granularity

Suppose a transaction, T1, wants to scan a table R and update a few of its tuples. What kind of locks should T1 have on R, its pages, and the tuples that are updated?

Suppose a transaction, T1, wants to scan a table R and update a few of its tuples. What kind of locks should T1 have on R, its pages, and the tuples that are updated?

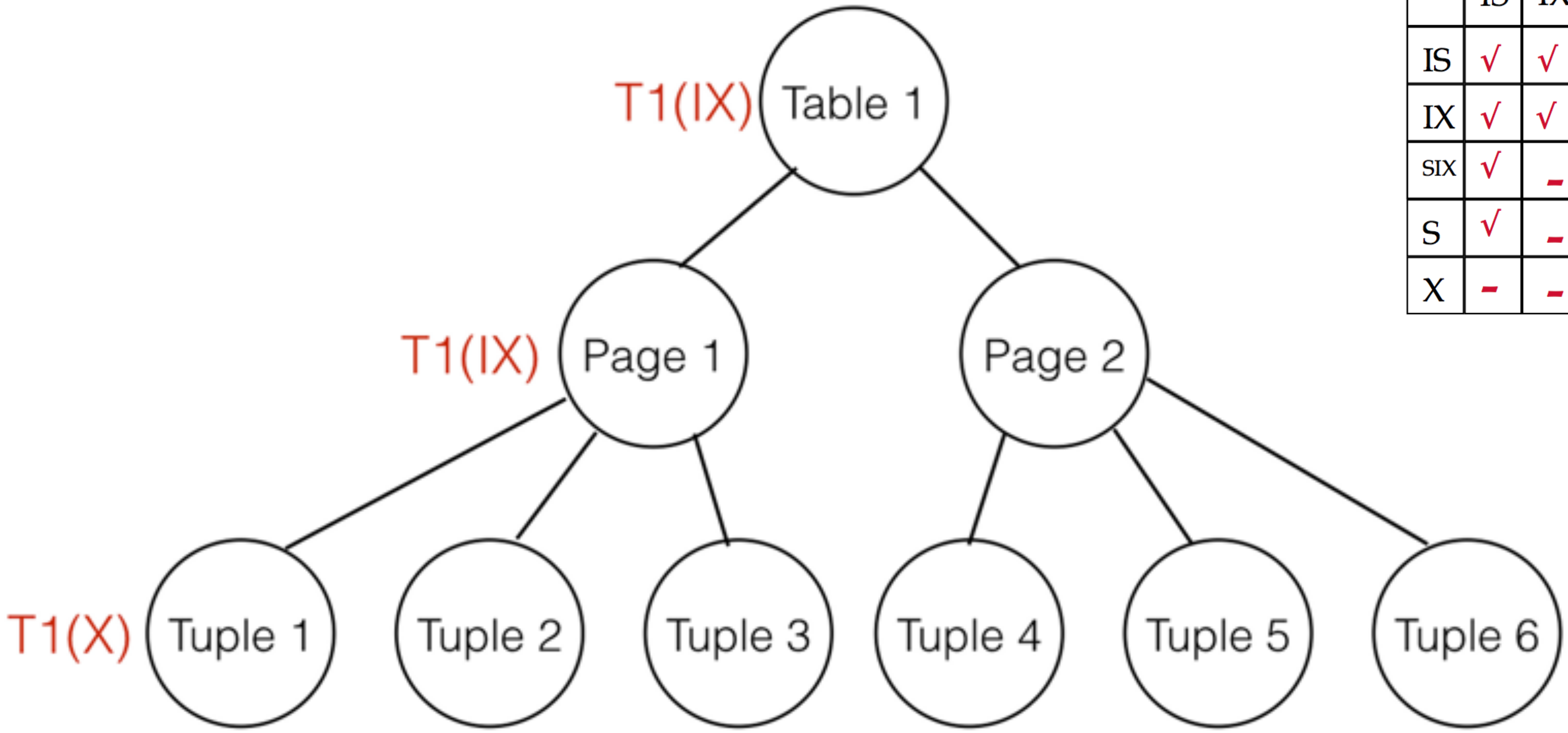
- SIX lock on R
- SIX or IX lock on pages
- X lock on updated tuples
- S lock on other tuples

Is an S lock compatible with an IX lock? Explain why or why not. Make your description as simple as possible.

Is an S lock compatible with an IX lock? Explain why or why not. Make your description as simple as possible.

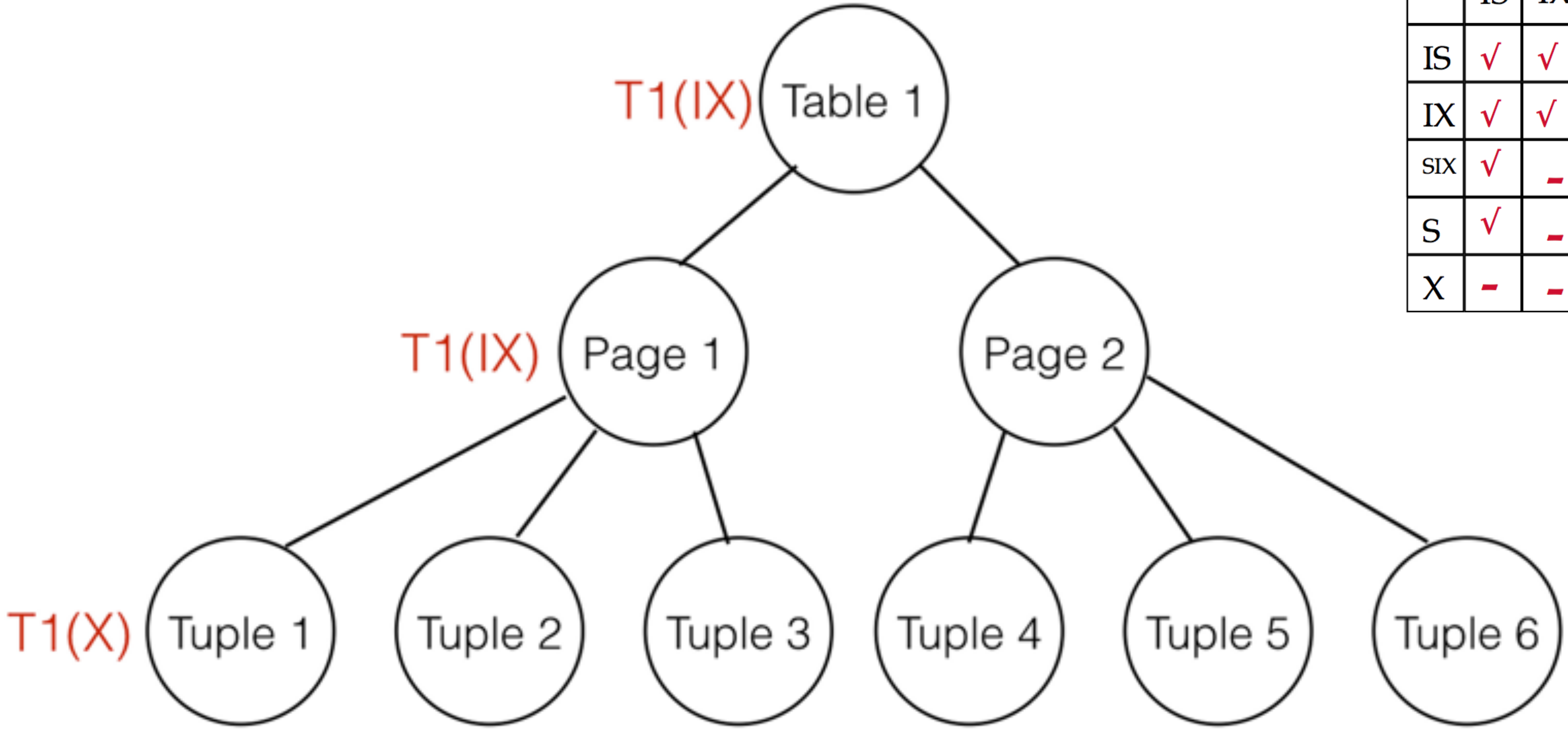
- Incompatible:
  - T1 has S lock on Students table to calculate average grade
  - T2 wants IX lock to change some grades

Which locks could be granted to transaction T2 for tuple 2?



	IS	IX	SIX	S	X
IS	√	√	√	√	-
IX	√	√	-	-	-
SIX	√	-	-	-	-
S	√	-	-	√	-
X	-	-	-	-	-

Which locks could be granted to transaction T2 for tuple 2?

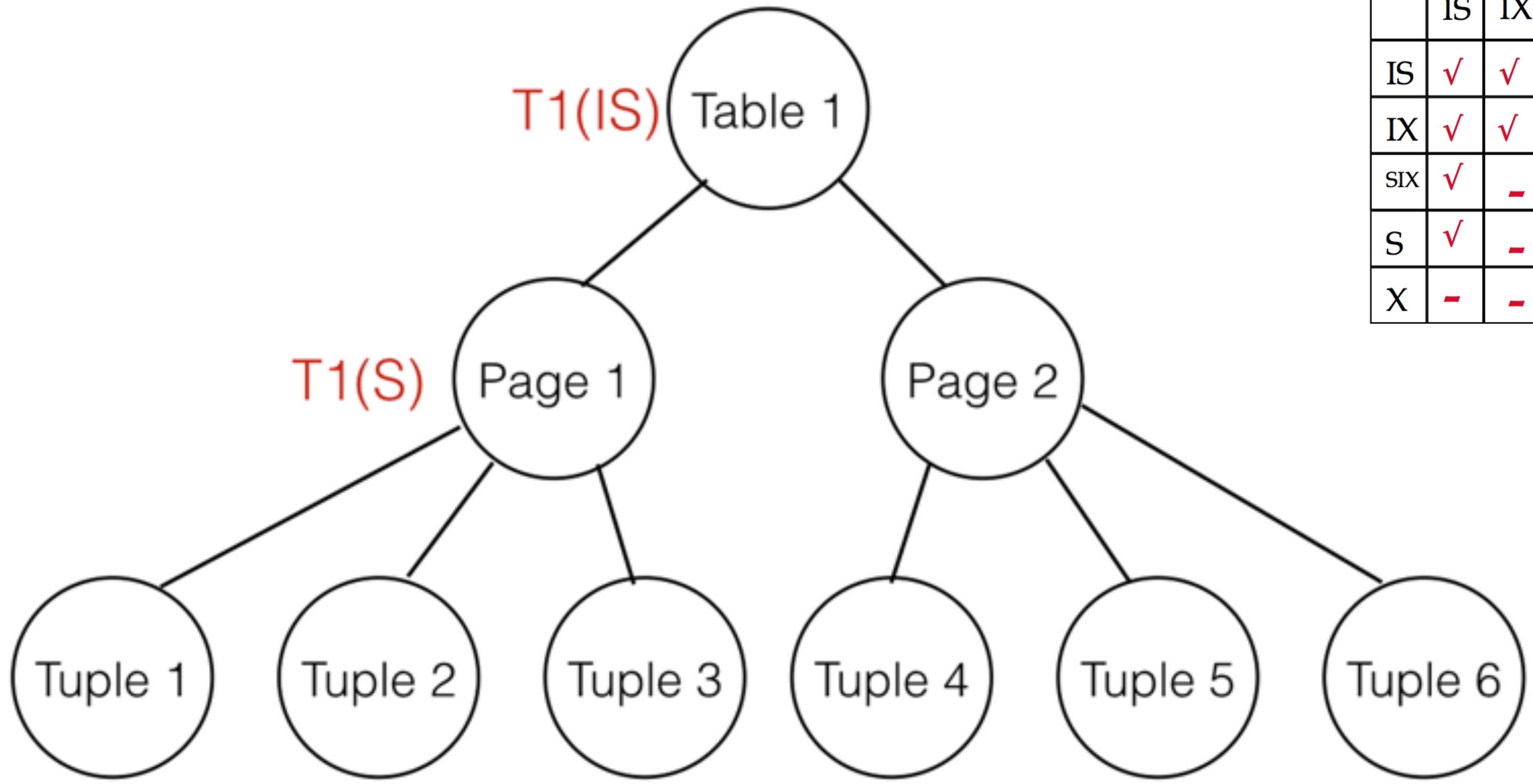


	IS	IX	SIX	S	X
IS	√	√	√	√	-
IX	√	√	-	-	-
SIX	√	-	-	-	-
S	√	-	-	√	-
X	-	-	-	-	-

X, S

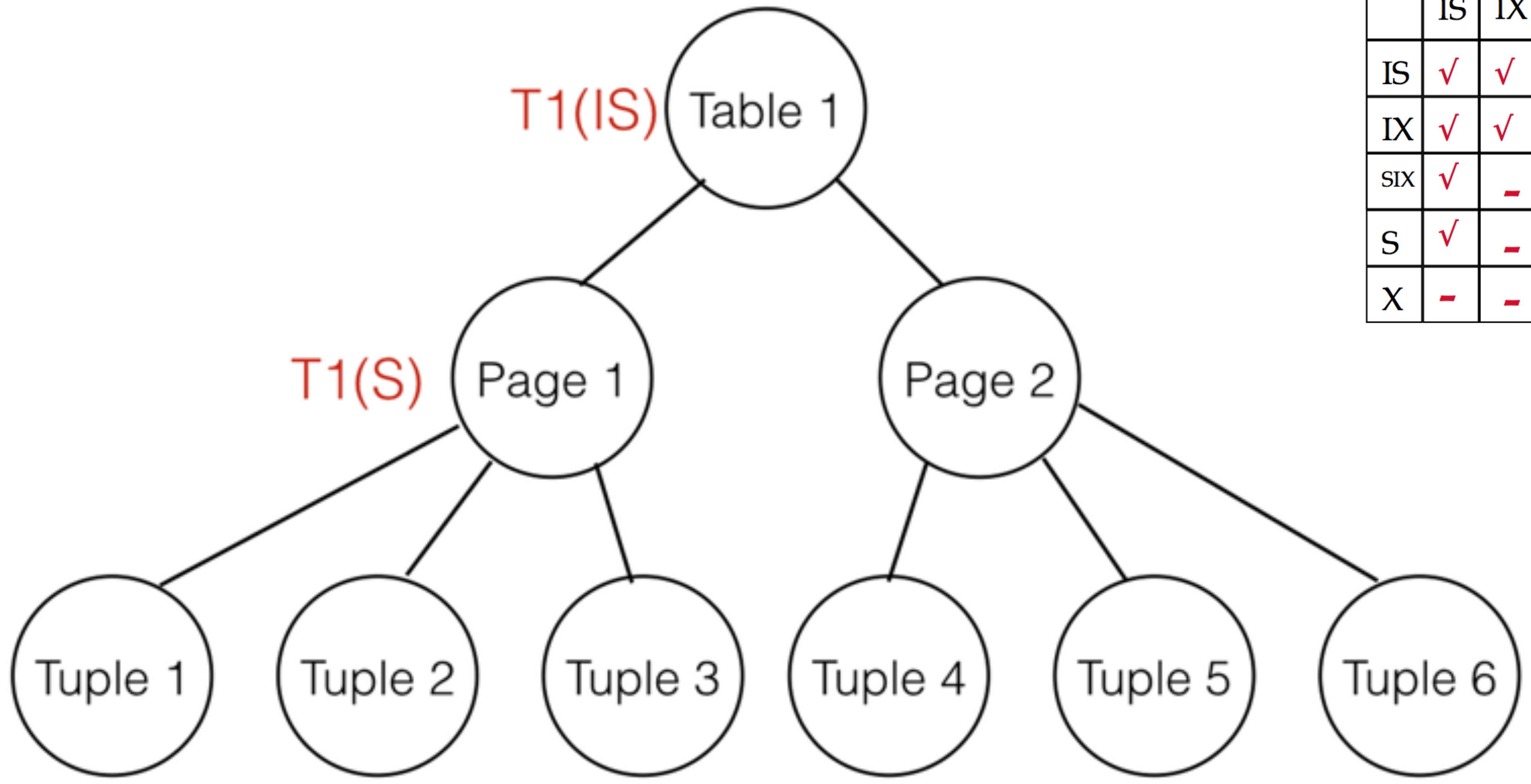
Can get IX, IS, but does not make sense at tuple-level.

What locks could be granted to T2 for page 1?



	IS	IX	SIX	S	X
IS	√	√	√	√	-
IX	√	√	-	-	-
SIX	√	-	-	-	-
S	√	-	-	√	-
X	-	-	-	-	-

What locks could be granted to T2 for page 1?



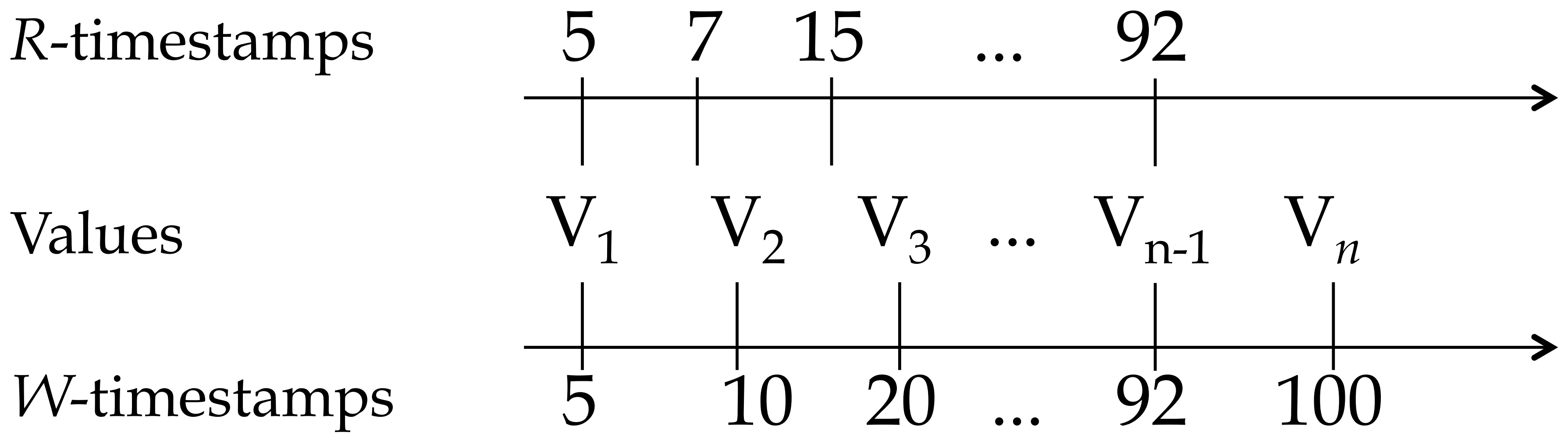
	IS	IX	SIX	S	X
IS	√	√	√	√	–
IX	√	√	–	–	–
SIX	√	–	–	–	–
S	√	–	–	√	–
X	–	–	–	–	–

S, IS

Multiversion Concurrency Control

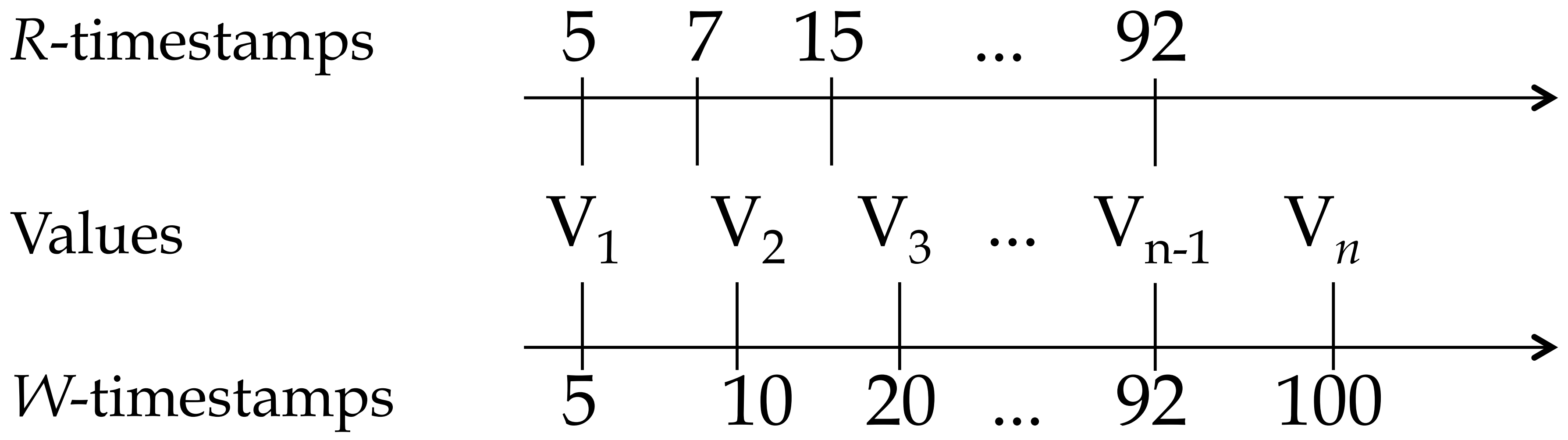
- Alternative to 2PL
  - Less waiting, but more aborts
- Timestamp Ordered MVCC:
  - Each transaction gets timestamp upon entry
  - Keep timeline of read timestamps and versions

TO-MVCC



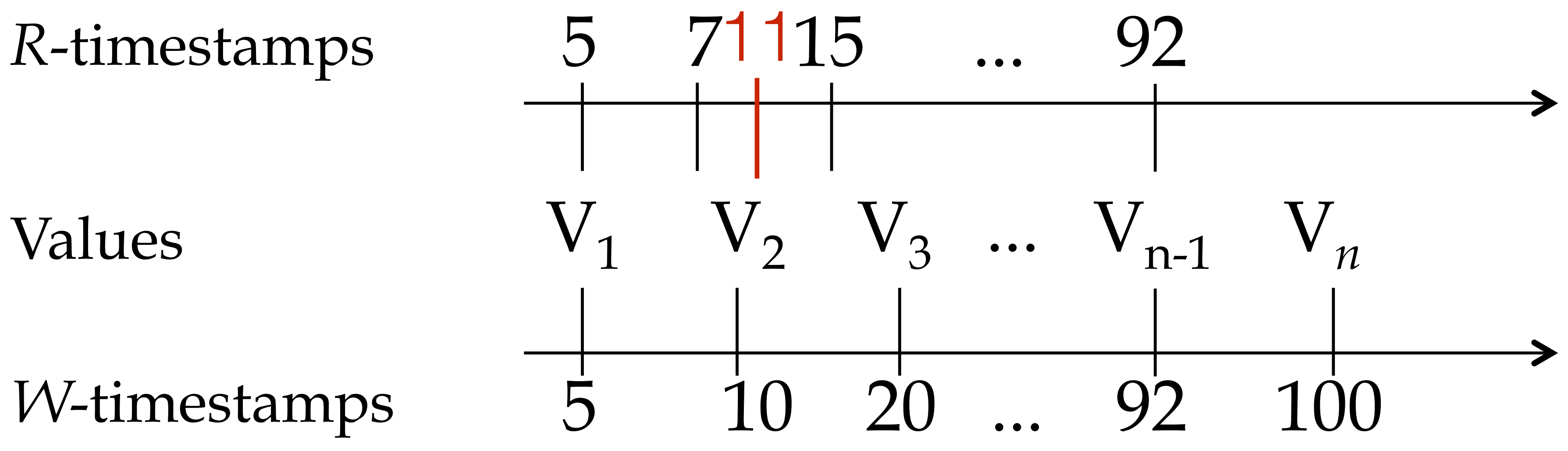
R-timestamps
5
7
15
...
92
Values
V_1
V_2
V_3
...
V_{n-1}
V_n
W-timestamps
5
10
20
...
92
100

TO-MVCC



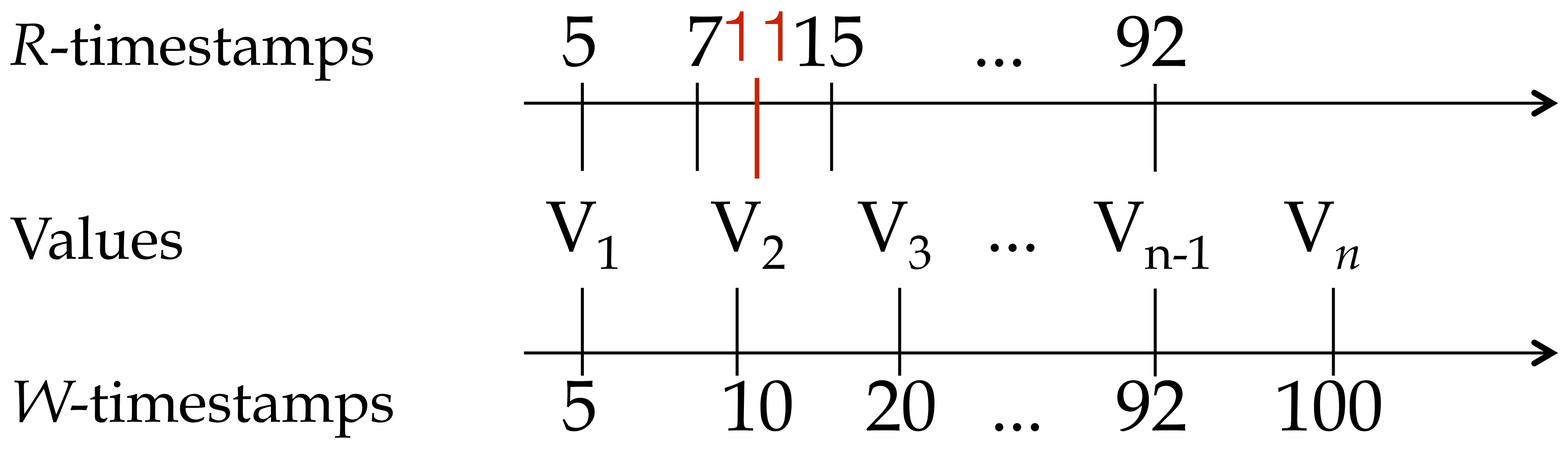
Reads: Read version with biggest timestamp smaller than current timestamp

TO-MVCC



Reads: Read version with biggest timestamp smaller than current timestamp

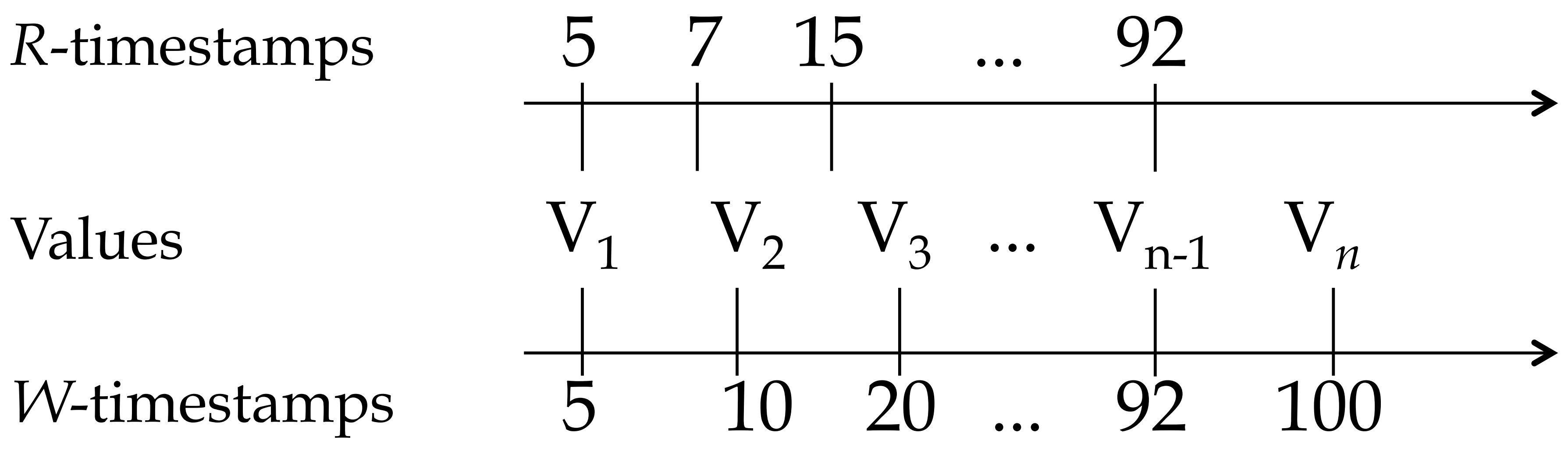
TO-MVCC



Reads: Read version with biggest timestamp smaller than current timestamp

R(X) @ 11 will read V2

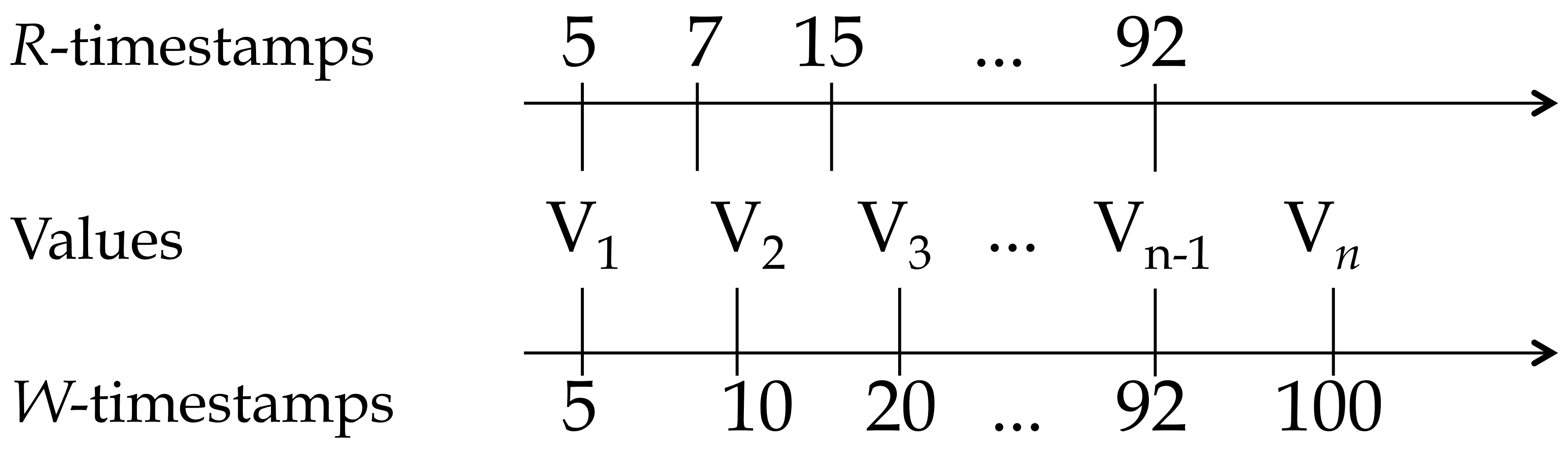
TO-MVCC



Writes: Find interval from given timestamp X to smallest timestamp bigger than X

If there a read is in the interval, abort!

TO-MVCC

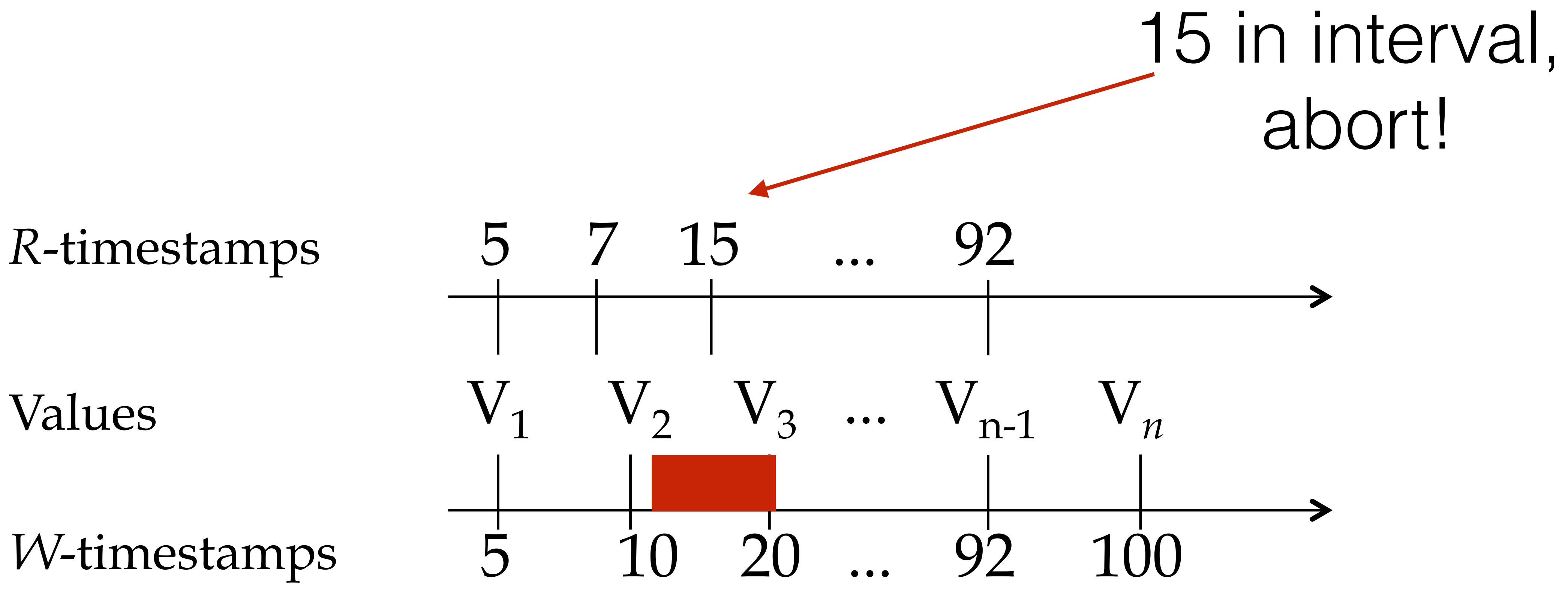


Writes: Find interval from given timestamp X to smallest timestamp bigger than X

If there a read is in the interval, abort!

W(X) @ 11

TO-MVCC

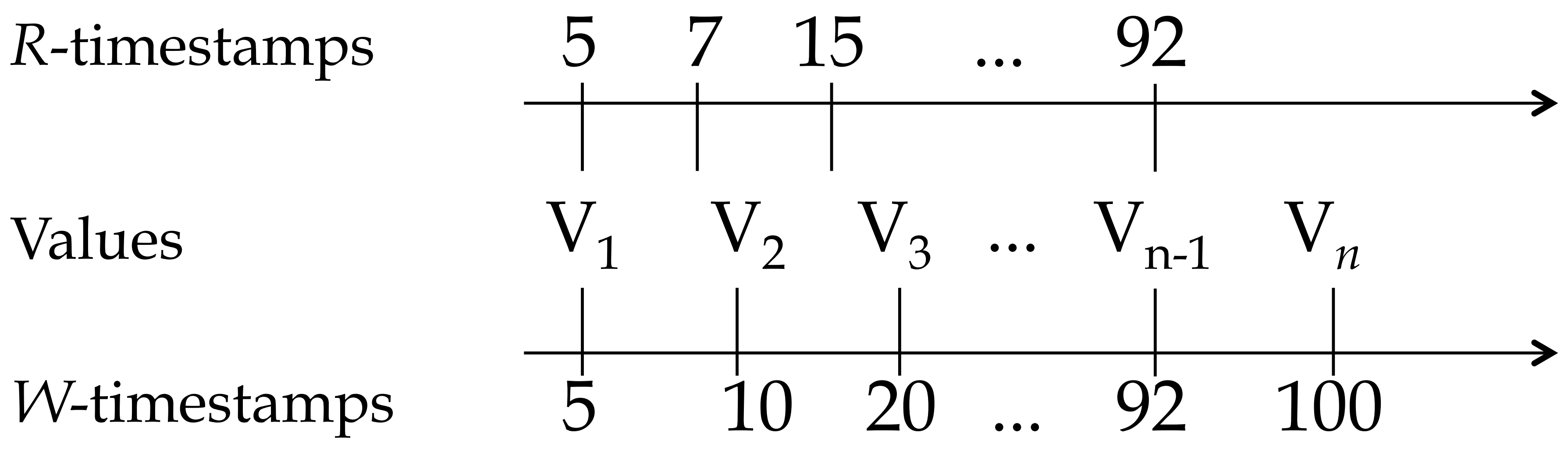


Writes: Find interval from given timestamp X to smallest timestamp bigger than X

If there a read is in the interval, abort!

W(X) @ 11

TO-MVCC

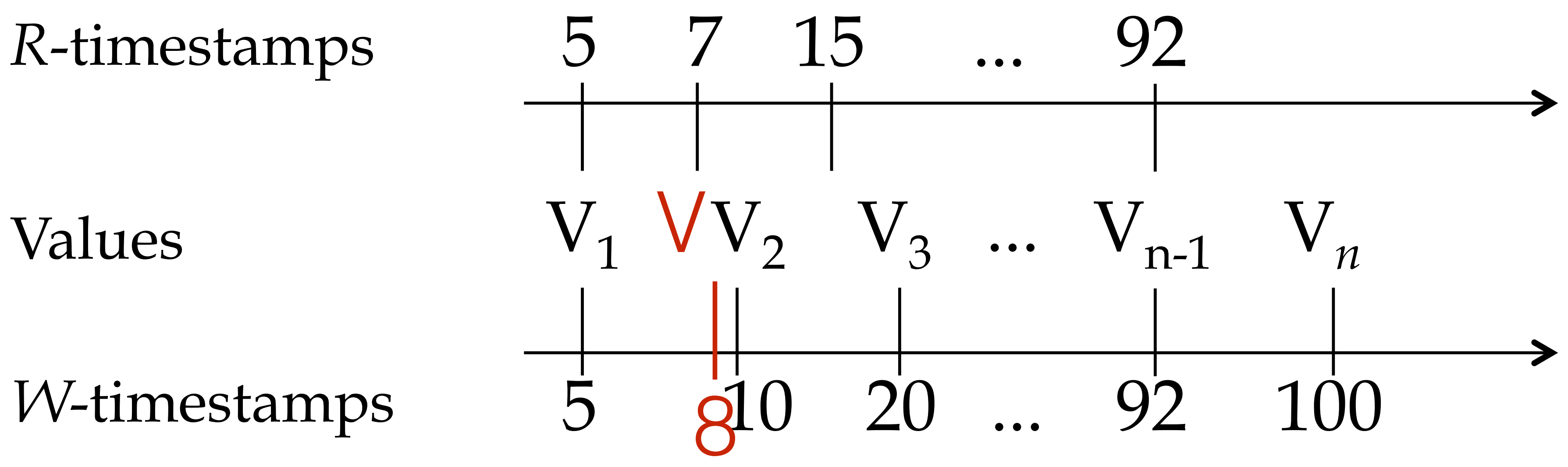


Writes: Find interval from given timestamp X to smallest timestamp bigger than X

If there a read is in the interval, abort!

W(X) @ 8

TO-MVCC



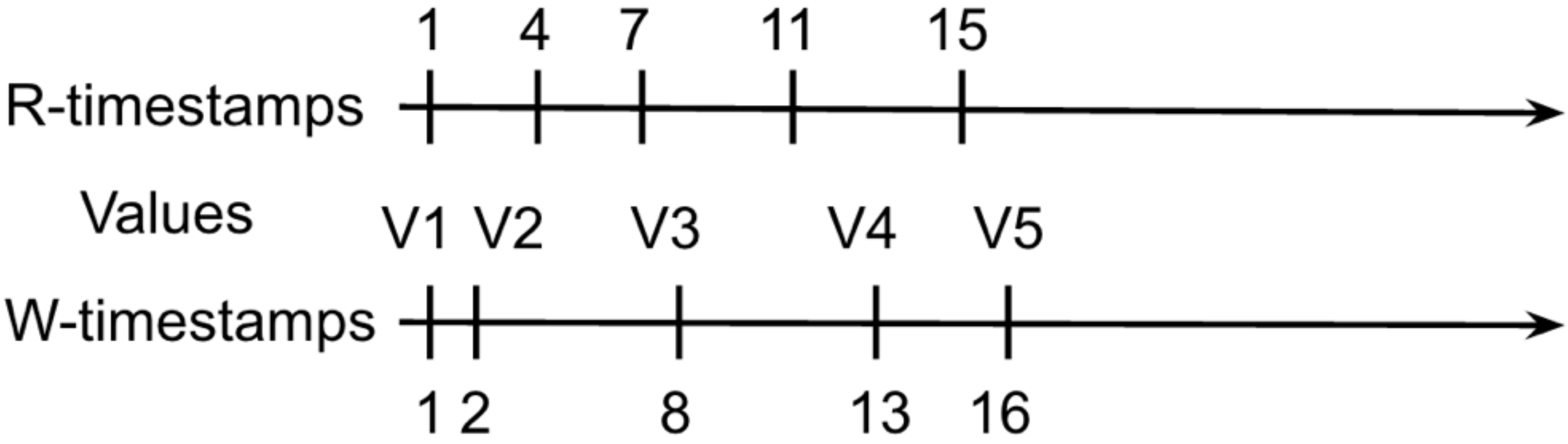
Writes: Find interval from given timestamp X to smallest timestamp bigger than X

If there a read is in the interval, abort!

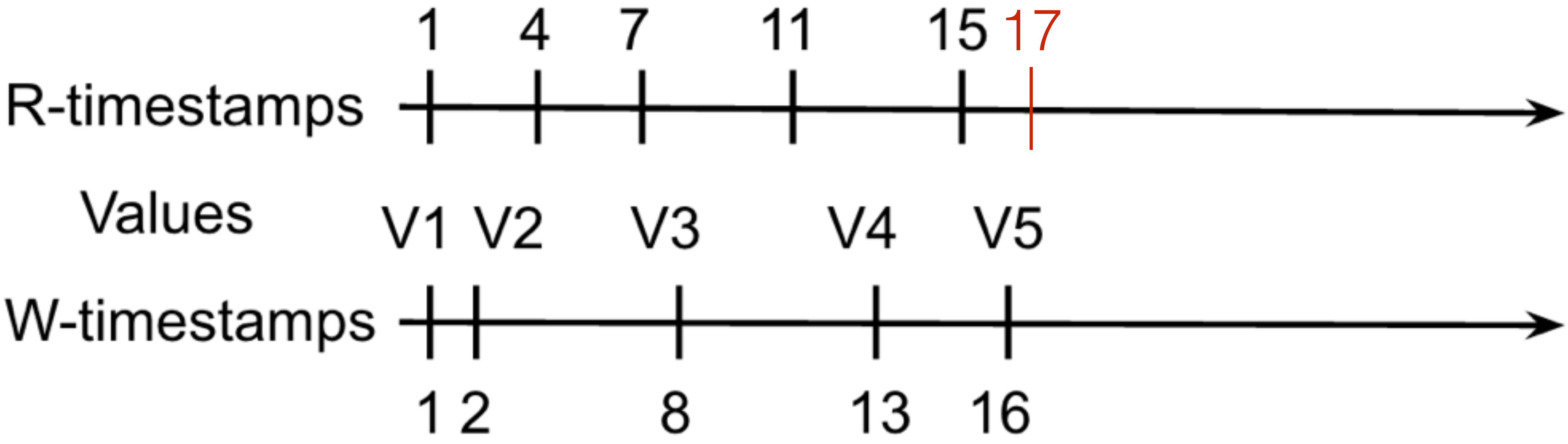
W(X) @ 8

Worksheet - MVCC

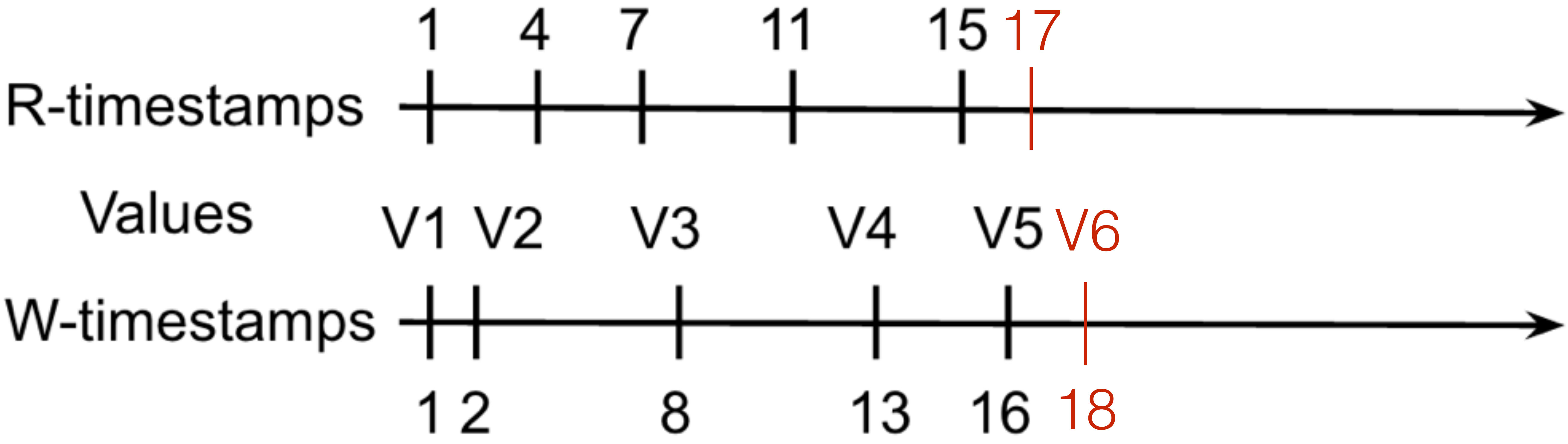
R(X)@17, W(X)@18, W(X)@14, W(X)@12, R(X)@20, R(X)@19, R(X)@23, W(X)@26, W(X)@24



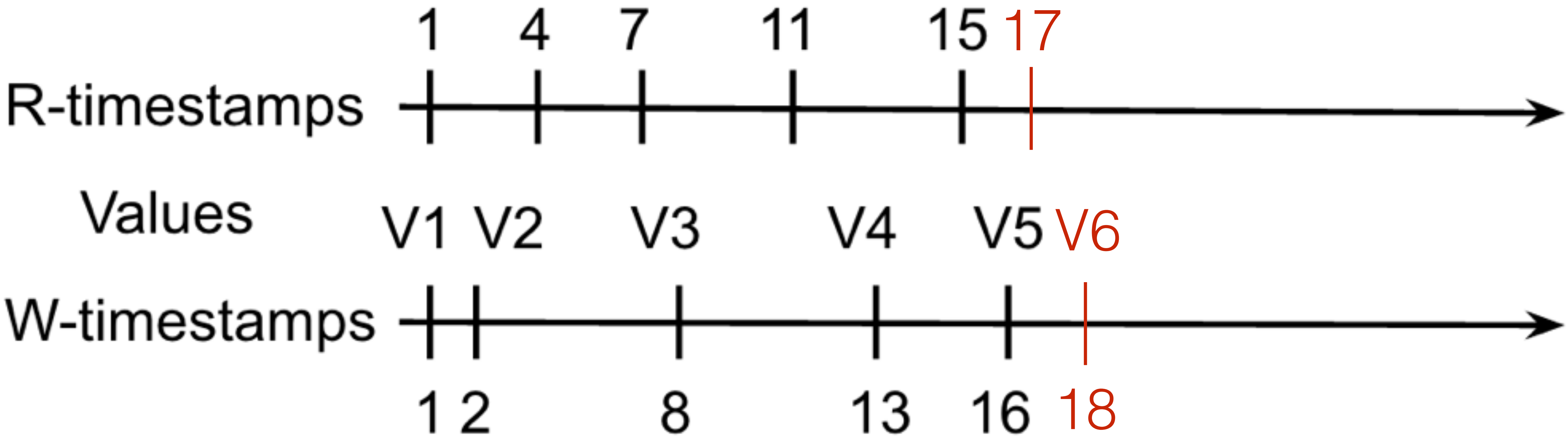
R(X)@17, W(X)@18, W(X)@14, W(X)@12, R(X)@20, R(X)@19, R(X)@23, W(X)@26, W(X)@24



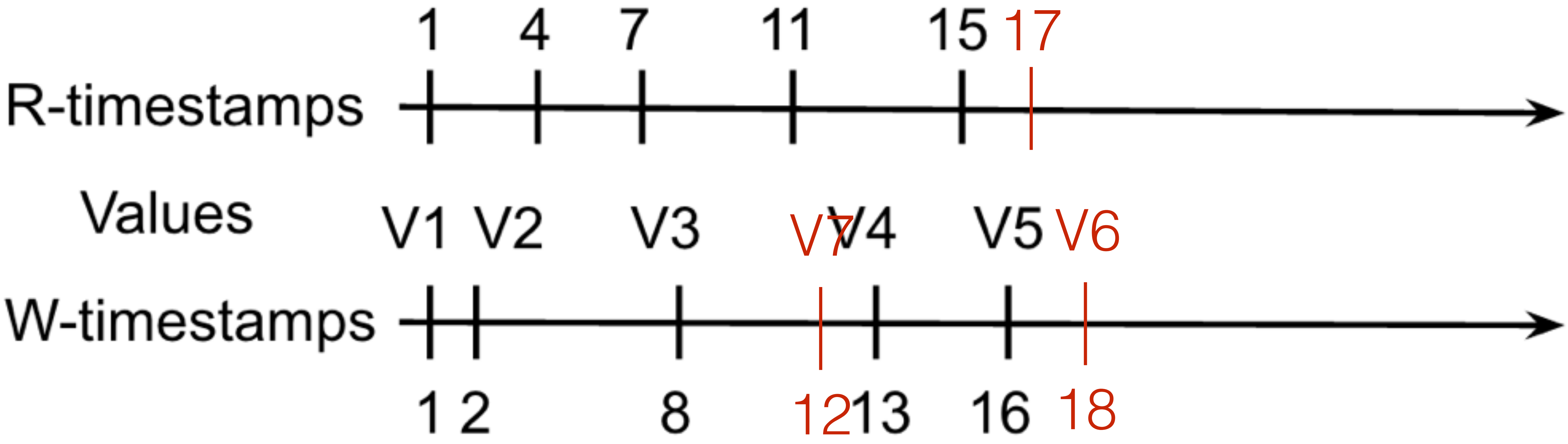
R(X)@17, W(X)@18, W(X)@14, W(X)@12, R(X)@20, R(X)@19, R(X)@23, W(X)@26, W(X)@24



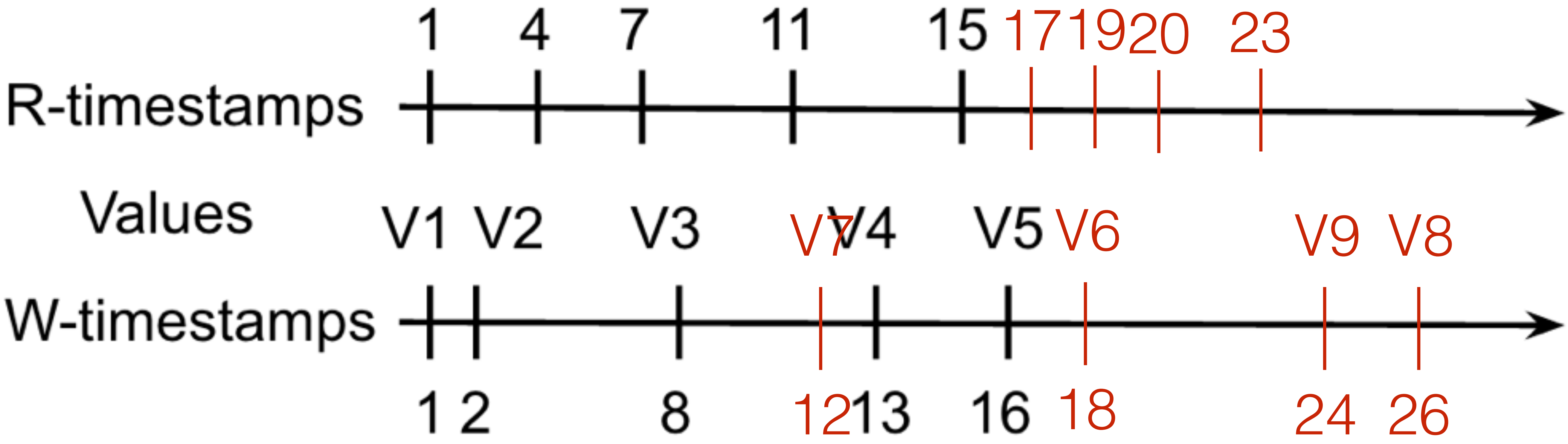
R(X)@17, W(X)@18, ~~W(X)@14~~, W(X)@12, R(X)@20, R(X)@19, R(X)@23, W(X)@26, W(X)@24



R(X)@17, W(X)@18, ~~W(X)@14~~, W(X)@12, R(X)@20, R(X)@19, R(X)@23, W(X)@26, W(X)@24



R(X)@17, W(X)@18, ~~W(X)@14~~, W(X)@12, R(X)@20, R(X)@19, R(X)@23, W(X)@26, W(X)@24



Distributed Concurrency Control

- Parallel database or distributed database
- Each transaction has one coordinator node
- Every node handles its own CC

Two Phase Commit

- Decides whether to commit/abort
- Ensures atomicity and durability
    - Writes happen in all replicas or none

Two Phase Commit

- Phase 1:
  - Coordinator tells participants to prepare
  - Participants respond with “yes” commit or “no” commit
- Phase 2:
  - If **all** participants say “yes”, issue GLOBAL COMMIT
  - Else, abort

2PC & Logging

- Phase 1:
  - Each slave writes an entry to its undo and redo logs
- Phase 2:
  - If abort, the slaves use their undo log to rollback its effects