Timestamp-based Protocol

Timestamp-based protocol: Aim is to order/arrange the transactions globally in such a way that older transactions get priority in the event of a conflict.

* It determines the serializability order of n transactions.

$TS(T_i) \longrightarrow$ unique fixed timestamp for each transaction T_i

Old transaction - $TS(T_i)$

New transaction - $TS(T_j)$

then $TS(T_i) < TS(T_j)$

Methods:

1. System clock
2. logical counter

Timestamp values:-

- W-timestamp(Q): denotes the largest timestamp of any transaction that executed write(Q) operation successfully.

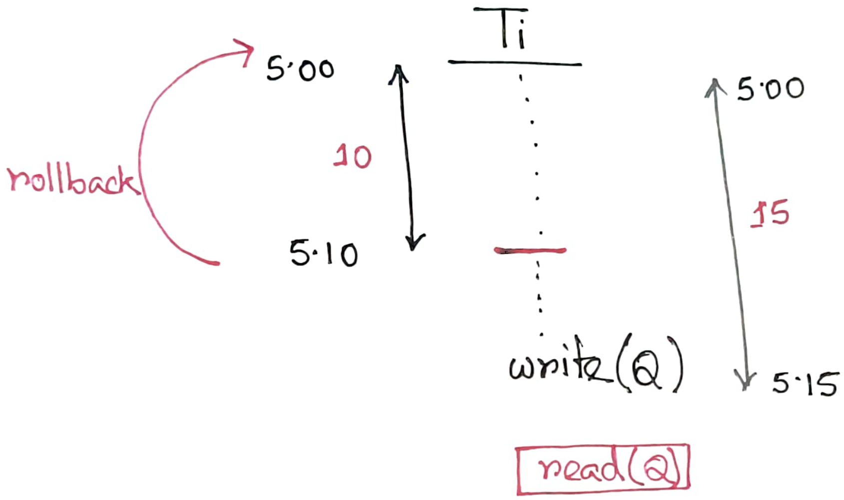
- R-timestamp(Q): denotes the largest timestamp of any transaction that executed read(Q) successfully.

T_1		
	5:00 pm	5:00 pm
r(A)	→ 5 mins	→ 15 mins.
	5:05 pm	
w(A)		5:15 pm

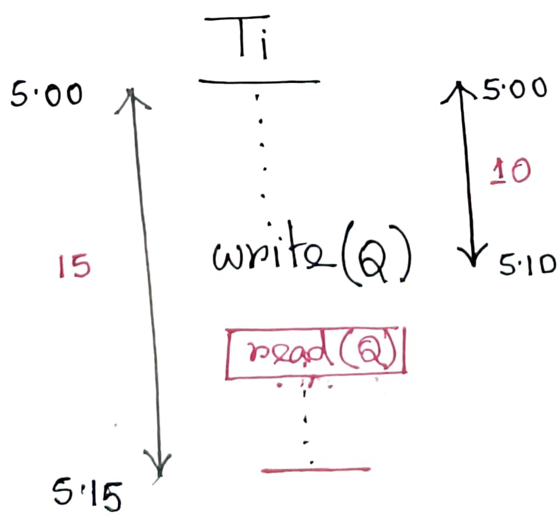
Timestamp ordering Protocol :- This protocol operates as fo

① Suppose that transaction T_i issues read(Q)

a) If $TS(T_i) < W\text{-timestamp}(Q)$, then T_i needs to read a value of Q that was already overwritten. Hence, the read operation is rejected, and T_i is rolled back.

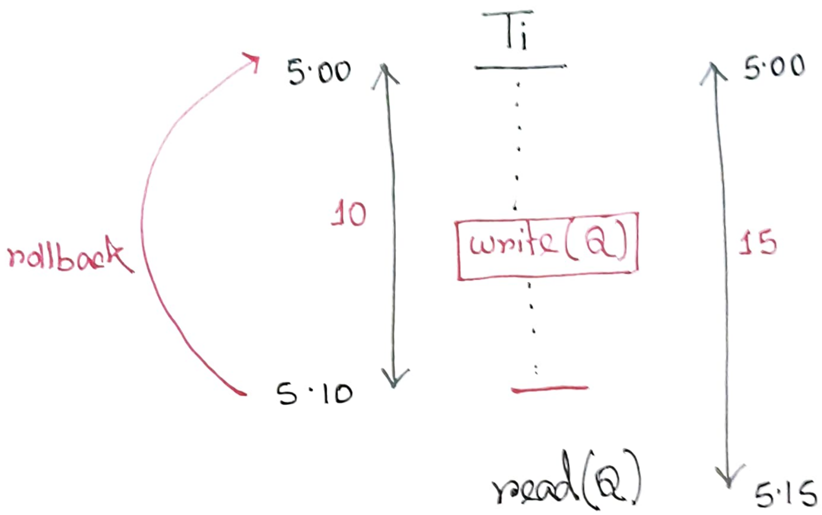


b) If $TS(T_i) \geq W\text{-timestamp}(Q)$, then the read operation is executed, and R-timestamp(Q) is set to the maximum of R-timestamp and $TS(T_i)$



Suppose that transaction T_i issues write(Q)

a) If $TS(T_i) < R\text{-timestamp}(Q)$, then the value of Q that T_i is producing was needed previously, and the system assumed that value would never be produced. Hence, the system rejects the write operation and rolls T_i back.



b) If $TS(T_i) < W\text{-timestamp}$, then T_i is attempting to write an obsolete value of Q. Hence the system rejects this write operation and rolls T_i back.

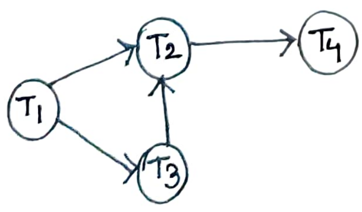
c) Otherwise the system executes the write operation and set W-timestamp(Q) to $TS(T_i)$

Thomas' Write Rule: Above rule 2(a) to 2(c)

Dead Lock

- **Deadlock Prevention** : Two different deadlock prevention schemes
  - → **wait-die** : Nonpreemptive technique.
    When transaction T_i requests a data item currently held by T_j, T_i is allowed to wait only if it has a timestamp smaller than that of T_j (that is, T_i is older than T_j). Otherwise T_i is rolled back (dies)
  - → **wound-wait** : Preemptive technique.
    When transaction T_i requests a data item currently held by T_j, T_i is allowed to wait only if it has a timestamp larger than that of T_j. (that is, T_i is younger than T_j). Otherwise T_j is rolled back (T_j is wounded by T_i)

- **Deadlock Detection** : wait-for graph
  - → When transaction T_i requests a data item currently being held by T_j, then the edge $T_i \rightarrow T_j$ is inserted in the wait-for graph.
  - → A deadlock exists in the system if and only if the wait for graph contains a cycle.



- Transaction T_1 is waiting for T_2 & T_3
- Transaction T_3 is waiting for T_2
- Transaction T_2 is waiting for T_4

No cycles → not in a deadlock state