TIMESTAMP-BASED PROTOCOLS

Timestamp-Based Protocols

Timestamp-Based Protocols

- Each transaction is issued a timestamp when it enters the system. If an old transaction T_i has time-stamp TS(T_i), a new transaction T_j is assigned time-stamp TS(T_j) such that TS(T_i) <TS(T_j).
- The protocol manages concurrent execution such that the time-stamps determine the serializability order

- In order to assure such behavior, the protocol maintains for each data Q two timestamp values:
  - **W-timestamp**(Q) is the largest time-stamp of any transaction that executed **write**(Q) successfully
  - **R-timestamp**(Q) is the largest time-stamp of any transaction that executed **read**(Q) successfully

Timestamp-Based Protocols

Timestamp-Based Protocols

- The timestamp ordering protocol ensures that any conflicting **read** and **write** operations are executed in timestamp order

- Suppose a transaction T_i issues a **read**(Q)
  1. If TS(T_i) ≤ **W**-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.
  2. If TS(T_i) ≥ **W**-timestamp(Q), then the **read** operation is executed, and **R**-timestamp(Q) is set to **max**(**R**-timestamp(Q), TS(T_i)).

Timestamp-Based Protocols (Cont.)

Timestamp-Based Protocols (Cont.)

- Suppose that transaction T_i issues **write**(Q).
  1. If TS(T_i) < **R**-timestamp(Q), then the value of Q that T_i is producing was needed previously, and the system assumed that that value would never be produced
     - Hence, the **write** operation is rejected, and T_i is rolled back
  2. If TS(T_i) < **W**-timestamp(Q), then T_i is attempting to write an obsolete value of Q
     - Hence, this **write** operation is rejected, and T_i is rolled back
  3. Otherwise, the **write** operation is executed, and **W**-timestamp(Q) is set to TS(T_i)

Example Use of the Protocol

Example Use of the Protocol

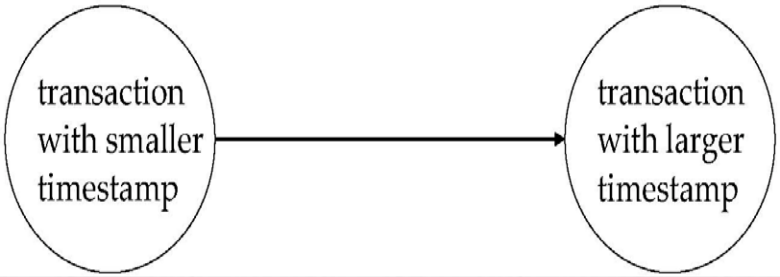
A partial schedule for several data items for transactions with timestamps 1, 2, 3, 4, 5

T_1	T_2	T_3	T_4	T_5
				read (X)
	read (Y)			
read (Y)				
		write (Y)		
		write (Z)		
				read (Z)
	read (Z)			
	abort			
read (X)				
			read (W)	
		write (W)		
		abort		
				write (Y)
				write (Z)

Correctness of Timestamp-Ordering Protocol

Correctness of Timestamp-Ordering Protocol

- The timestamp-ordering protocol guarantees serializability since all the arcs in the precedence graph are of the form:



Thus, there will be no cycles in the precedence graph

- Timestamp protocol ensures freedom from deadlock as no transaction ever waits
- But the schedule may not be cascade-free, and may not even be recoverable