**Protocolo de estampillas y multiversión**

**Protocolo de estampillas de tiempo**

**Notaciones:**

* ts(Ti) = estampilla de tiempo de Ti
* R-ts(Q) = estampilla de lectura
* W-ts(Q) = estampilla de escritura

**Cada transacción Ti se le asigna una estampilla de tiempo ts(Ti)** (hora de inicio de la transacción)

Las reglas del protocolo garantizan que la planificación sea equivalente a la serie producida por el orden de las estampillas: Si ts(Ti) < ts(Tj) el protocolo asegura que la planificación obtenida será equivalente a la serie Ti seguido de Tj. Si no pude garantizar la serie impuesta por las estampillas hará retroceder a alguna de las transacciones (por alguna de reglas del protocolo).

**Cada dato Q tiene una estampilla de lectura R-ts(Q) y una estampilla de escritura W-ts(Q),** que **corresponden** a la **estampilla** de la **transacción** **más** **nueva** (**estampilla** **mayor**) **que** **leyó** y **escribió**, **respectivamente**, **con** **éxito** el **dato** **Q**. Cada vez que una transacción **Ti ejecuta con éxito:**

* Read(Q), entonces a R-ts(Q) se le asigna el valor mayor entre R-ts(Q) y ts(Ti)
* Write(Q), entonces a W-ts(Q) se le asigna el valor mayor entre W-ts(Q) y ts(Ti)

**Reglas: cada vez** queuna **transacción Ti intenta ejecutar** una **instrucción** se **verifica:**

1. Si Ti ejecuta Read(Q) y ts(Ti) < W-ts(Q) entonces Ti retrocede (Ti intenta leer un “valor del futuro” producido por una transacción posterior)
2. Si Ti ejecuta un Write(Q)
   * Si ts(Ti) < R-ts(Q) entonces Ti retrocede (la escritura “llego tarde”, una transacción posterior a Ti ya leyó Q)
   * Si ts(Ti) < W-ts(Q) entonces Ti retrocede (una transacción posterior “pisó” el valor de Q)

En este mismo caso por regla de Thomas la operación Write(Q) se puede omitir (no se realiza) y Ti continua su ejecución (siempre y cuando no haya retrocedido por otra regla).

1. En cualquier otro caso la instrucción se realiza con éxito y se actualiza la estampilla correspondiente del dato Q según la instrucción:
   * Read(Q) => R-ts(Q) ← Mayor ( R-ts(Q), ts(Ti) )
   * Write(Q) => W-ts(Q) ← Mayor ( W-ts(Q), ts(Ti) )

**Reglas resumidas**

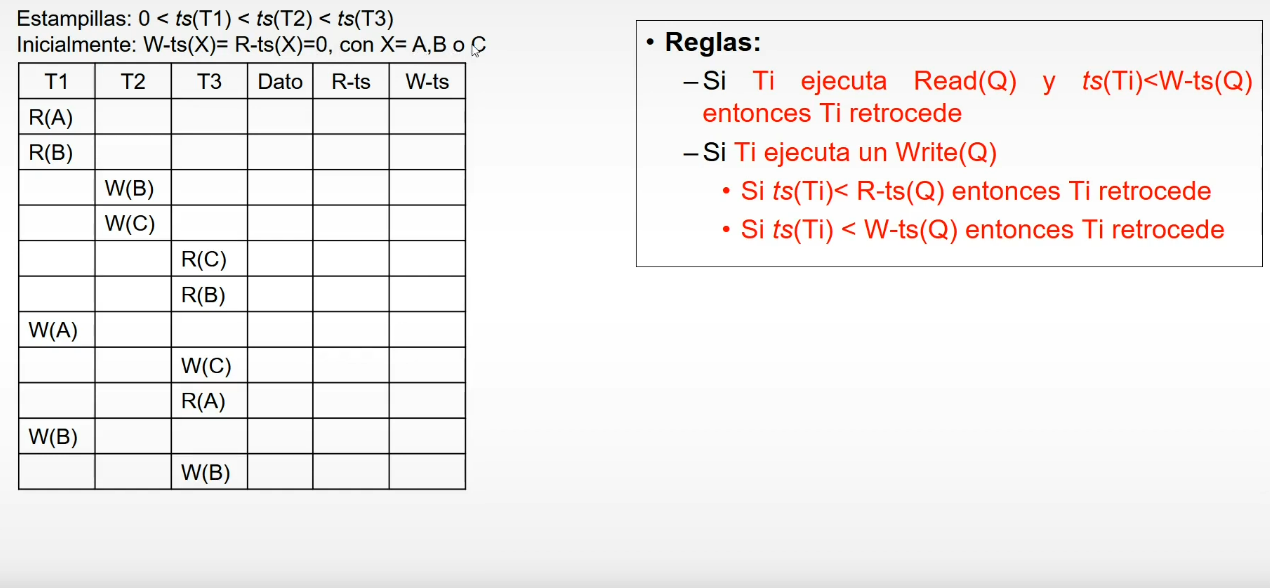
Si Ti ejecuta Read(Q) y ts(Ti) < W-ts(Q) entonces Ti retrocede

Si Ti ejecuta Write(Q):

* Si ts(Ti) < R-ts(Q) entonces Ti retrocede
* Si ts(Ti) < W-ts(Q) entonces Ti retrocede pero con regla Thomas se omite

Inicialmente tenemos las estampillas de lectura y escritura en 0 para todos los datos

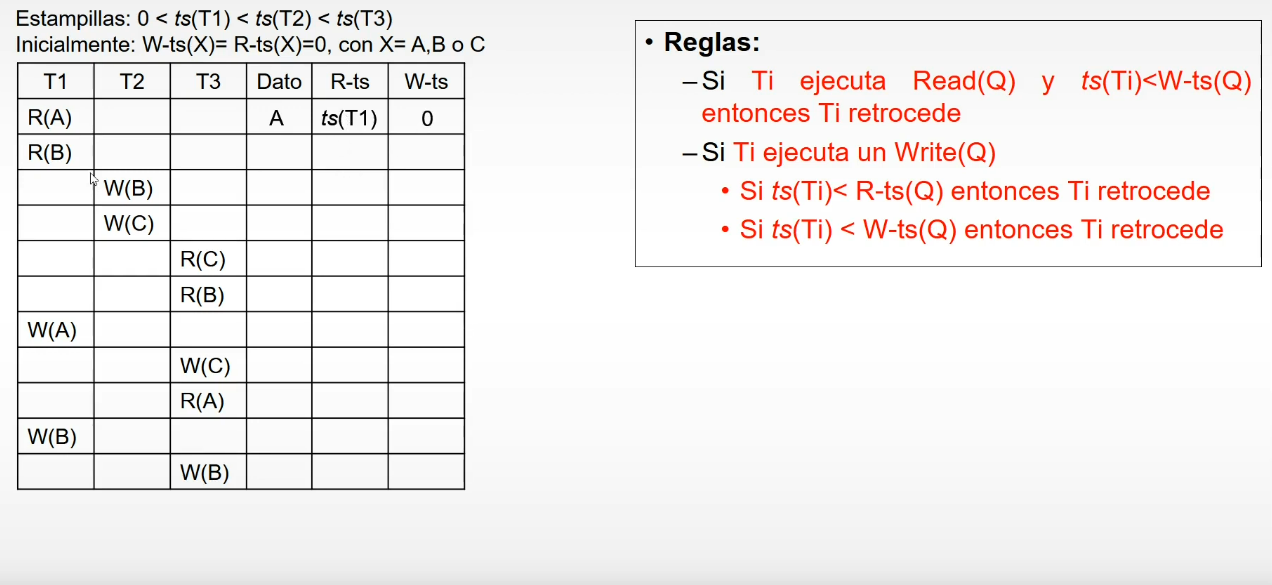
El ordenamiento de las estampillas de la imagen quiere decir que primero ingreso al sistema T1, luego T2, luego T3.



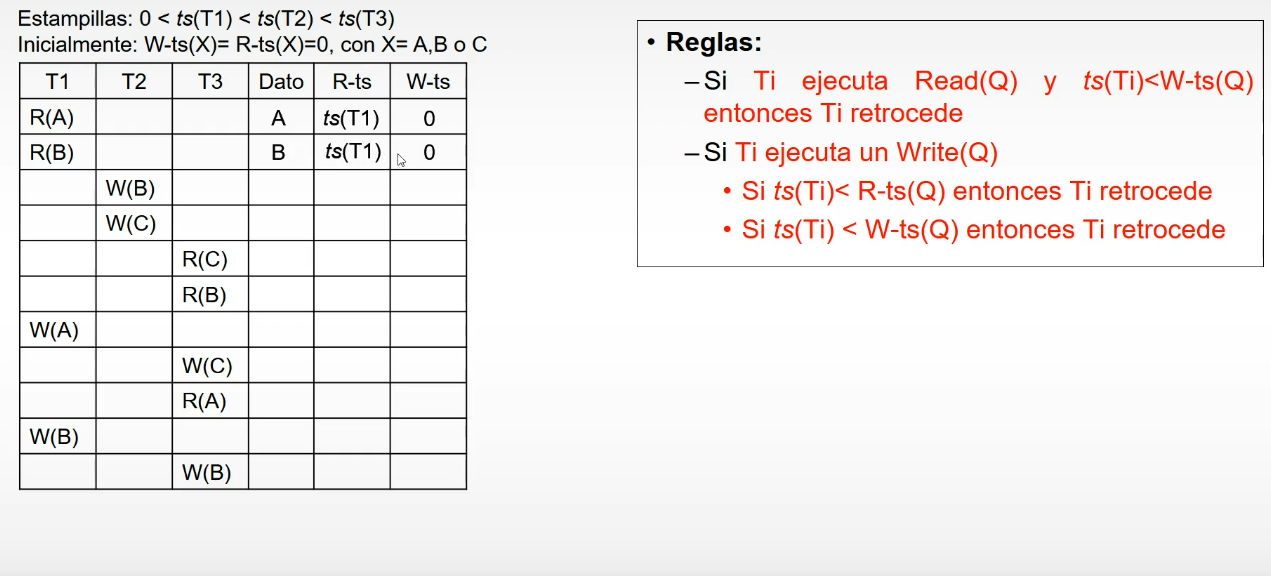
Iremos completando la tabla viendo cómo se actualiza la estampilla de lectura y escritura según la instrucción.

Read(A) lee el dato A por lo tanto en dato ponemos A, verificamos si la estampilla de escritura del dato A (W-ts(A)) es mayor a la estampilla de transacción que está leyendo, como inicialmente están en 0 entonces se puede realizar la operación.

Actualizamos la estampilla de lectura a T1 y la de escritura queda en 0 porque estamos haciendo una lectura



Con R(B) pasa lo mismo



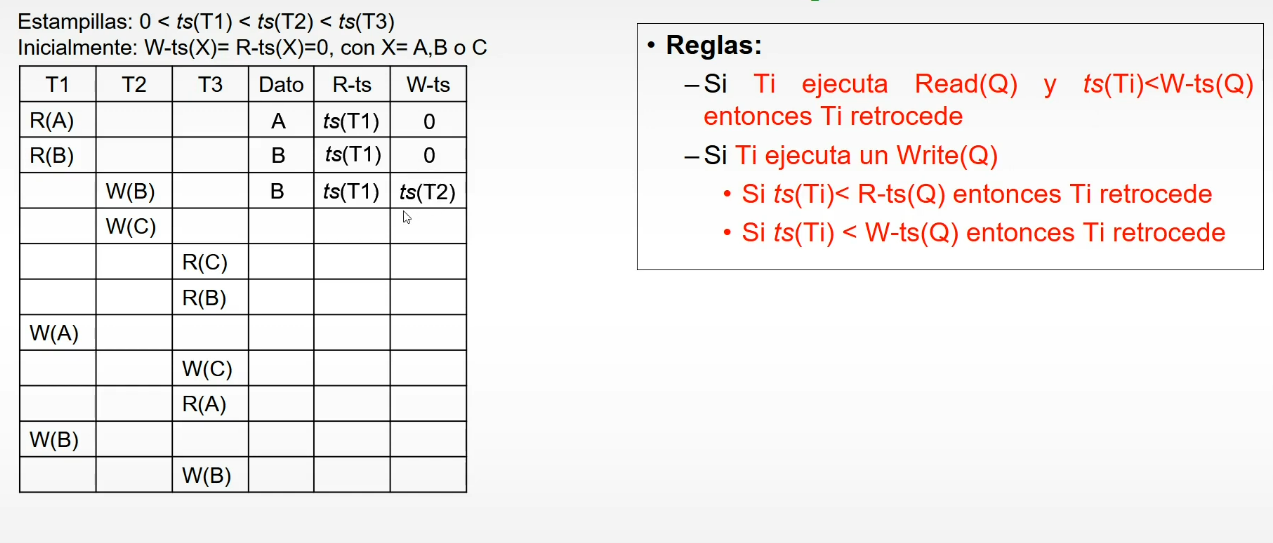
T2 ejecuta W(B) debemos verificar la regla de escritura

Verificamos si la estampilla de lectura de B no es mayor que la estampilla de T2:

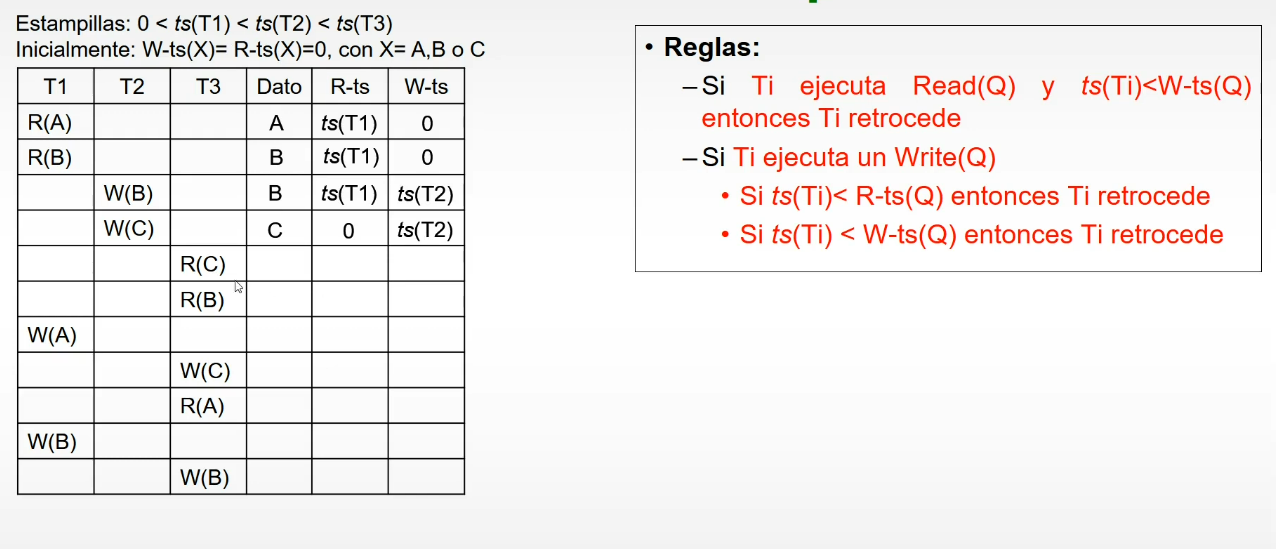
La estampilla de lectura de B está en la estampilla de T1 que es menor que T2, esto es:

¿Ts(T2) < R-ts(B)? esto es igual ¿Ts(T2) < Ts(T1)? Falso, por lo tanto, no retrocede y se escribe.

Entonces la estampilla de lectura queda como estaba y la de escritura se actualiza con la estampilla de T2 porque siempre se actualiza con la mayor y la estampilla T2 > 0.

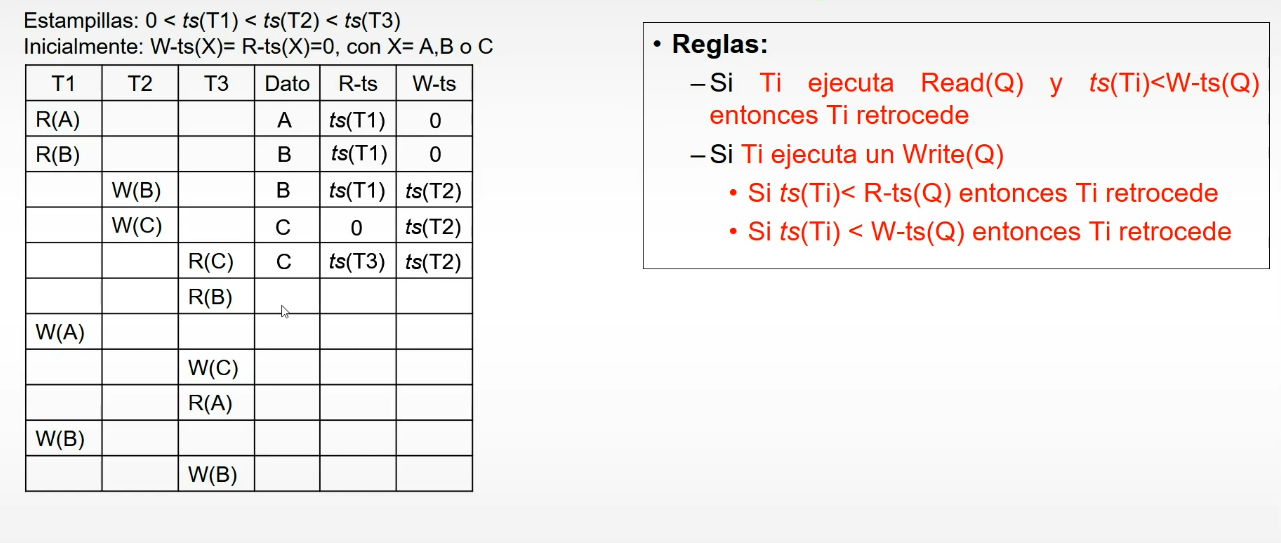


T2 ejecuta W(C), cuya estampilla de lectura W-ts(Q) = 0, la estampilla de lectura queda en 0 porque estamos escribiendo, y actualizamos la estampilla de escritura a ts(T2) porque estaba inicialmente en 0 (creo que es porque ts(T2) < 0 es falso entonces como no retrocede, escribe).

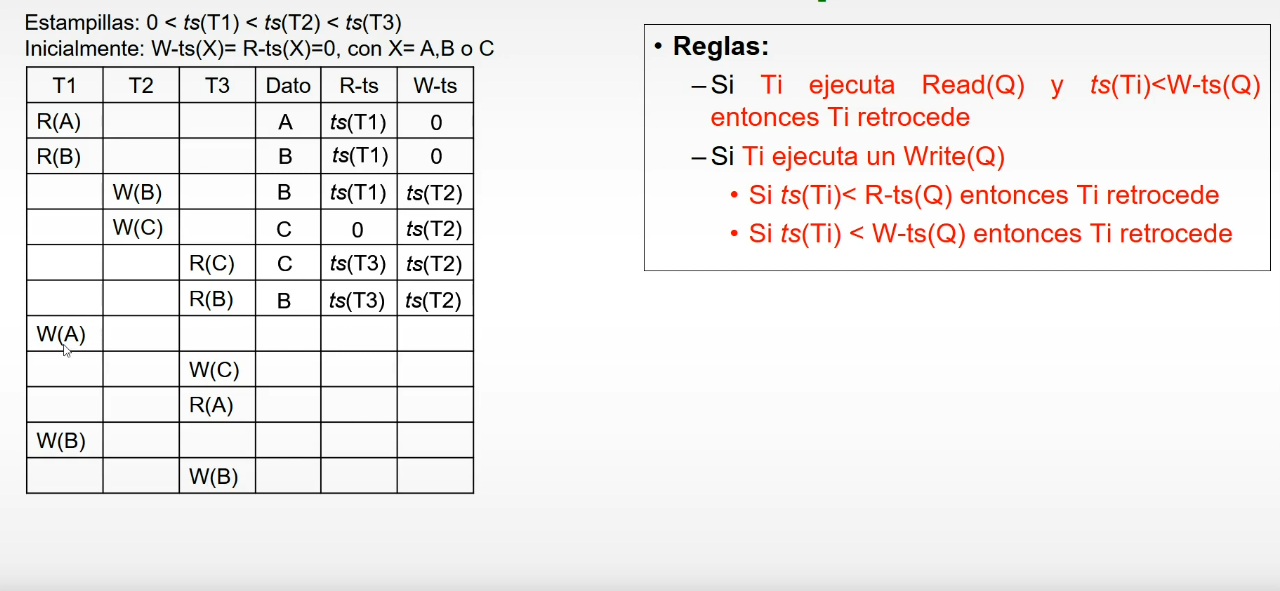


T3 ejecuta R(C) aplicamos la 1er regla

W-ts(C) = ts(T2) viendo la tabla luego ¿ts(T3) < ts(T2)? Falso entonces no retrocede por lo tanto se ejecuta el read con exito y se actualiza la estampilla de lectura con ts(T3) y la de escritura queda como estaba con ts(T2).



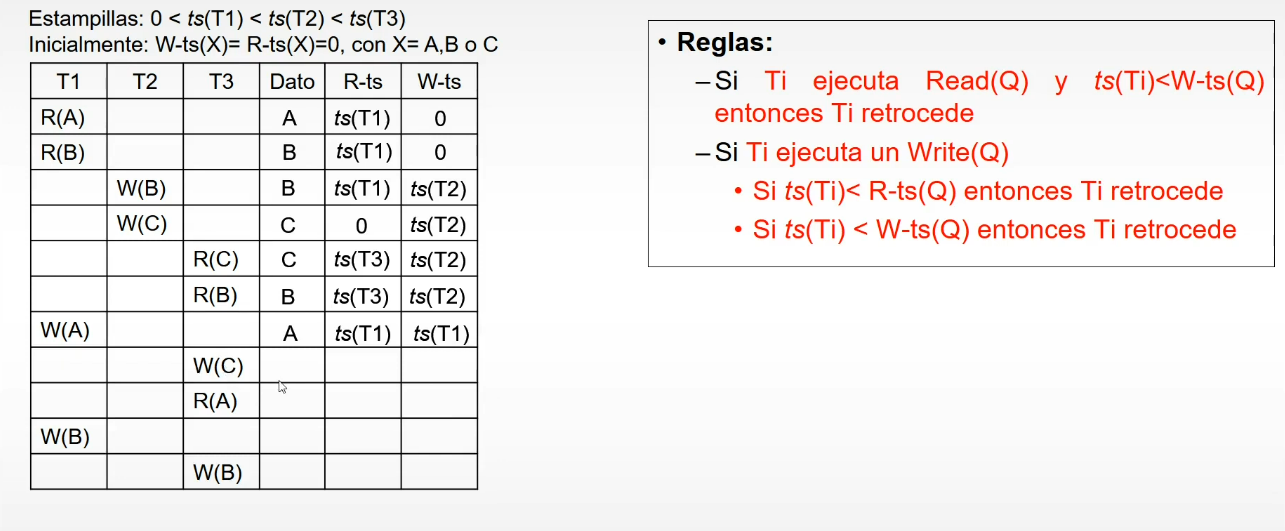
T3 ejecuta R(B) aplicamos 1er regla y como ts(T3) < W-ts(T2) es falso entonces se lee con éxito, actualizamos la estampilla de lectura con ts(T3) y la de escritura queda como estaba o sea ts(T2).



T1 ejecuta W(A), en la tabla la última modificación de A esta arriba del todo, aplicamos la 2da regla las 2 condiciones.

ts(T1) < ts(T1) ? Falso entonces se deja ts(T1)

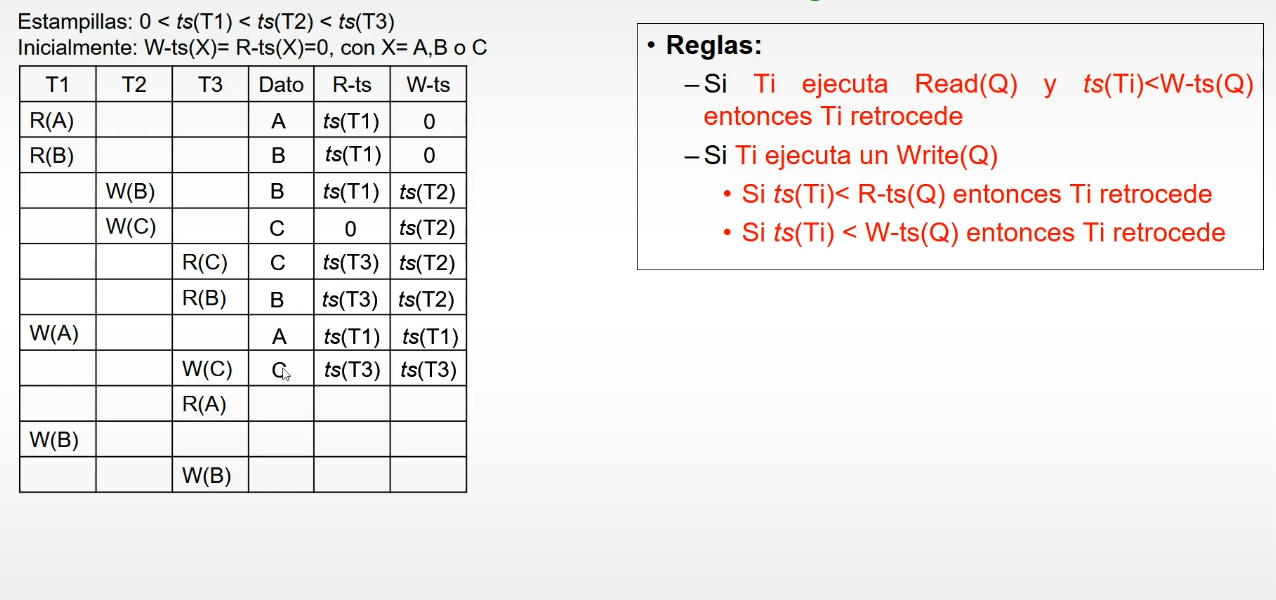
ts(T1) < 0? Falso entonces se escribe ts(T1)



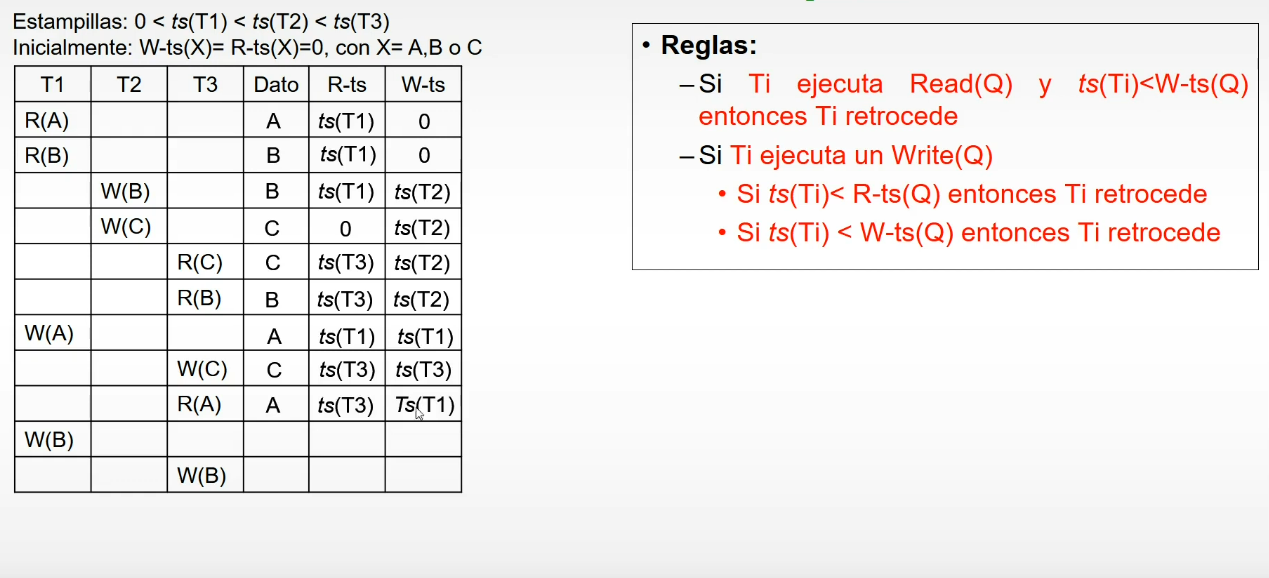
T3 ejecuta W(C) vemos en la tabla la última modificación de T3.

ts(T3) < ts(T3)? Falso se deja ts(T3)

ts(T3) < ts(T2)? Falso se escribe ts(T3)

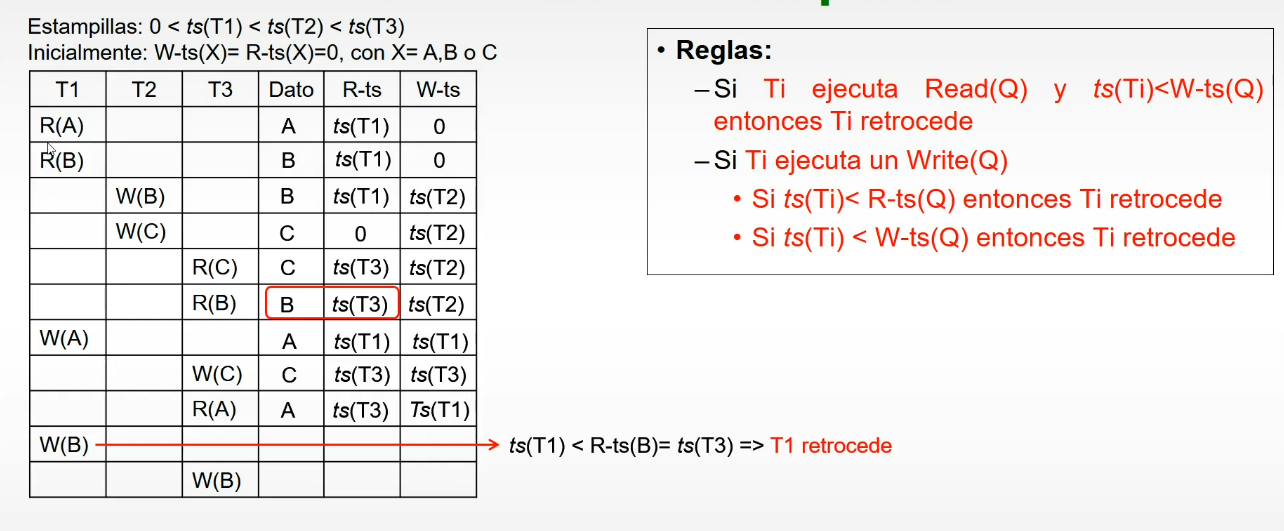


T3 ejecuta R(A) como ts(T3) < ts(T1) es falso se guarda en la estampilla de lectura ts(T3) y la estampilla de escritura queda como esta.

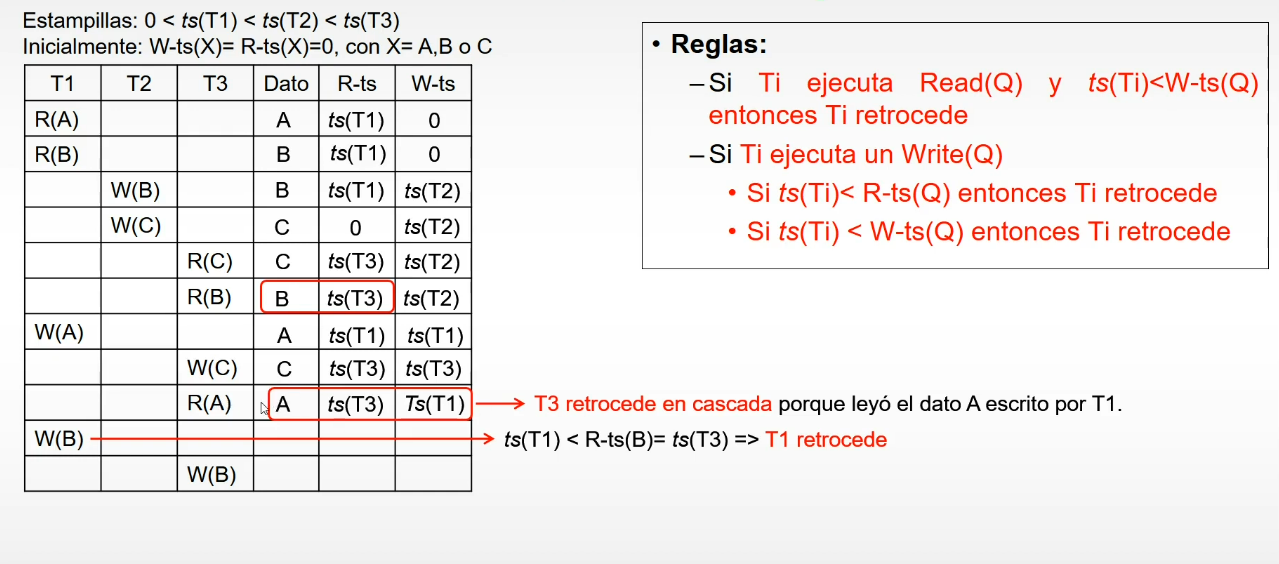


T1 ejecuta W(B) vemos la última vez que se actualizo B en la tabla

ts(T1) < ts(T3)? Verdadero por lo tanto T1 retrocede



**Como T1 retrocede** hay que ver si no hay retroceso en cascada, hay que ver si hay alguna transacción distinta a T1 que haya leído algún dato por escrito por T1.

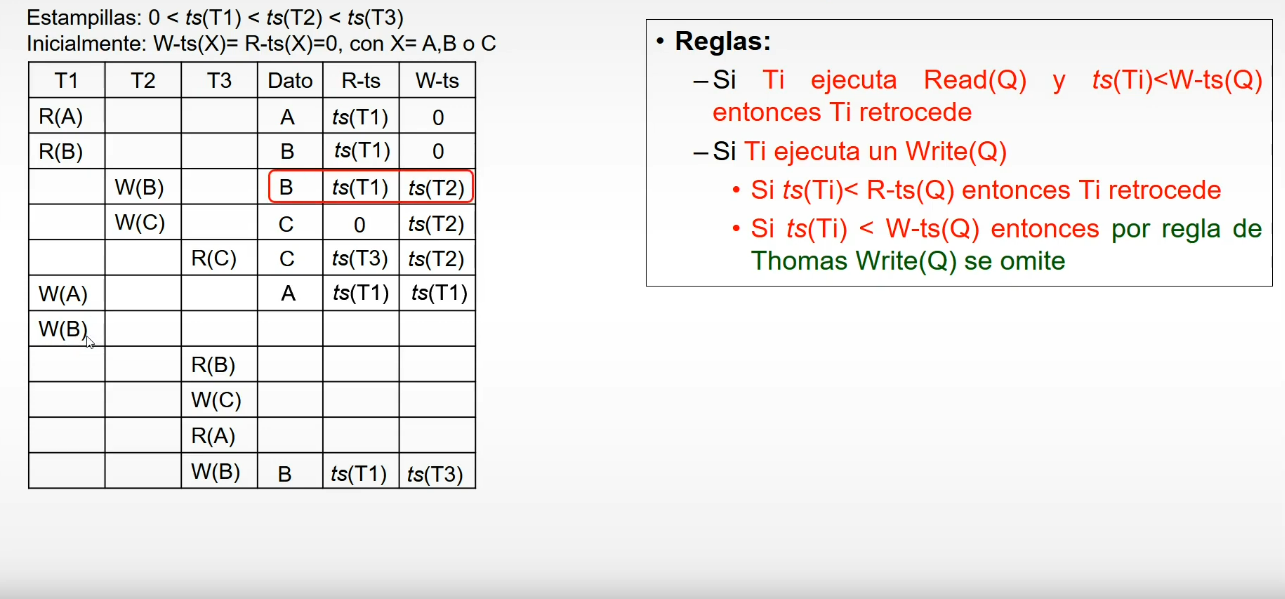


Esto ocurre en la instrucción anterior donde se ve que T3 leyó el dato escrito por T1, por lo tanto, T3 retrocede en cascada.

Repetimos el proceso para T3.

T2 como no lee ningún dato no se ve afectado ni por T1 ni por T3.

Otro ejemplo donde se utiliza la regla de Thomas:

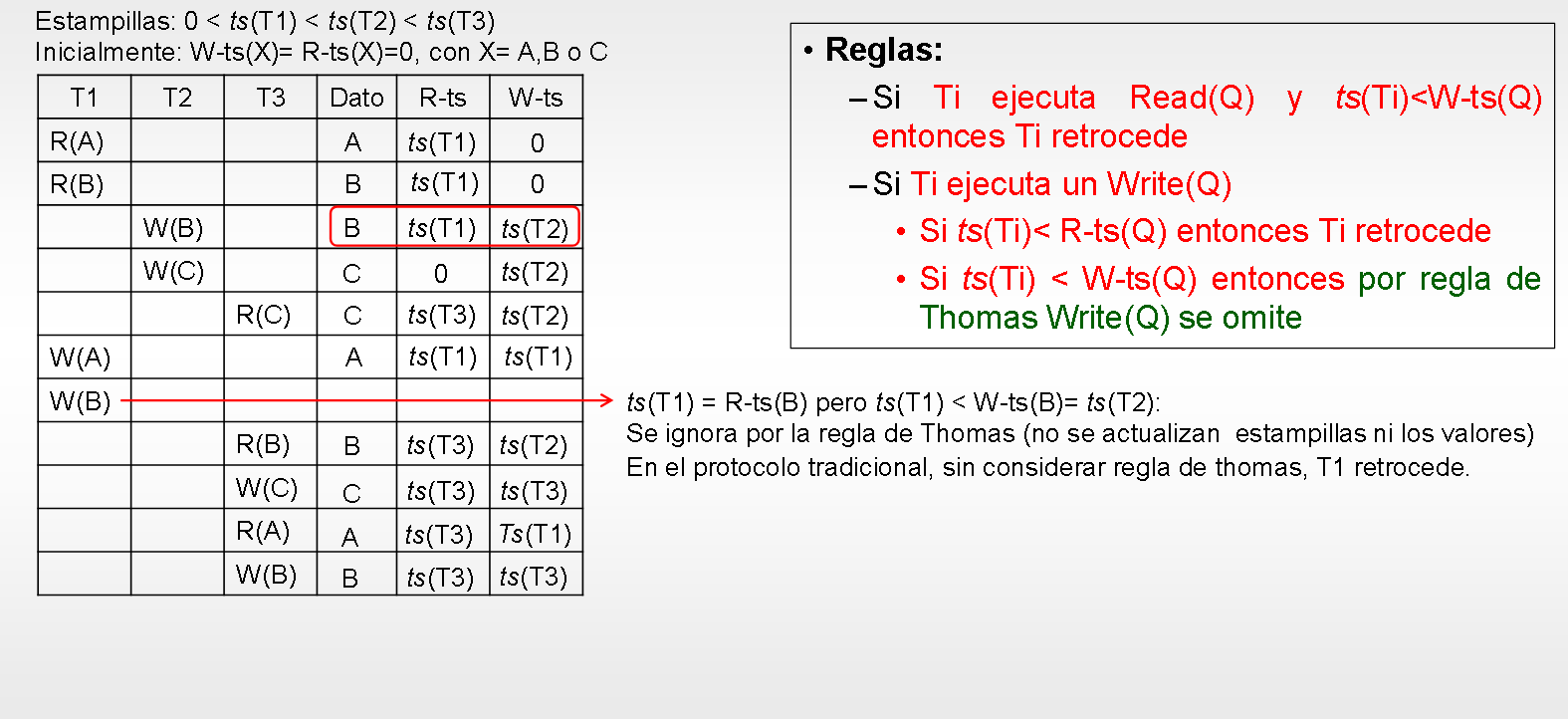


T1 ejecuta W(B)

Ts(T1) < ts(T1)? Falso entonces se escribe ts(T1)

**Ts(T1) < ts(T2)? Verdadero por regla de Thomas ignoramos esta instrucción**

**Sigue la planificación normalmente**

****

**Protocolo multiversion**

Cada transacción Ti se le asigna una estampilla de tiempo ts(Ti) (hora de inicio de la transacción)

• **El primer Write(Q) de cada transacción crea una nueva versión del dato Q**. Existe una **versión incial Q0** para cada dato Q.

**El número que acompaña a Q será el numero de la transacción en la que se ubique.**

• Para cada versión Qk se almacenan:

– El valor de la versión Qk

– R-ts(Qk) la estampilla de tiempo mayor de todas las transacciones que leyeron Qk

– W-ts(Qk) la estampilla de tiempo de la transacción que creo la versión Qk

Cuando una transacción Ti ejecuta Read(Q) se selecciona la versión de Q correspondiente a Ti, esta es: la versión Qk tal que W-ts(Qk) es la estampilla mas grande de todas las versiones de Q que verifica W-ts(Qk) <= Ts(Ti) (i.e., la versión anterior mas próxima en el tiempo a Ti)

**Reglas: para una transacción Ti, sea Qk la versión correspondiente a Ti (ver ítem anterior)**

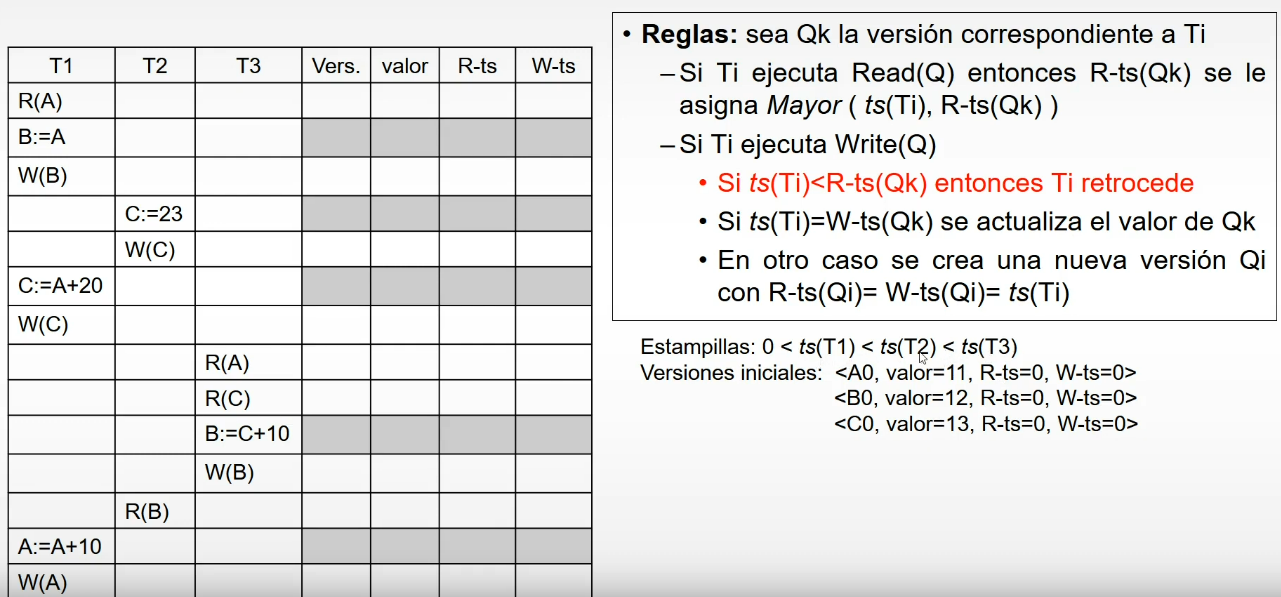
– Si Ti ejecuta Read(Q) entonces R-ts(Qk) se le asigna Mayor ( ts(Ti), R-ts(Qk) )

– Si Ti ejecuta Write(Q)

* Si ts(Ti) < R-ts(Qk) entonces Ti retrocede (la escritura “llego tarde”)
* Si ts(Ti) = W-ts(Qk) entonces solo se actualiza el valor de Qk (es la versión que creo Ti)
* En otro caso se crea una nueva versión Qi con R-ts(Qi)= W-ts(Qi)= ts(Ti)

Recordar: **Los read(Q) nunca fallan**

Ejemplo:



**Algoritmo COMPLETO:**

**Reglas resumidas**

Si Ti ejecuta Read(Q) y ts(Ti) < W-ts(Q) entonces Ti retrocede

Si Ti ejecuta Write(Q):

* Si ts(Ti) < R-ts(Q) entonces Ti retrocede
* Si ts(Ti) = W-ts(Q) entonces Ti retrocede pero con regla Thomas se omite

Seguir las reglas y recordar que **cuando ejecutamos un read/write** hay que ver con que versión nos quedamos, **nos quedamos con la que mayor W-ts tenga que sea menor que W-ts(Ti).**

* El read no cambia la versión QK
* El write si es que no retrocede entonces actualiza QK

Recordar: El read no modifica el W-ts

