**Protocolo 2 fases y árbol**

**Para cada transacción**

Antes de escribir W(X) o leer R(X) un dato X se debe solicitar y obtener el bloqueo**1** correspondiente sobre X. Si no logra obtener un bloqueo la transacción espera hasta que ese dato es liberado por otra transacción.

Debe respetar 2 fases:

**Fase de crecimiento:** La transacción solo puede **solicitar o aumentar (upgrade) bloqueos.**

**Fase de decrecimiento:** Una vez que una transacción se **libero** o **disminuyo** (**downgrade**) un **bloqueo** no puede solicitar o aumentar más bloqueos, solo puede liberar o disminuir.

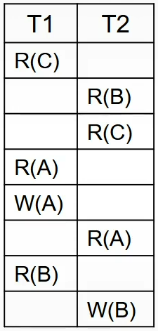
Resumen según el profe: Para obtener el bloqueo el dato tiene que estar sin utilizar o en un bloqueo compatible (puede ser que 2 transacciones bloqueen el mismo dato para lectura) y respetar las 2 fases.

Para verificar si es posible obtener la planificación anterior agregaremos bloqueos de la siguiente forma:

1. Solicitar bloqueo mínimo permitido para poder ejecutar la instrucción
2. Solicitar los bloqueos lo mas tarde posible en la planificación
3. Liberar los bloqueos lo antes posible (siempre respetando 2 fases)

**Ejemplo**

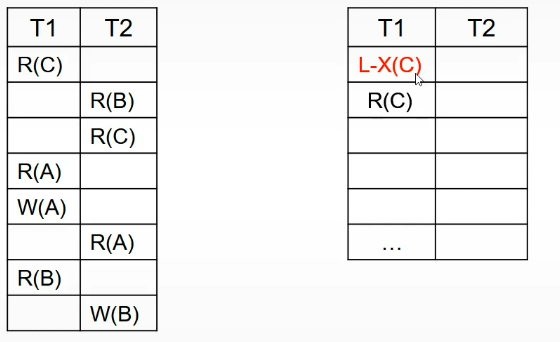
Utilizando solo **bloqueos** **exclusivos** Lock-X(A) o L-X(A) tanto para leer como para escribir un dato A.



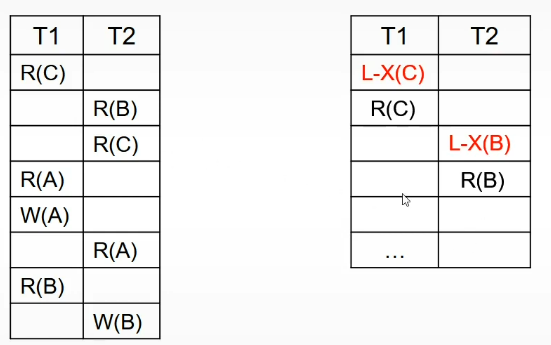
Respetando el orden de las instrucciones iremos incorporando los bloqueos.

* **Factos:  
  Una vez unlockeado un dato no se puede usar para ninguna otra cosa, ni leer, ni escribir.**
* **Upgrade y Downgrade se utilizan en la misma transición**

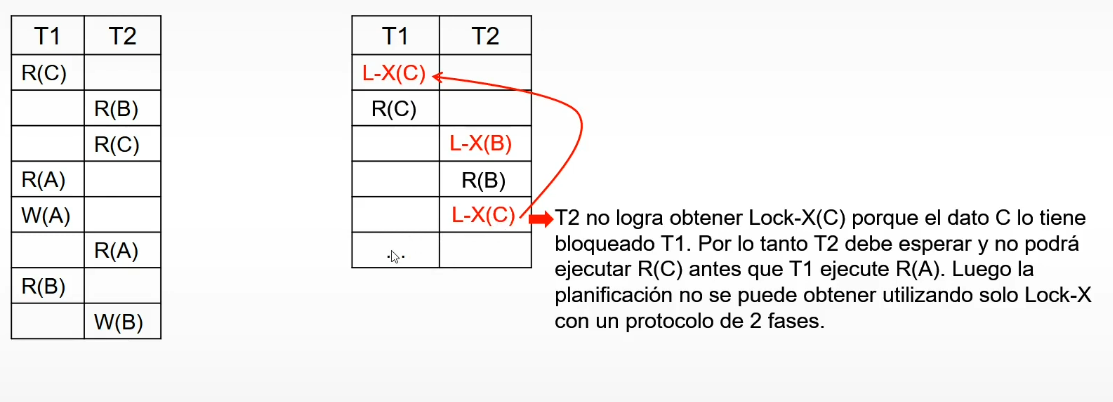
Como T1 empieza leyendo el dato C requerimos el bloqueo exclusivo sobre el dato C y lo obtiene porque nadie lo tiene bloqueado a C



Luego vendría T2 leyendo el dato B, para ello solicita el bloqueo exclusivo del dato B y lo obtiene porque nadie lo tiene bloqueado a B



Luego tendríamos que solicitar el bloqueo del dato C, cuando se solicita el bloqueo T1 todavía lo tiene bloqueado por lo tanto no obtiene el bloqueo

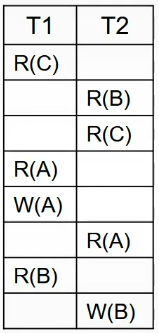


En el mundo real T2 no se va a poder ejecutar y se va a seguir ejecutando T1 con lo cual R(C) no se va a poder realizar y luego seguirá por T1 pero esto cambiaria la planificación.

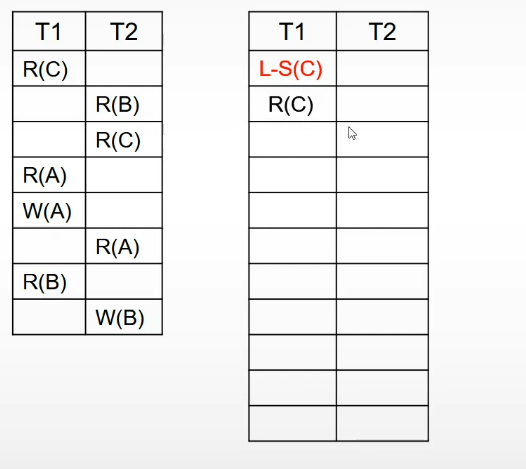
Tampoco podríamos liberar C porque entraríamos en la etapa de decrecimiento y no podríamos solicitar el bloqueo de A para leer A, y esto cambiaria la planificación.

Por lo tanto, la planificación no se puede obtener utilizando el bloqueo de 2 fases con solo locks exclusivos.

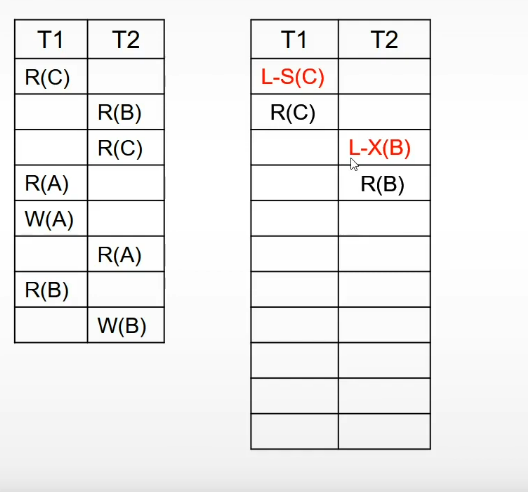
**Ejemplo** utilizando bloqueos **compartidos** Lock-S(A) o L-S(A) y **exclusivos** Lock-X(A) o L-X(A)



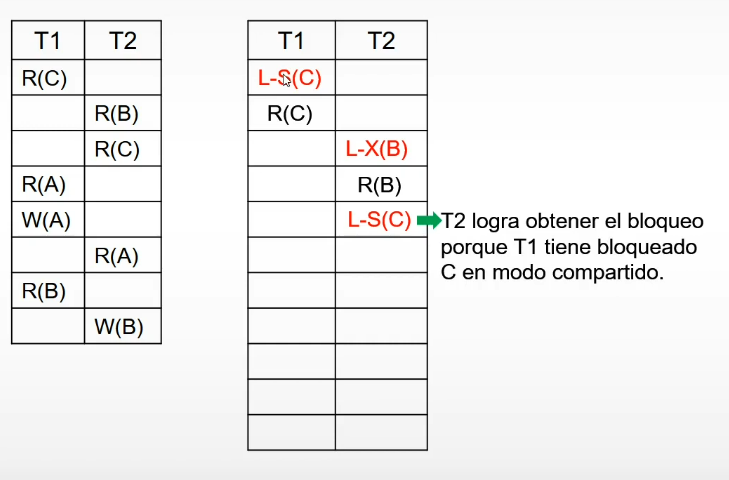
**Cuando tengamos que acceder para la lectura de un dato y no lo vamos a escribir en ninguna transacción podemos bloquearlo en modo compartido**, ejemplo T1 va a leer el dato C y luego no lo va a escribir, entonces lo bloqueamos en modo compartido y luego accedemos a C:



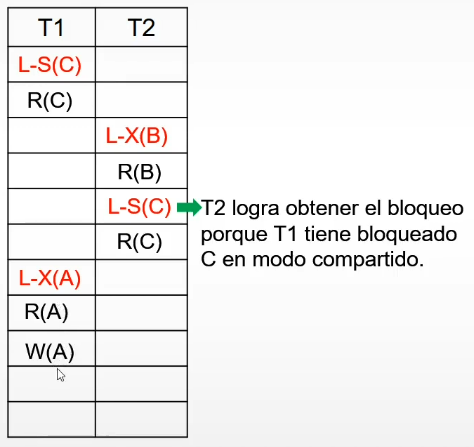
En T2 el dato C lo va a leer y luego lo va a escribir entonces lo solicitamos en modo exclusivo y lo leemos:



Luego solicitamos el dato C en modo compartido porque T2 solo lo va a leer y se logra obtener porque T1 lo tiene bloqueado en modo compartido y son modos compatibles porque 2 transacciones pueden leer simultáneamente el mismo dato

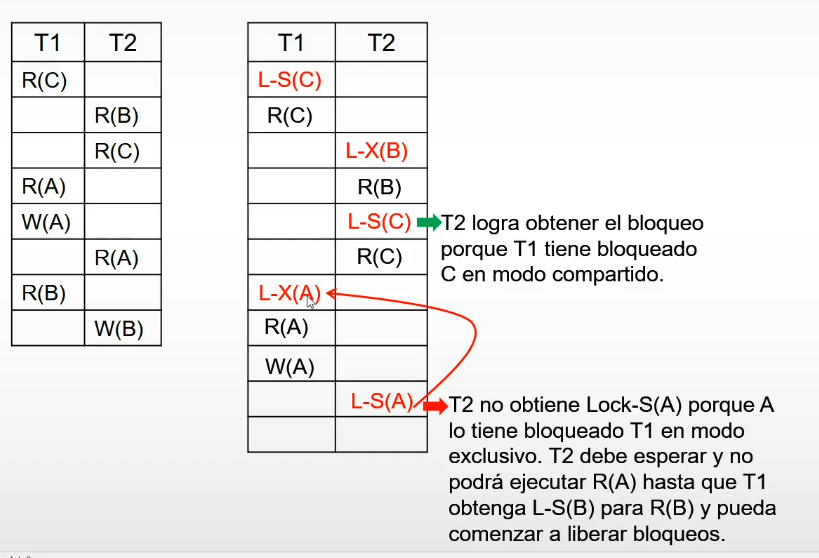


Luego como leemos y escribimos A en T1 entonces lo solicitamos en modo exclusivo, luego lo leemos y lo escribimos:

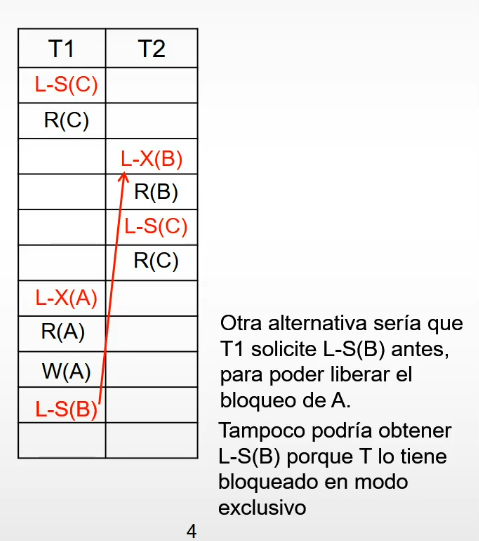


Ahora si liberaríamos el dato A entonces entraríamos en la etapa de liberación y no podríamos bloquear los datos que le siguen, por lo tanto, no liberamos a A.

Luego T2 solo necesita leer el dato A, solicitamos el bloqueo compartido de A pero no se puede obtener porque T1 lo tiene bloqueado en modo exclusivo



Se podría intentar otro camino pero que también resulta fallido, luego de escribir A en T1 intentamos bloquear B porque es el último dato a bloquear y ya podríamos empezar a liberar luego, pero no podemos porque B ya está bloqueado en modo exclusivo por T2:



**Ejemplo** utilizando bloqueos **compartidos**, **exclusivos**, **upgrade** Up(A), **downgrade** Do(a) y **unlock** Un(A).

**Upgrade**: Si tenemos un bloqueo en modo modo compartido al hacer un upgrade el bloqueo pasa a ser exclusivo

**Downgrade**: Si tenemos un bloqueo en modo exclusivo al hacer downgrade podemos pasarlo a compartido

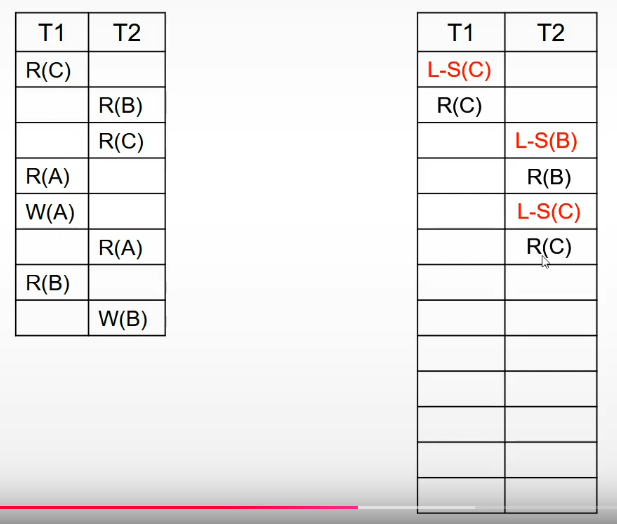
Solicitamos un bloqueo compartido de C y leemos C:



Ahora como tenemos la opción de upgrade, y vamos a leer y luego escribir B en T2 podemos pedir un bloqueo compartido y mas abajo hacer el upgrade. Tambien leemos B:



Solicitamos el bloqueo compartido de C y leemos C



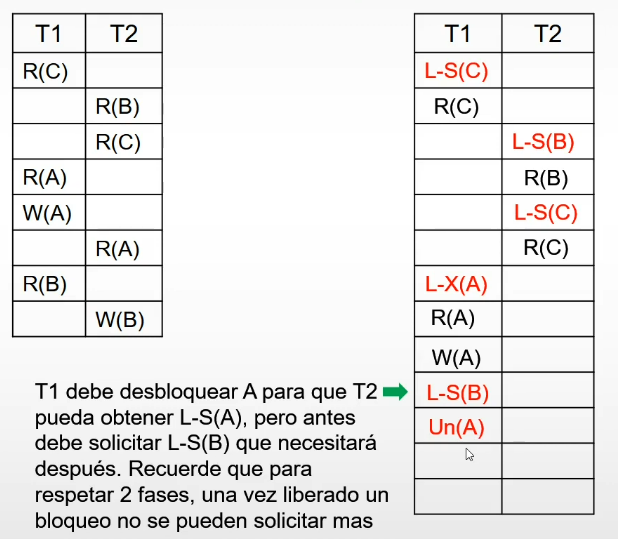
Solicitamos el bloqueo exclusivo de A para leer y escribir A, luego leemos y escribimos A:



Antes de desbloquear A pedimos el bloqueo de B porque si desbloqueamos a A antes entonces no podemos bloquear datos luego



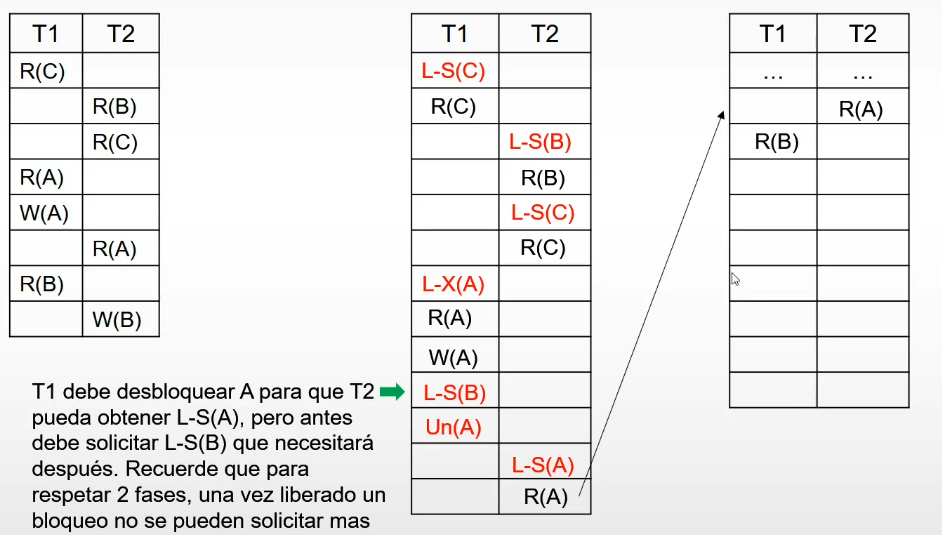
Ahora desbloqueamos A y T1 empieza en la fase de decrecimiento o sea T1 solo podrá liberar bloqueos o hacer downgrades:



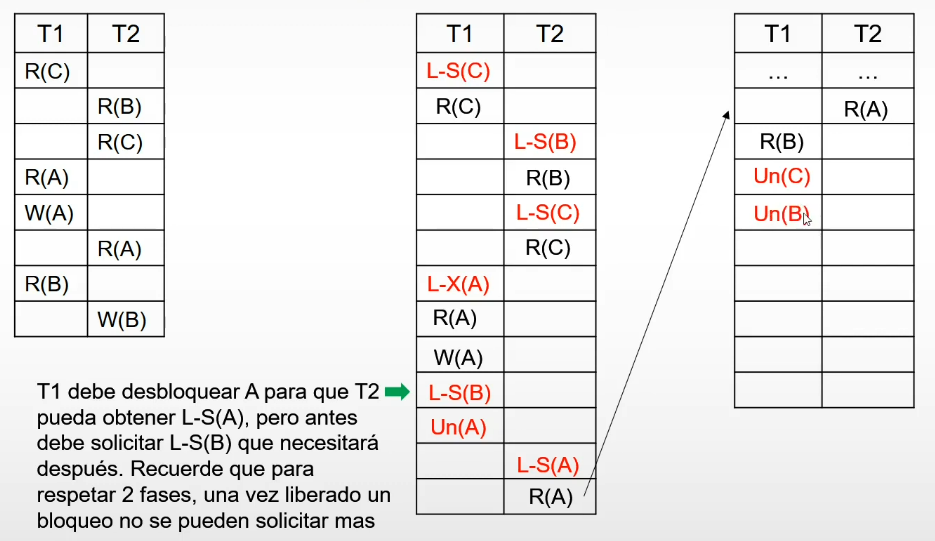
Esto permite que T2 pueda solicitar el bloqueo de A porque A esta liberado y pueda leer A:



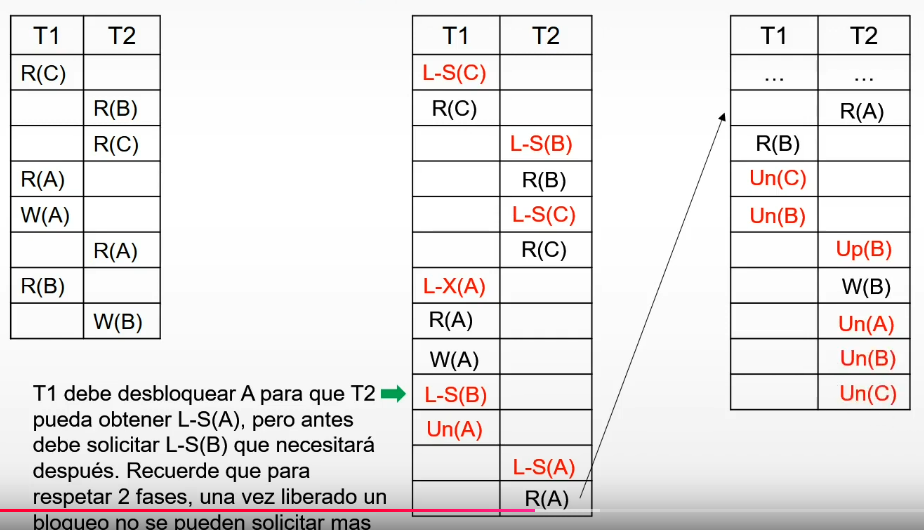
Como ya habíamos solicitado en modo compartido el dato B antes entonces simplemente leemos B:



Como T1 termina puede empezar a liberar datos los datos bloqueados (C y B)



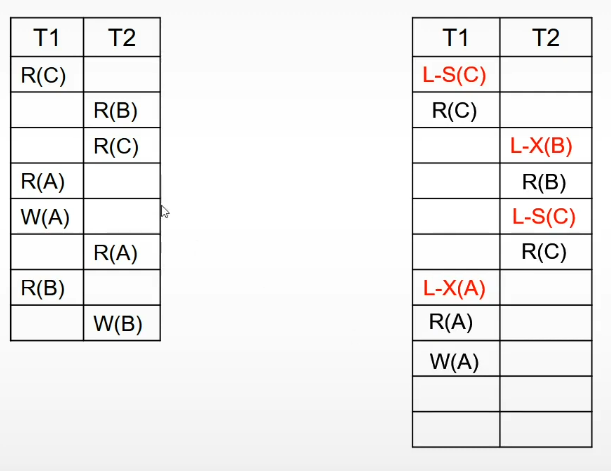
Y ahora para poder escribir en T2 el dato B necesitamos hacer un upgrade del dato B porque T2 lo tiene bloqueado en modo compartido, esto es posible porque T1 libero el dato B antes, luego escribimos B y liberamos los datos.



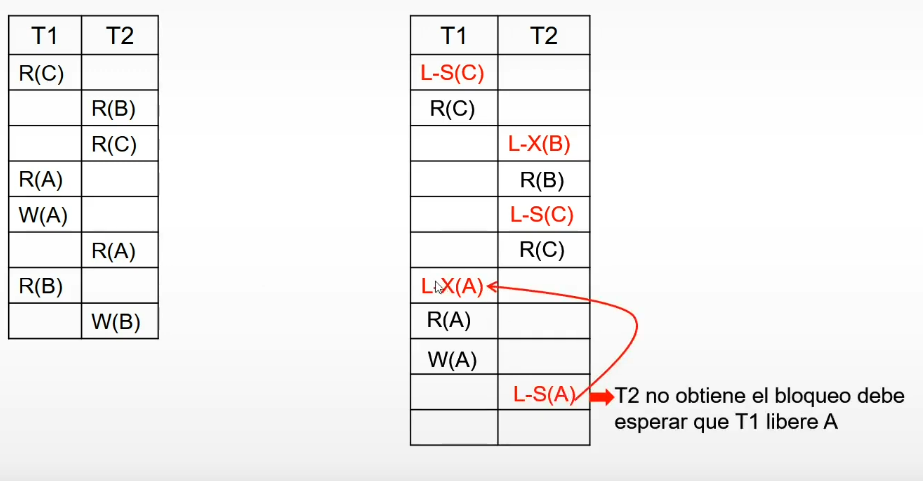
Siempre chequear que ambas transacciones respeten las 2 fases o sea para cada transacción que no pidan un bloqueo luego de haber liberado un bloqueo

Ejemplo de deadlock

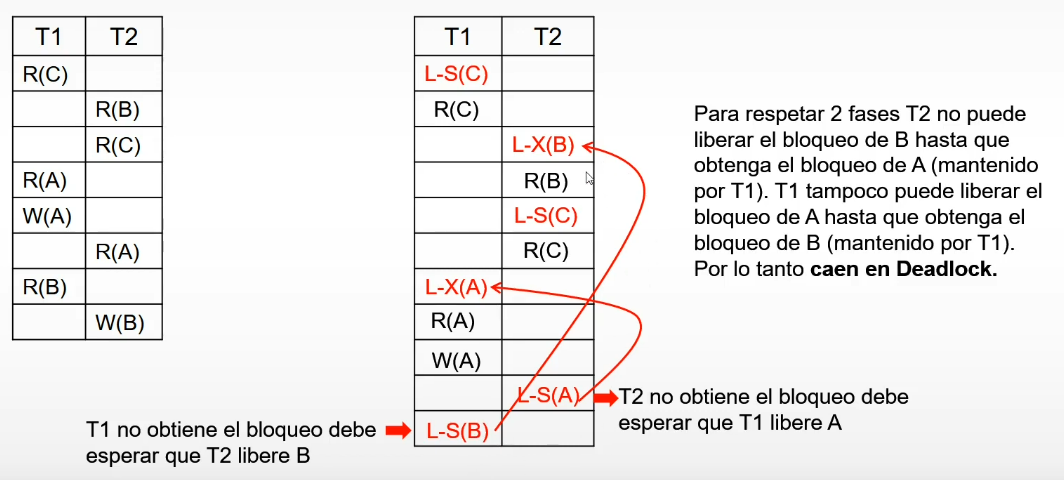
Bloqueamos como venimos haciendo:



T2 va a querer bloquear en modo compartido a A pero no puede porque T1 lo tiene bloqueado en modo exclusivo



Luego T1 tiene que leer B y solicita bloqueo compartido de B pero no lo obtiene porque T2 lo tiene bloqueado en modo exclusivo



**Protocolo de Arbol**

Un árbol define el ordenamiento de los bloqueos para todos los datos del sistema, siguiendo las siguientes reglas:

1. El único **bloqueo** permitido es el **exclusivo**: L-X(A)
2. **El primer bloqueo de cada transacción puede ser sobre cualquier dato.**
3. **Después de su primer bloqueo una transacción puede bloquear un dato Q si tiene bloqueado al padre de Q en el árbol**
   1. El 1er dato R que bloquea una transacción restringe los datos que podrá bloquear a lo largo de su ejecución. Solo podrá bloquear a los datos descendientes en el sub-árbol que tenga el dato R como raíz, avanzando de padres a hijos.
4. Cuando **una transacción desbloquea un dato Q (Un(Q)) no puede volver a bloquear a Q.**
5. Los datos pueden desbloquearse en cualquier momento, es decir**, puede no respetar 2 fases.**

**Ejemplo:**

Analizaremos 2 planificaciones distintas con las mismas transacciones T1 y T2 que utilizamos para el protocolo de 2 fases. Verificaremos si es posible obtener las planificaciones considerando el árbol de la derecha:



Como T2 empieza solicitando el dato B tendríamos que solicitar el bloqueo de B, pero el 1er bloqueo restringe hacia donde vamos a acceder en el árbol, entonces si leemos B y bloqueamos B después solo podemos acceder a A pero no a C.

**Como T2 accede a los datos B, C y A entonces no queda otra que bloquear la raíz para poder acceder a B y A.**

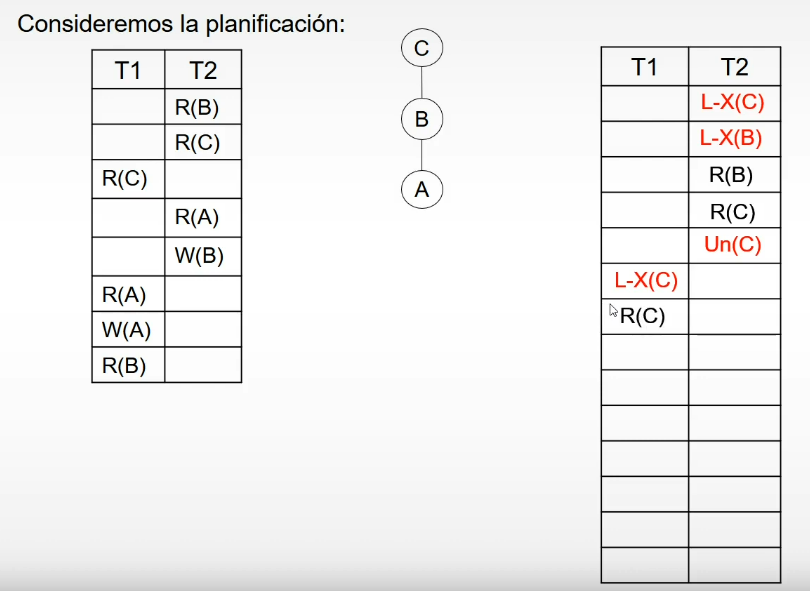
**Entonces bloqueamos a C, ahora podemos acceder al hijo y por ello bloqueamos a B.** Ahora podemos leer C y B:



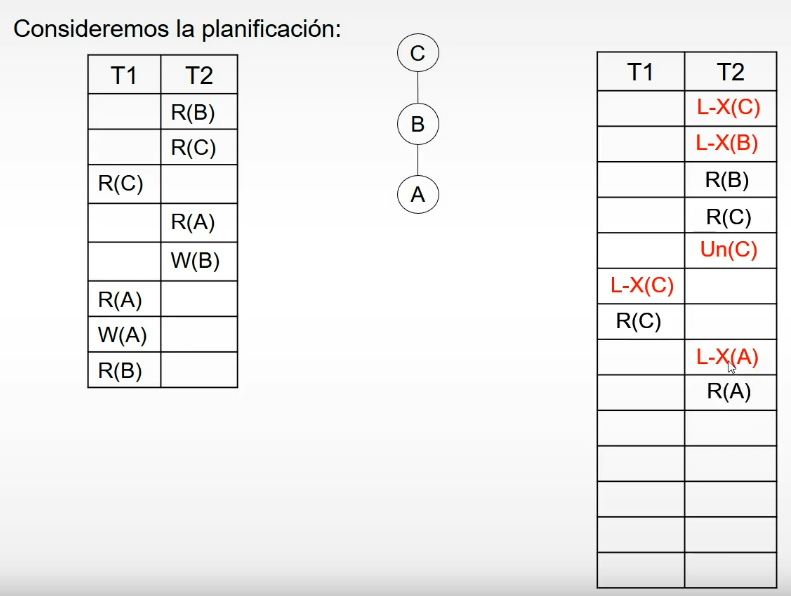
Para que T1 pueda leer C tendría q bloquear C, entonces habría que liberarlo para que T1 lo pueda bloquear.

Acá no hay que respetar las 2 fases, podemos desbloquear C siempre que luego podamos acceder a A (o sea tengamos bloqueado al padre de A).

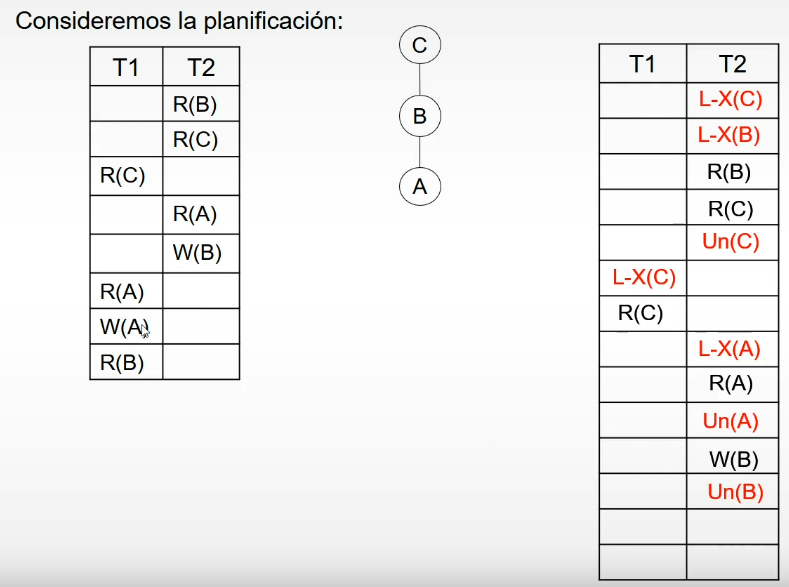
Ya podemos liberar C porque ya lo leímos y no lo volveremos a utilizar en la planificación.



En T2 pasamos a leer A, lo bloqueamos exclusivamente y esto lo podemos hacer porque tenemos bloqueado el dato B (el padre de A) y podrá leer A.



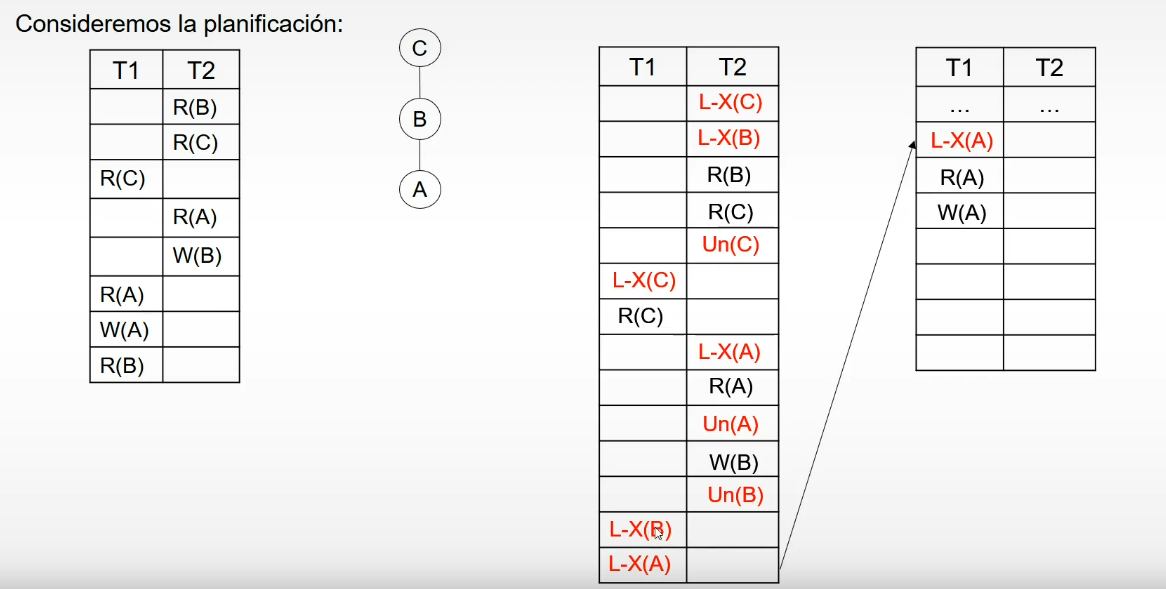
Podemos desbloquear A porque no lo vamos a usar en T2 y podemos escribir B porque mantenemos el bloqueo de B y podemos desbloquear B porque no necesitamos más datos.



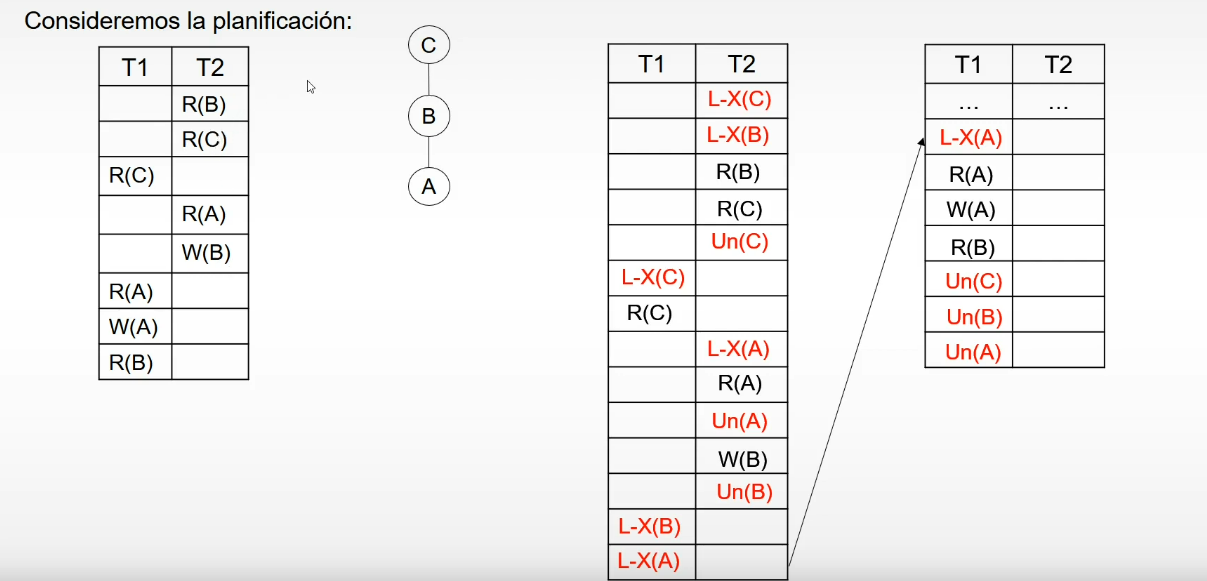
Necesitamos solicitar A pero para solicitarlo necesitamos bloquear al padre (B).

Entonces bloqueamos B que lo podemos hacer porque tenemos bloqueado C en T1.

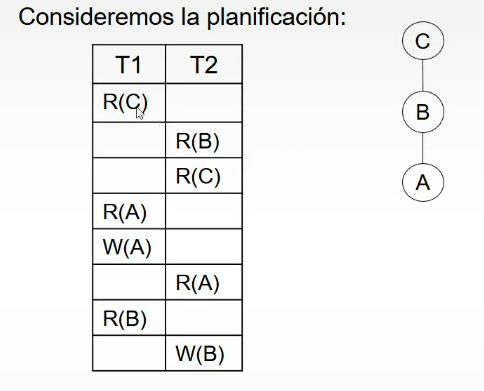
Como tenemos el bloqueo de B podemos bloquear A, y leer y escribir A.



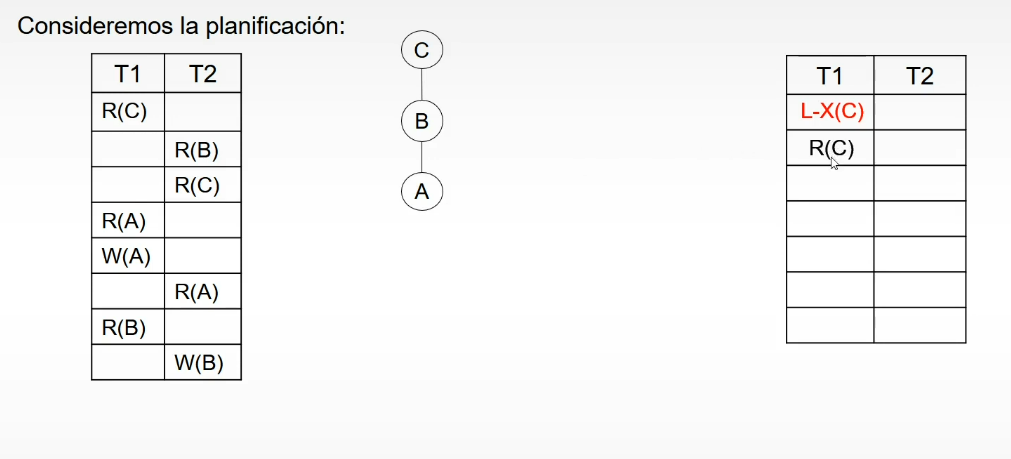
Como tenemos bloqueado B podemos leer B y como termina la transacción desbloqueamos todos los datos.



Ejemplo de lo anterior con otra planificación

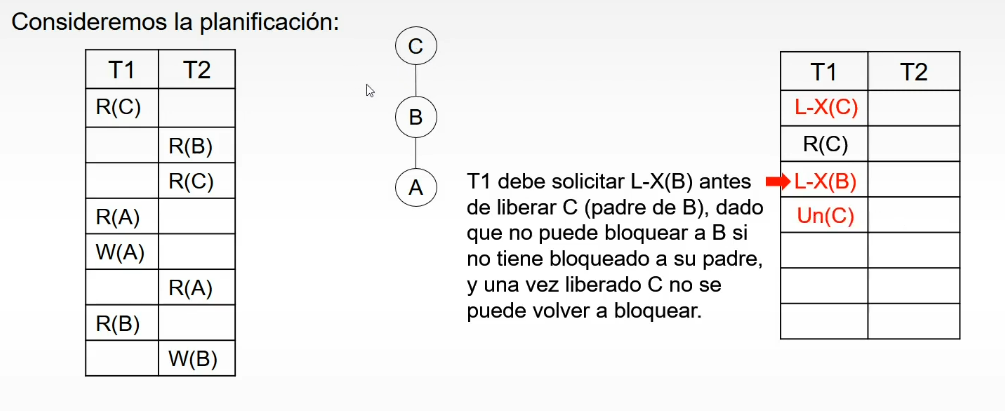


T1 va a acceder a C, A y B entonces solicitamos el bloqueo de C para poder acceder luego a B y A y leemos C.



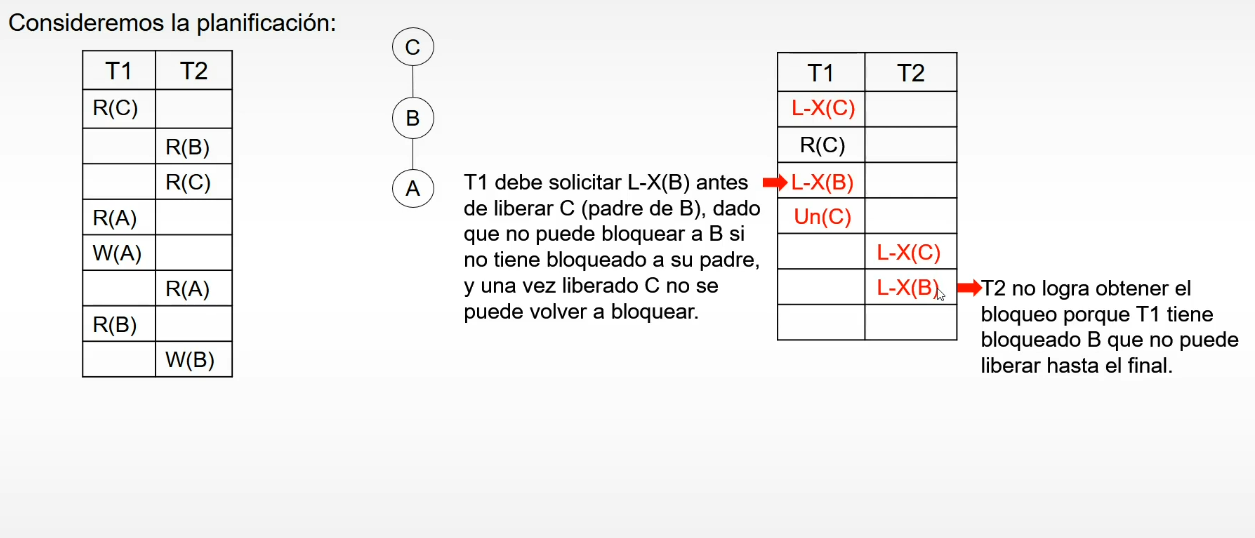
Tendríamos q pasar a la ejecución de T2 pero para que T2 acceda a C, T1 tendría que liberar a C.

Si liberamos a C en T1 tendríamos que antes bloquear a B porque si no, no podríamos acceder a nada luego. Luego desbloqueamos C.



Ahora T2 puede bloquear C y a partir de ahí acceder al resto de datos.

Intentamos acceder a B pero T1 tiene bloqueado exclusivamente a B



No se puede dar la planificación en el protocolo de árbol.

1. Una forma de asegurar la serializabilidad es exigir que el acceso a los datos se haga de modo mutuamente excluyente.

**Tipos de bloqueo:**

**Compartido (Lock-S):** si una transacción T ha obtenido un bloqueo compartido sobre un dato Q entonces **T puede leer** el dato, **pero no escribir Q.**

**Exclusivo (Lock-X):** si una transacción T ha obtenido un bloqueo exclusivo sobre un dato Q entonces **T puede leer y escribir Q.**