Bases de Datos 1

Control de Concurrencia

Concurrencia

- Múltiples usuarios tienen acceso simultáneo a la base de datos.
- Problema cuando: dos o más transacciones requieren simultáneamente el mismo objeto de la base de datos.

Transacción

- Una transacción es la ejecución de un programa que accede a una base de datos, la cual es compartida simultáneamente por varios usuarios.
- Secuencia de operaciones sobre un grupo de objetos de la base de datos.
- Ejemplo:
  - Aplicar una comisión del 2% sobre el monto de ventas a todos los agentes vendedores que hayan generado ventas superiores a los 4millones durante el mes de enero.

Transacción

- Operaciones sobre los objetos de la base de datos:
  - Begin transaction.
  - Read objet.
  - Write objet.
  - Rollback.
  - Commit.
- La transacción debe dejar la base de datos en un estado consistente.

Transacción - Propiedades

- Atomicidad: La transacción se realiza completamente o no se lleva cabo del todo. Es todo o nada.
- Consistencia: La transacción hace que la base de datos cambie inicialmente de estado, pero debe dejarla en un estado consistente.
- Independencia: Cada transacción se ejecuta independientemente de las demás. Como si fuese única.
- Durabilidad: Los efectos de una transacción que llegó a su fin (commit realizado) deben ser permanentes en la base de datos.

Tipos de bloqueo

Binarios

Modo múltiple (Compartidos/Exclusivos o de Lectura/Escritura)

¿Qué es un bloqueo?

- Medio para sincronizar el acceso de las transacciones concurrentes a los elementos de la BD
- Tienen naturaleza y tipo de bloqueo
- Se usan para garantizar la serialización de la planificación de las transacciones

Bloqueo Binario

- Desbloqueado

- Bloqueado

Bloqueos

- Las operaciones de bloquear(X) y desbloquear(X) son indivisibles (sesiones críticas en los sistemas operativos)

¿Cómo se implementaría un bloqueo binario?

- Tabla de bloqueo
  - Nombre del elemento de datos
  - Bloquear
  - Transacción de bloqueo

- Cola con las transacciones que están en espera de acceder el elemento

¿Qué reglas deberían cumplir las transacciones entonces?

1. Una transacción debe primero emitir la operación bloquear(X) antes de ejecutar cualquier operación de leer_elemento(X) o escribir_elemento(X)
2. Desbloquear_elemento(X) después de completar todas las operaciones de la transacción T
3. No bloquear_elemento(X) si ya lo tiene bloqueado
4. No desbloquear(X) a menos que lo tenga bloqueado
5. A lo sumo una transacción debe puede poseer el bloqueo de un elemento

- ¿Dónde están implementadas las reglas?
- En el SABD.

Bloqueos compartidos/exclusivos

Valores iniciales x = 20, y = 30

Tiempo	Transaction T1	Transaction T2	Resultado para T1 – T2	Resultado para T2 – T1
T1	Bloquear_lectura(Y)	Bloquear_lectura(X)		
T2	Leer_elemento(Y)	Leer_elemento(X)		
T3	Desbloquear(Y)	Desbloquear(X)		
T4	Bloquear_escritura(X)	Bloquear_escritura(Y)		
T5	Leer_elemento(X)	Leer_elemento(Y)		
T6	X:=X+Y	Y:=X+Y		
T7	Escribir_elemento(X)	Escribir_elemento(Y)		
T8	Desbloquear(X)	Desbloquear(Y)	X=50 Y=80	X=70 Y=50

Bloqueos compartidos/exclusivos

Tiempo	Transaction T1	Transaction T2
T1	Bloquear_lectura(Y)	
T2	Leer_elemento(Y)	
T3	Desbloquear(Y)	
T4		Bloquear_lectura(X)
T5		Leer_elemento(X)
T6		Desbloquear(X)
T7		Bloquear_escritura(Y)
T8		Leer_elemento(Y)
T9		Y:=X+Y
T10		Escribir_elemento(Y)
T11		Desbloquear(Y)
T12	Bloquear_escritura(X)	
T13	Leer_elemento(X)	
T14	X:=X+Y	
T15	Escribir_elemento(X)	
T16	Desbloquear(X)	

X=50
Y=50
No serial

Mecanismos para controlar la concurrencia

Protocolo de dos fases

Protocolo de dos fases

- Bloquear_lectura, bloquear_escritura antes de la primera operación de desbloqueo
- Fase de expansión o crecimiento
- Fase de reducción
- Promoción de bloqueos
- Degradación de los bloqueos

Mecanismos para controlar la concurrencia

- Protocolo de dos fases.
- Compatibilidad de los cerrojos:

Cerrojo	s-lock	x-lock
s-lock	Compatible	Conflicto
x-lock	Conflicto	Conflicto

Mecanismos para controlar la concurrencia

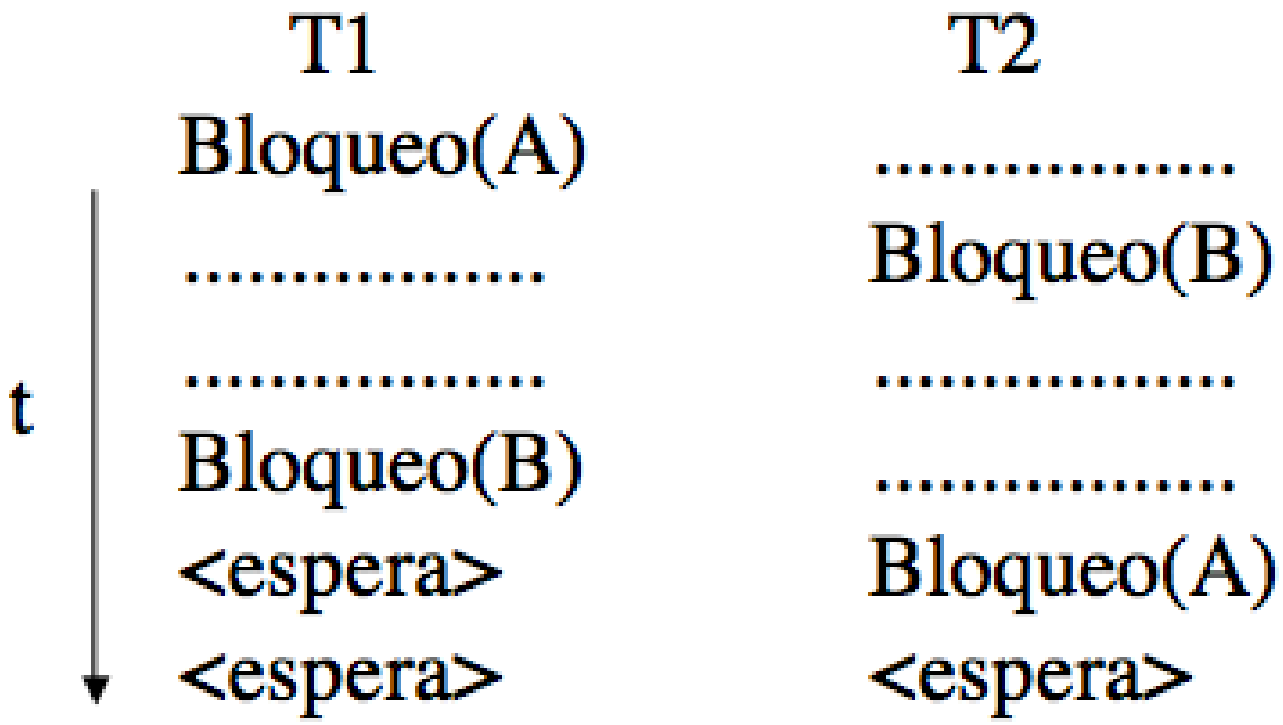
- Protocolo de dos fases.
- Dos transacciones no pueden tener cerrojos en conflicto en forma simultánea.
- Toda transacción que libera un cerrojo no puede adquirir otro.
- Fase de expansión: cuando se adquieren los cerrojos.
- Fase de reducción: se liberan los cerrojos.

Interbloqueo

- Se produce cuando cada transacción T en un conjunto de 2 o más transacciones está esperando a algún elemento que está bloqueado por alguna otra transacción T de dicho conjunto

Tiempo	T1	T2
T1	Bloquear_lectura(Y)	
T2	Leer_elemento(Y)	
T3		Bloquear_lectura(X)
T4		Leer_elemento(X)
T5	Bloquear_escritura(X)	
T6		Bloquear_escritura(Y)

Interbloqueo



Detección del interbloqueo

- Es una metodología
- Abortar la o las transacciones que generan el interbloqueo

Detección del interbloqueo

Gráfico de espera

Tiempos limitados

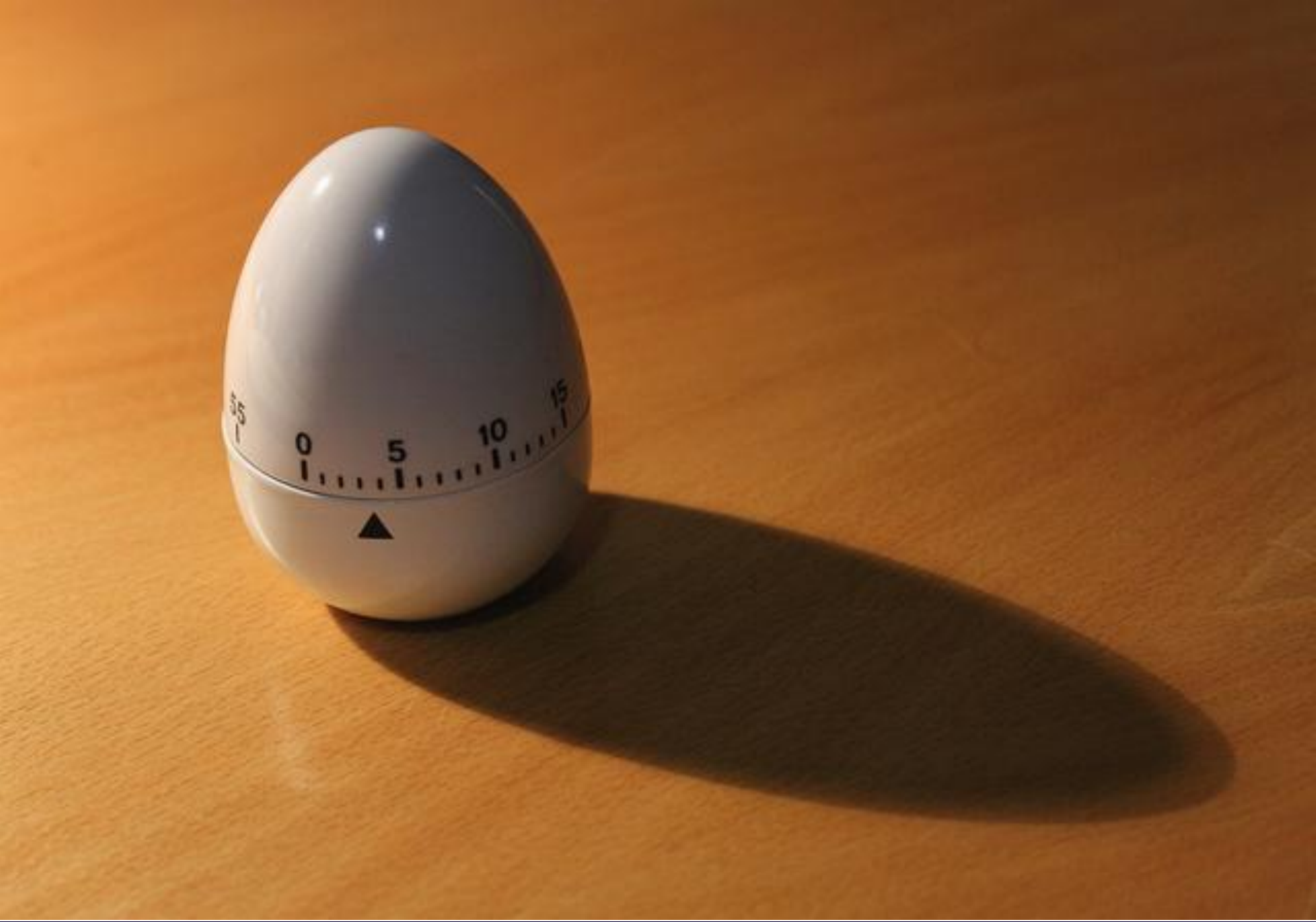
Inanición

Estrategia Gráfico de espera

- Nodo por transacción que se está ejecutando
- Transacción Ti que está esperando a bloquear un elemento X que actualmente está bloqueado por una transacción Tj
- En el gráfico (grafo) se crea un arco de Ti --> Tj
- Cuando Tj libera el bloqueo sobre el elemento X, el arco desaparece
- ¿Cuándo se identifica un interbloqueo en el gráfico de espera?
- ¿Problema?
- ¿Criterios para identificar interbloqueos?
- Algoritmo de selección de la víctima
  - Evitar la selección de transacciones que han estado ejecutándose por mucho tiempo o que han hecho muchas actualizaciones

Tiempos limitados

- Si una transacción lleva en espera un tiempo superior al tiempo definido por el sistema, se asume que la transacción puede estar bloqueada y se elimina (independientemente de si existe o no un bloqueo


55
0
5
10
15

Inanición

- Problema que surje a veces con el bloqueo cuando el esquema de espera para los elementos bloqueados es injusto
- Prioridades a transacciones no bien asignadas

Ordenación de marcas de tiempo

Marcas de tiempo

- Planificación de transacciones equivalente
- No usa bloqueos, por lo tanto, no hay interbloqueos

Marcas de tiempo

- Identificador único creado por el DBMS para identificar una transacción
- Normalmente, los valores son asignados en el orden en el que las transacciones son enviadas al sistema

Marcas de tiempo

- Un contador
- El valor de fecha/hora actual del sistema

Algoritmo

- Las marcas temporales de las transacciones determinan el orden de secuencia
- De este modo, si MT(*Ti*) < MT(*Tj*) entonces el sistema debe asegurar que toda planificación que produzca es equivalente a una planificación secuencial en la cual la transacción *Ti* aparece antes que la transacción *Tj*

Algoritmo

Para implementar este esquema se asocia a cada elemento de datos *Q* dos valores de marca temporal:

- marca_temporal-E(*Q*) denota la mayor marca temporal de todas las transacciones que ejecutan con éxito escribir(*Q*).
- marca_temporal-L(*Q*) denota la mayor marca temporal de todas las transacciones que ejecutan con éxito leer(*Q*)

Estas marcas temporales se actualizan cada vez que se ejecuta una nueva operación leer(Q) o escribir(Q)

Algoritmo

- El protocolo de ordenación por marcas temporales asegura que todas las operaciones leer y escribir conflictivas se ejecutan en el orden de las marcas temporales. Este protocolo opera como sigue:

Supóngase que la transacción Ti ejecuta leer(Q)

- Si MT(*Ti*) < marca_temporal-E(*Q*) entonces *Ti* necesita leer un valor de *Q* que ya se ha sobrescrito. Por tanto se rechaza la operación leer y
  Ti se retrocede.

- Si MT(*Ti*) ≥ marca_temporal-E(*Q*) entonces se ejecuta la operación leer y marca_temporalL(*Q*) se asigna al máximo de marca_temporalL(*Q*) y de MT(*Ti*).

Supóngase que la transacción Ti ejecuta escribir(Q)

- Si MT(*Ti*) < marca_temporal-L(*Q*) : se rechaza la operación escribir y *Ti* se retrocede.
- Si MT(*Ti*) < marca_temporal-E(*Q*) : se rechaza la operación escribir y *Ti* se retrocede.
- En otro caso se ejecuta la operación escribir y MT(*Ti*) se asigna a marca_temporal-E(*Q*)

Ventajas

- El protocolo de ordenación por marcas temporales asegura la secuencialidad en cuanto a conflictos. Esta afirmación se deduce del hecho de que las operaciones conflictivas se procesan durante la ordenación de las marcas temporales.
- El protocolo asegura la ausencia de interbloqueos, ya que ninguna transacción tiene que esperar. Sin embargo, existe una posibilidad de inanición de las transacciones largas si una secuencia de transacciones cortas conflictivas provoca reinicios repetidos de la transacción larga.
- Si se descubre que una transacción se está reiniciando de forma repetida, es necesario bloquear las transacciones conflictivas de forma temporal para permitir que la transacción termine

Regla de escritura de Thomas

Supóngase que la transacción *Ti* ejecuta escribir(*Q*).

- 1. Si MT(*Ti*) < marca_temporal-L(*Q*), entonces el valor de *Q* que produce *Ti* se necesita previamente y el sistema asume que dicho valor no se puede producir nunca. Por tanto, se rechaza la operación escribir y *Ti* se retrocede.
- 2. Si MT(*Ti*) < marca_temporal-E(*Q*), entonces *Ti* está intentando escribir un valor de *Q* obsoleto. Por tanto se puede ignorar dicha operación escribir.

- 3. En otro caso, se ejecuta la operación escribir y MT(*Ti*) se asigna a marca_temporal-E(*Q*).

Regla de escritura de Thomas

- La regla de escritura de Thomas hace uso de la secuencialidad en cuanto a vistas borrando las operaciones escribir obsoletas de las transacciones que la ejecutan. Esta modificación de las transacciones hace posible generar planificaciones que no serían posibles con otros protocolos

Conflictos entre transacciones concurrentes

- Interés sobre un objeto de la base de datos: para lectura o escritura.
- Si dos transacciones T1 y T2 se interesan en el mismo objeto, al mismo tiempo:
  - T1: lectura , T2: lectura --> Ok, no hay conflicto.
  - T1: escritura, T2: escritura --> conflicto.
  - T1: escritura, T2: lectura --> conflicto.
  - T1: lectura, T2: escritura --> conflicto.

Los problemas que se pueden presentar

- La concurrencia puede provocar los siguientes problemas:
  - Actualizaciones perdidas.
  - Dependencias no confirmadas.
  - Análisis inconsistente.

El problema de la actualización perdida

- La Transacción A recupera la tupla t en el tiempo t1.
- La Transacción B recupera la misma tupla t en el tiempo t2.
- La Transacción A actualiza la tupla t en el tiempo t3.
- La Transacción B actualiza la misma tupla t en el tiempo t4.
- La actualización de la Transacción A se pierde en t4.

El problema de la dependencia no confirmada

- Una Transacción recupera o actualiza una tupla actualizada por otra Transacción (aún no finalizada).
- La Transacción B actualiza la tupla t en el tiempo t1.
- La Transacción A recupera la tupla t en el tiempo t2.
- La Transacción B deshace la actualización de la tupla t en el tiempo t3 (undo).
- La Transacción A está operando bajo suposición falsa.

El problema del análisis inconsistente

- La Transacción A acumula saldos de 3 cuentas (c1, c2, c3).
- La Transacción B transfiere $10 de c3 a c1 y se confirma, antes de que A acumule el saldo de la cuenta c3.
- Si A finaliza con éxito, produce inconsistencia en la BD.

Conflictos entre transacciones concurrentes

- Escritura – Escritura.
- Alguna de las actualizaciones se podría perder.
- Ejemplo:

Tiempo	Transaction T1	Transaction T2	Valor real	Valor esperado
T1		Begin T2		
T2	Begin T1			
T3		Read X	X=3000	X=3000
T4	Read X		X=3000	X=3000
T5		X=X+2000		
T6	X=X+5000			
T7		Write X	X=5000	X=5000
T8		Commit T2		
T9	Write X		X=8000	X=10000

Conflictos entre transacciones concurrentes

- Escritura – Lectura.
- Lectura indebida de un objeto:
  - Una transacción no debe leer ni modificar un objeto que ha sido modificado por otra transacción que aún no ha ejecutado el commit.
- Ejemplo:

Conflictos entre transacciones concurrentes

Tiempo	Transaction T1	Transaction T2	Valor real	Valor esperado
T1	Begin T1			
T2		Begin T2		
T3	Read X	Read X	X=1000	X=1000
T4	X=X+2000			
T5	Write X		X=3000	X=3000
T6		Read X	X=3000	X=3000
T7		Read Y	Y=5000	Y=5000
T8		Y=X+1000		
T9		Write Y	Y=4000	Y=4000
T10	Abort T1			

Conflictos entre transacciones concurrentes

- Lectura – Escritura.

Tiempo	Transaction T1	Transaction T2	Valor real	Valor esperado
T1	Begin T1			
T2	Read X		X=1000	X=1000 (T1)
T3		Begin T2		
T4		Read X	X=1000	X=1000 (T2)
T5	X=X+2000			
T6	Write X		X=3000	X=3000 (T1)
T7	Commit T1			
T8		Read X	X=3000	X=1000 (T2)

Itinerario

- Cuando dos o más transacciones se realizan concurrentemente.
- Scheduler es el programa que controla la ejecución concurrente.
- Acciones: ejecutar, rechazar, dejar en espera la operación.
- Se busca itinerarios consistentes: equivalentes a uno secuencial.

Itinerario

- Ejemplo:
- T1: Read a – Write a
- T2: Read b – Write b

I1	I2	I3	I4	I5	I6
Read a	Read b	Read a	Read b	Read b	**Read b**
Write a	Write b	Read b	Read a	Read a	**Read b**
Read b	Read a	Write a	Write a	Write b	**Write b**
Write b	Write a	Write b	Write b	Write a	**Write a**

Itenerario

- Ejemplo:
- T1: Read a – Write a
- T2: Read b – Write b

I1	I2	I3	I4	I5	I6
Read a	Read b	Read a	Read b	Read b	**Read b**
Write a	Write b	Read b	Read a	Read a	**Read b**
Read b	Read a	Write a	Write a	Write b	**Write b**
Write b	Write a	Write b	Write b	Write a	**Write a**

Itenerarios no secuenciales

¿Cuáles son consistentes?

I_1		Valor real	I_2		Valor real	I_3		Valor real
T1:	Begin		T1:	Begin		T1:	Begin	
T1:	Read A	A=2000	T1:	Read A	A=2000	T1:	Read A	A=2000
T1:	Read B	B=3000	T2:	Begin		T2:	Begin	
T1:	A= A -1000		T2:	Read B	B= 3000	T1:	Read B	B=3000
T1:	B=B+1000		T2:	Read C	C=4000	T2:	Read C	C=4000
T1:	Write A	A=1000	T1:	A= A -1000		T1:	A= A -1000	
T1:	Write B	B=4000	T1:	Write A	A=1000	T2:	Read B	B=3000
T1:	Commit		T2:	B=B+1000		T1:	B=B+1000	
T2:	Begin		T2:	C=C-1000		T2:	C=C-1000	
T2:	Read C	C=4000	T2:	Write B	B=4000	T1:	Write A	A=1000
T2:	Read B	B=4000	T2:	Write C	C=3000	T2:	B=B+1000	
T2:	C=C-1000		T1:	Read B	B=4000	T1:	Write B	B=4000
T2:	B=B+1000		T2:	Commit		T2:	Write C	C=3000
T2:	Write C	C=3000	T1:	B=B+1000		T1:	Commit	
T2:	Write B	B=5000	T1:	Write B	B=5000	T2:	Write B	B =4000
T2:	Commit		T1:	Commit		T2:	Commit	

Mecanismos para controlar la concurrencia

- Uso de cerrojos (locks).
- Protocolo de exclusión mutua.
- Ninguna transacción puede efectuar una actualización o lectura sobre un objeto de la BD si no adquiere previamente un cerrojo sobre él.
- Si una transacción no puede adquirir un cerrojo de un objeto (porque otra lo está utilizando) deberá esperar a que la primera transacción lo libere.

Mecanismos para controlar la concurrencia

Tiempo	Transaction T1	Transaction T2	Valor real	Valor esperado
T1	Begin T1			
T2	Read X		X=1000	X=1000 (T1)
T3		Begin T2		
T4		Read X	X=1000	X=1000 (T2)
T5	X=X+2000			
T6	Write X		X=3000	X=3000 (T1)
T7	Commit T1			
T8		Read X	X=3000	X=1000 (T2)

Mecanismos para controlar la concurrencia

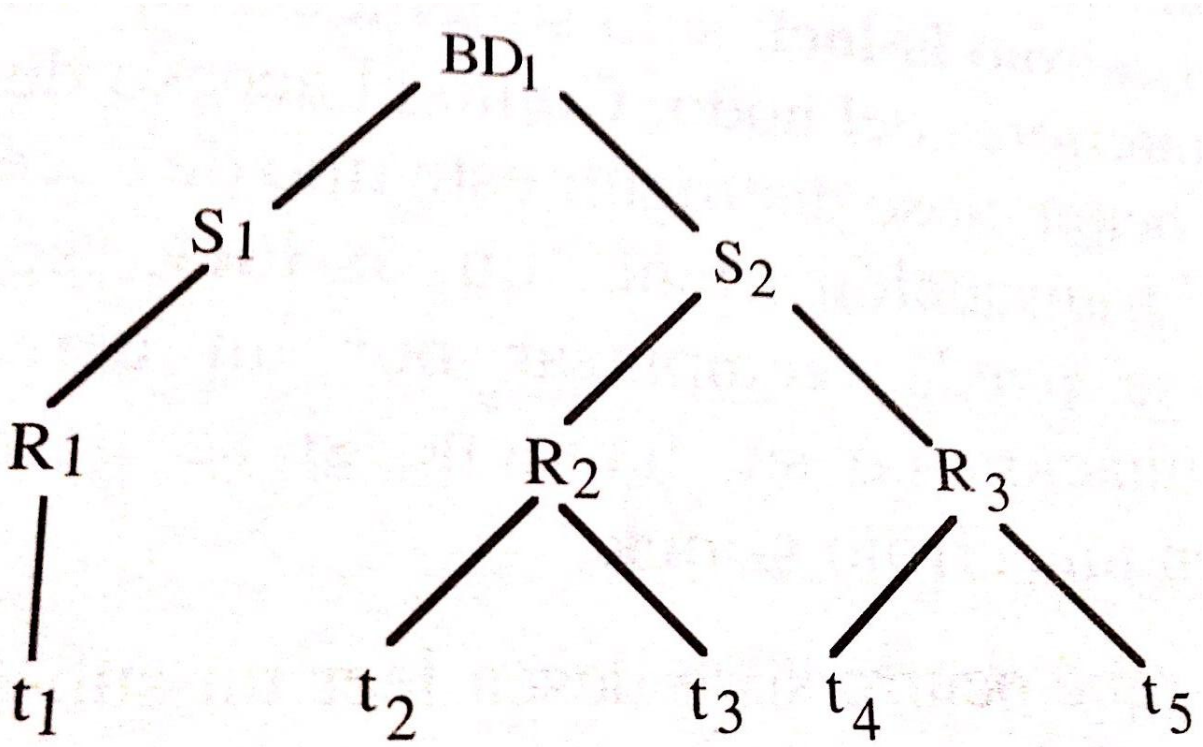
- Protocolo de exclusión mutua.
- Se puede presentar un interbloqueo (deadlock): espera indefinida.
- Una forma de resolverlo: uso de estampillas de tiempo:
  - Con decomiso (wait – die).
  - Sin decomiso (wound wait) .
- Protocolos nuevos que eviten el interbloqueo (ejemplo: protocolo de árbol).

Interbloqueo

- ¿Cómo detectarlo?
- Grafo de dependencias:
  - Un nodo por transacción.
  - Si Ti está esperando un recurso/objeto que tiene bloqueado Tj --> hay un arco entre Ti y Tj.
  - Si hay un ciclo en el grafo --> hay interbloqueo.

Mecanismos para controlar la concurrencia

- Protocolo de granularidad múltiple.
- Establecer el nivel del objeto que se quiere bloquear (desde un atributo hasta la base de datos completa).



Mecanismos para controlar la concurrencia

- Protocolo de granularidad múltiple.
- Establecer el nivel del objeto que se quiere bloquear (desde un atributo hasta la base de datos completa).
- Cerrojo de intención: marca los ancestros de un nodo al que se quiere aplicar un cerrojo (exclusivo o compartido):
  - Cerrojo de lectura en intención (is-lock).
  - Cerrojo de escritura en intención (ix-lock).

Mecanismos para controlar la concurrencia

- Protocolo de granularidad múltiple.
- El cerrojo de lectura en intención (is-lock) solicita cerrojos s-lock e is-lock en los niveles inferiores del nodo.

Multiversión

- Utiliza marcas de tiempo.
- Varias transacciones leen y escriben diferentes versiones del mismo dato.
- El control de concurrencia es de varias versiones de un dato.
- Cuando una transacción quiere acceder a un dato, se compara su marca de tiempo con las marcas de tiempo de las versiones del dato.

Mecanismos para controlar la concurrencia

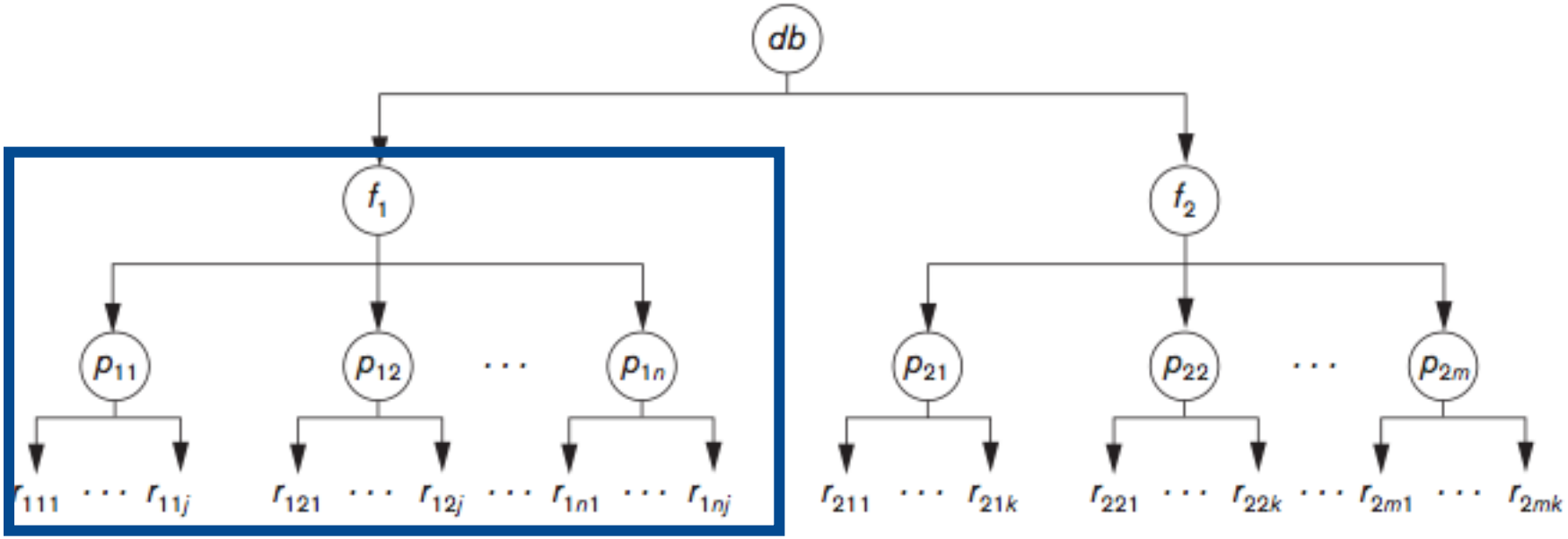
- Protocolo de árbol.
- Asegura la ausencia de interbloqueos.
- Desbloquea en cualquier momento (tiempos de espera menores).
- Sin embargo:
  - Puede bloquear elementos a los que no accede (bloqueos adicionales, + tiempo de espera).
  - Bloquear la raíz del árbol.

Mecanismos para controlar la concurrencia

- Protocolo de granularidad múltiple.
- Granularidad fina: parte inferior del árbol alta concurrencia y bloqueos.
- Granularidad gruesa: niveles superiores del árbol pocos bloqueos y baja concurrencia.

Mecanismos para controlar la concurrencia

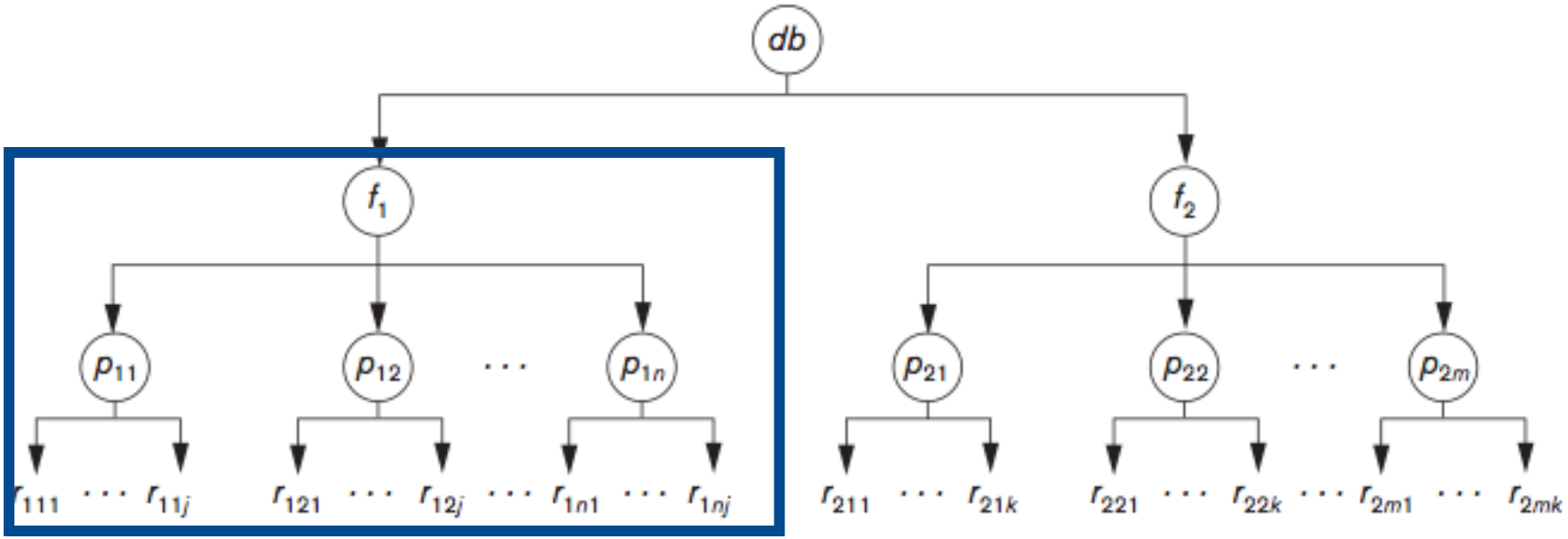
- Protocolo de granularidad múltiple.



Suponga que T1 requiere actualizar todos los registros del archivo f1 y obtiene un cerrojo exclusivo sobre f1.

Mecanismos para controlar la concurrencia

- Luego, T_2 requiere leer algunos registros de la página p1n del archivo f1 y solicita un s-lock sobre ellos:



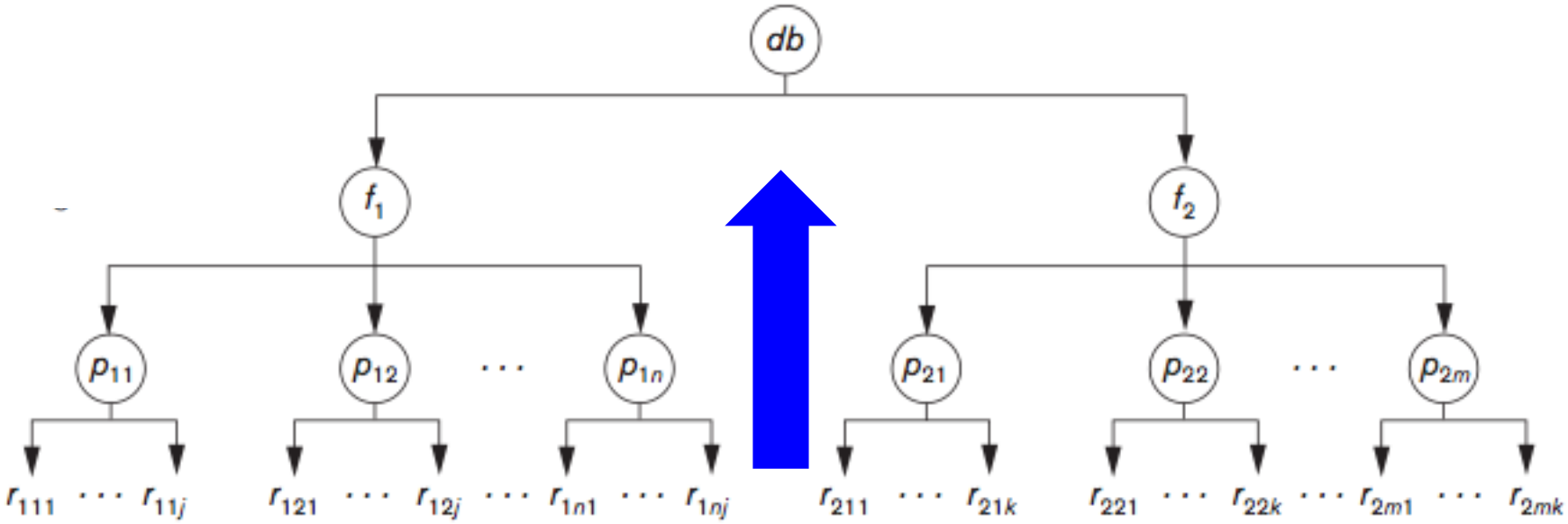
¿Qué sucedería?

Mecanismos para controlar la concurrencia

- El RDBMS debe verificar la compatibilidad de los cerrojos
- recorrido del árbol desde los registros.
- Si encuentra un cerrojo en conflicto deniega el cerrojo a T2 y la transacción debe esperar.

¿Qué pasaría si la solicitud de cerrojo de T2 se diera antes que la de T1?

Mecanismos para controlar la concurrencia



Se requieren otros tipos de cerrojos adicionales.

Mecanismos para controlar la concurrencia

- Protocolo de granularidad múltiple.
- Cerrojo de intención:
  - Marca los ancestros de un nodo al que se quiere aplicar un cerrojo (exclusivo o compartido).
  - Detectar conflictos de bloqueo en niveles superiores.
  - Intención compartido, intención exclusivo, intención compartida exclusiva.

Mecanismos para controlar la concurrencia

- Protocolo de granularidad múltiple.
- Cerrojo de intención compartido (is-lock):
  - Protección de bloqueos compartidos en niveles inferiores de la jerarquía.
  - Solicita cerrojos s-lock e is-lock en los niveles inferiores.
- Cerrojo de intención exclusivo (ix-lock):
  - Protección de bloqueos exclusivos en niveles inferiores de la jerarquía.
  - Protege solicitudes de bloqueos compartidos.

Mecanismos para controlar la concurrencia

- Protocolo de granularidad múltiple.
- Cerrojo de intención compartido exclusivo (isx-lock):
  - Niveles inferiores del árbol.
  - Protección bloqueos compartidos solicitados/adquiridos.
  - Permite bloqueos is-lock simultáneos en niveles superiores.
  - Sólo un bloqueo isx-lock simultáneo por recurso/objeto evitar actualizaciones.

Mecanismos para controlar la concurrencia

- Protocolo de granularidad múltiple.
- Los bloqueos se realizan de los niveles superiores a los inferiores, los desbloqueos a la inversa.

Mecanismos para controlar la concurrencia

- Matriz de compatibilidad.

	is-lock	ix-lock	s-lock	isx-lock	x-lock
is-lock	Ok	Ok	Ok	Ok	X
ix-lock	Ok	Ok	X	X	X
s-lock	Ok	X	Ok	X	X
isx-lock	Ok	X	X	X	X
x-lock	X	X	X	X	X

Mecanismos para controlar la concurrencia

- Protocolo de granularidad múltiple.
- Se cumple con la matriz de compatibilidad.
- La raíz se bloquea primero, en cualquier modo.
- Ti puede bloquear el nodo Q con (s-lock o is-lock) sólo si Ti está bloqueando al padre de Q con s-lock o is-lock.
- Ti puede bloquear el nodo Q con (x-lock, ix-lock o isx-lock) si Ti está bloqueando al padre de Q con ix-lock o isx-lock.

Mecanismos para controlar la concurrencia

- Protocolo de granularidad múltiple.
- Ti puede bloquear un nodo Q sólo si no ha desbloqueado un nodo cualquiera (protocolo 2 fases).
- Ti puede desbloquear un nodo Q sólo si no existen nodos descendientes de Q bloqueados por Ti.

Multiversión

- Para incrementar la concurrencia mantienen versiones anteriores de los datos:
  - Multiversion con ordenamiento de estampado de tiempo (timestamp).
  - Multiversión con bloqueo de dos fases.
- Cada operación write exitosa crea una nueva versión del objeto.
- Se usan timestamps para etiquetar los objetos.

Multiversión

- Cuando se ejecuta una operación read(Q), se elige y se retorna la versión correcta de Q, según el timestamp de la transacción.
- Las lecturas (read) nunca esperan, la versión correcta es retornada inmediatamente.

Multiversión – Ordenamiento de timestamps

- Cada objeto de datos Q tiene una secuencia de versiones <Q1, Q2,...., Qm>. Cada versión Qk contiene 3 campos de datos:
  - Contenido el valor de la versión Qk.
  - TS-W(Qk) timestamp de la transacción que creó (escribió) la versión Qk.
  - TS-R(Qk) timestamp mayor de una transacción que leyó exitosamente la versión Qk.
- Si Ti crea una nueva versión Qk de Q, los TS-W y los TS-R de Qk se inicializan = TS(Ti).
- El TS-R de Qk se actualiza cuando una transacción Tj lee Qk, y TS(Tj) >TS-R(Qk).

Multiversión

- Reglas:
- Si Ti ejecuta R(Q), retornar el valor de Qn inmediatamente inferior.
- Si Ti ejecuta W(Q):
  - Si TS(Ti) < TS-R(Qn), retroceder Ti.
  - Si TS(Ti) = TS-W(Qn), sobreescribir Qn.
  - Si no: es una nueva versión de Q.
- Las lecturas nunca fallan.
- Una escritura por parte de Ti se rechaza si:
  - Otra transacción Tj debe leer la escritura de Ti y ya leyó una versión de Q creada por otra transacción más antigua que Ti.
- Resuelve conflictos con retroceso.

Ejemplo

- Objeto / recurso / gránulo Q.

Transacción	Operación	TS-W	TS-R	Versión
T_1	Write	1	1	1
T_5	Write	5	5	5
T_7	Read	5	7	5
T_{10}	Write	10	10	10
T_{11}	Read	10	11	10

Ejemplo

- ¿Qué sucede si T_6 hace una lectura?

Transacción	Operación	TS-W	TS-R	Versión
T_6	Read	5	7	5

Ejemplo

- ¿Qué sucede si T_8 hace una lectura?

Transacción	Operación	TS-W	TS-R	Versión
T_5	Write	5	5	5
T_6	Read	5	7	5
T_7	Read	5	7	5

Transacción	**Operación**	**TS-W**	**TS-R**	**Versión**
T_5	Write	5	5	5
T_6	Read	5	7	5
T_7	Read	5	7	5
T_8	Read	5	8	5

Ejemplo

- ¿Qué sucede si T_6 hace una escritura?

Transacción	Operación	TS-W	TS-R	Versión
T_5	Write	5	5	5
T_6	Read	5	7	5
T_7	Read	5	7	5
T_8	Read	5	8	5

- Le corresponde la versión 5 del objeto.
- Pero se cumple: Si TS(Ti) < TS-R(Qn), es decir, TS(T6) < TS-R(Q5) abortar la transacción T6 o la T7.

Ejemplo

- Si se aborta T_7 y se reinicia con el mismo TS (6).

Transacción	Operación	TS-W	TS-R	Versión
T_6	Write	6	6	6
T_7	ABORTAR			
T_7	Read	6	7	6

Ejemplo

- Si se aborta T_6 y se reinicia con TS > último existente.

Transacción	Operación	TS-W	TS-R	Versión
T_6	ABORTAR			
T_{12}	Write	12	12	12

Gracias - ¿Preguntas?