UNIDAD 8: TÉCNICAS DE IMPLEMENTACIÓN DE LOS SGBD

(PARTE 2)

Gestión de Concurrencia

TÉCNICAS DE IMPLEMENTA-CIÓN DE LOS SGBDs

1. Mecanismos de RECUPERACIÓN de transacciones
2. **Gestión de accesos CONCURRENTES a la base de datos**

Uso “concurrente” de las calles/rutas

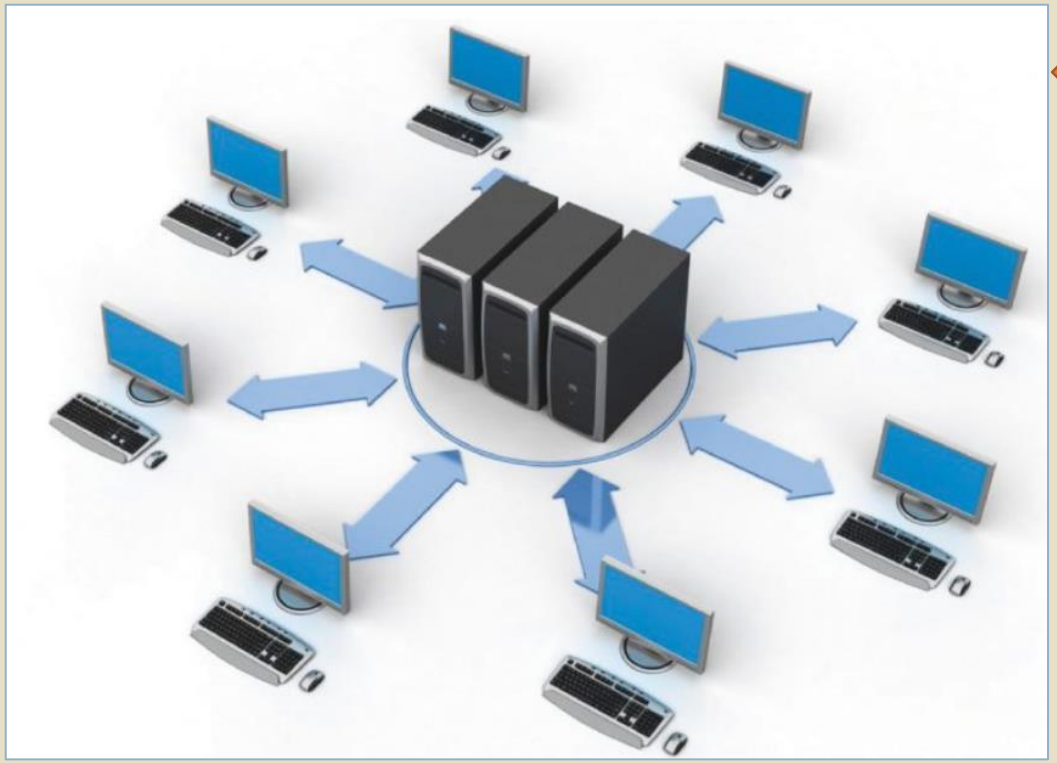
¿Será casual que no se vean problemas en la imagen, como por ejemplo, choques entre vehículos, vehículos que atropellan a peatones?

Protocolo

Conjunto de reglas de tránsito

Uso concurrente de una base de datos

¿Será casual que una BD, a pesar de ser utilizada por múltiples usuarios, se encuentre en un estado consistente/correcto?

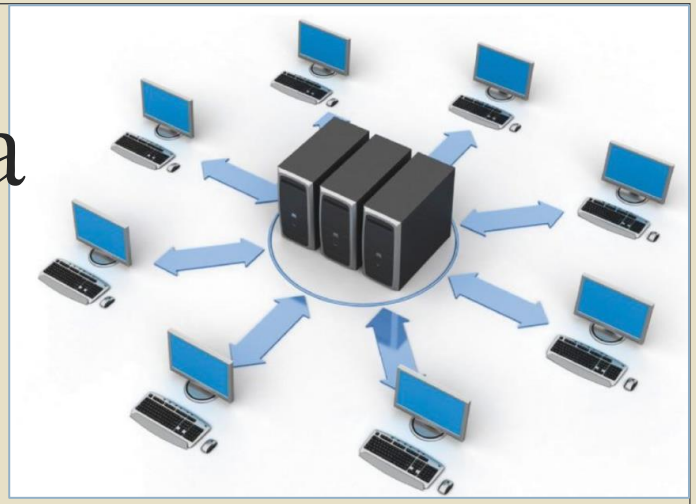


Protocolos

Conjunto de reglas de accesos a la BD

GESTOR DE CONCURRENCIA

SGBD: Gestor de Concurrencia



PROPÓSITO: Velar por que el **entrelazado de las operaciones**, llamadas **PLANIFICACIONES,** de las distintas transacciones produzcan un **estado consistente de la base de datos.**

tiempo

Planificación 1

T_1	T_2
read(A)	
$A := A - 50$	
write(A)	
read(B)	
$B := B + 50$	
write(B)	
commit	
	read(A)
	$temp := A * 0.1$
	$A := A - temp$
	write(A)
	read(B)
	$B := B + temp$
	write(B)
	commit

Planificación 2

T_1	T_2
read(A)	
$A := A - 50$	
write(A)	
	read(A)
	$temp := A * 0.1$
	$A := A - temp$
	write(A)
read(B)	
$B := B + 50$	
write(B)	
commit	
	read(B)
	$B := B + temp$
	write(B)
	commit

Planificación 3

T_1	T_2
read(A)	
$A := A - 50$	
	read(A)
	$temp := A * 0.1$
	$A := A - temp$
	write(A)
	read(B)
write(A)	
read(B)	
$B := B + 50$	
write(B)	
commit	
	$B := B + temp$
	write(B)
	commit

Planificación 4

T_1	T_2
	read(A)
	$temp := A * 0.1$
	$A := A - temp$
	write(A)
	read(B)
	$B := B + temp$
	write(B)
	commit
read(A)	
$A := A - 50$	
write(A)	
read(B)	
$B := B + 50$	
write(B)	
commit	

Gestor de Concurrencia: Propiedades ACID

Los SGBDs Relacionales aseguran las siguientes propiedades:

- **A**tomicity (Atomicidad)
- **C**onsistency (Consistencia)
- **I**solation (Aislamiento)
- **D**urability (Durabilidad)

GESTOR DE TRANSACCIONES
Componente del SGBD

Principio de correctitud de toda transacción

"Toda transacción ejecutada de forma aislada":

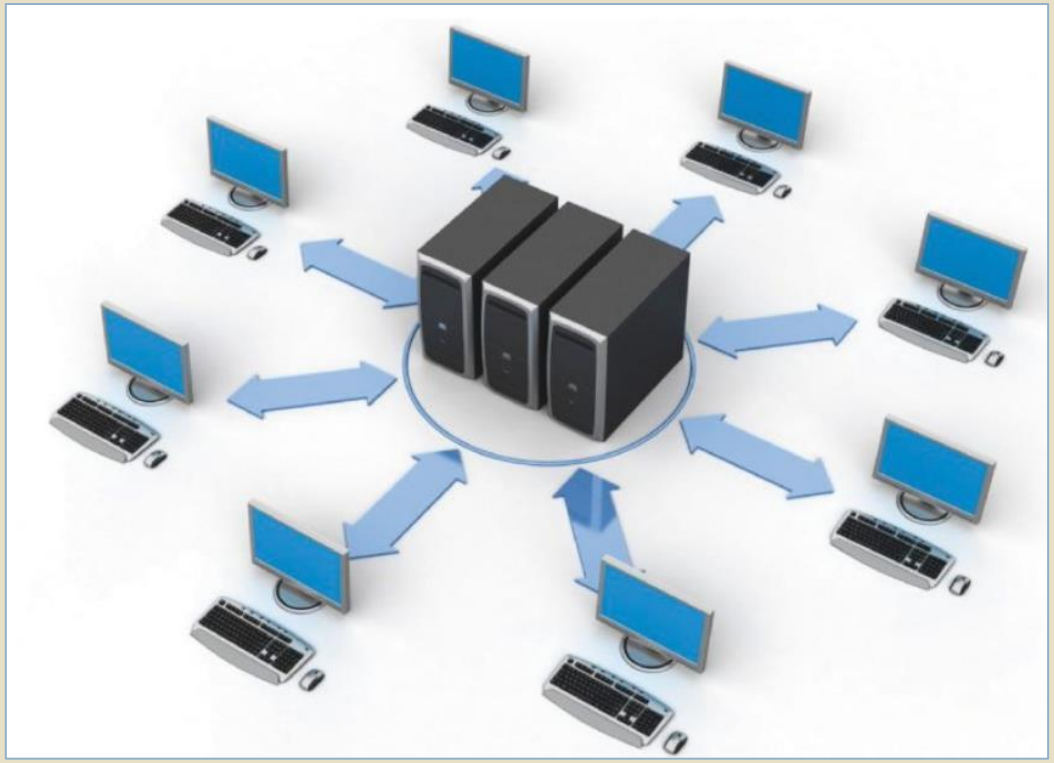
- Transforma **un estado consistente** de base de datos en **otro estado consistente (o el mismo)** de base de datos
- Sin embargo, **durante la ejecución** de una transacción, pueden existir **estados inconsistentes**.

… las transacciones se ejecutan concurrentemente,

sin esperar que no hayan transacciones ejecutándose…

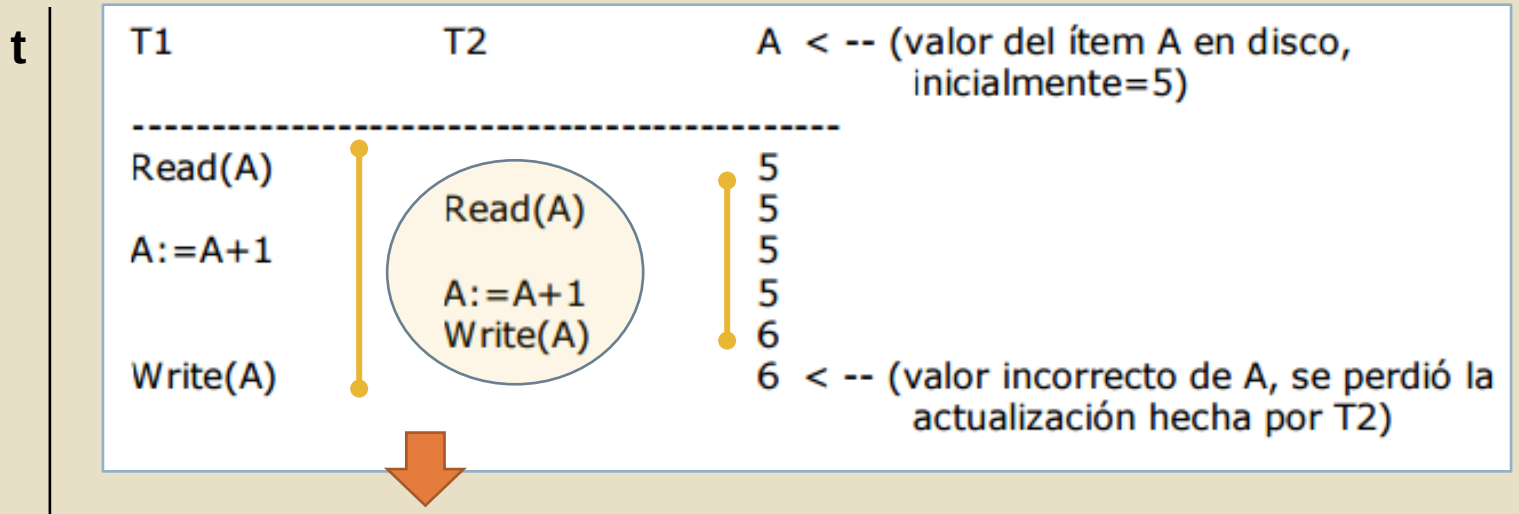
Algunos problemas de accesos concurrentes
(si no existieran protocolos)

- Pérdida de Update
- Lectura Sucia



Problema: Pérdida de Update

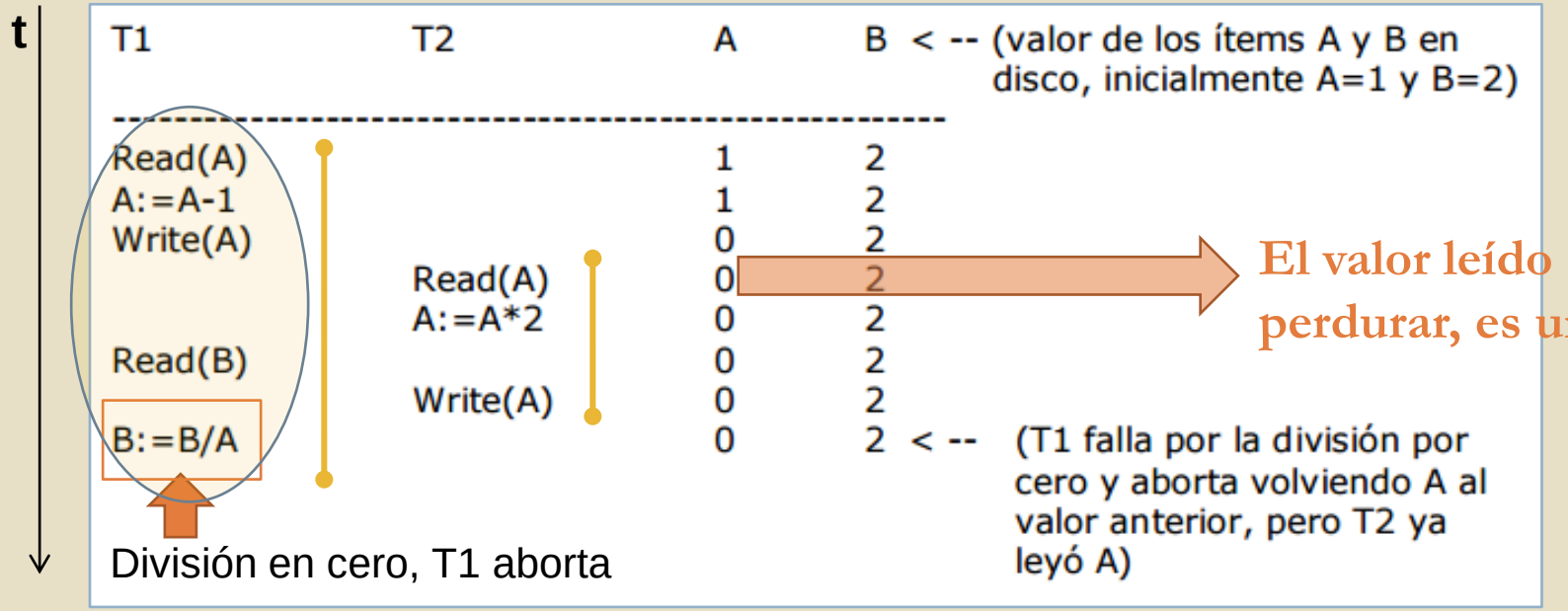
Ocurre cuando se pierde la actualización realizada por una transacción T1 debido a la acción de otra transacción T2 sobre el mismo ítem.



El valor de A no se incrementó 2 veces, se perdió un incremento

Problema: Lectura Sucia

Ocurre cuando una transacción lee un dato producido por una transacción que aborta posteriormente

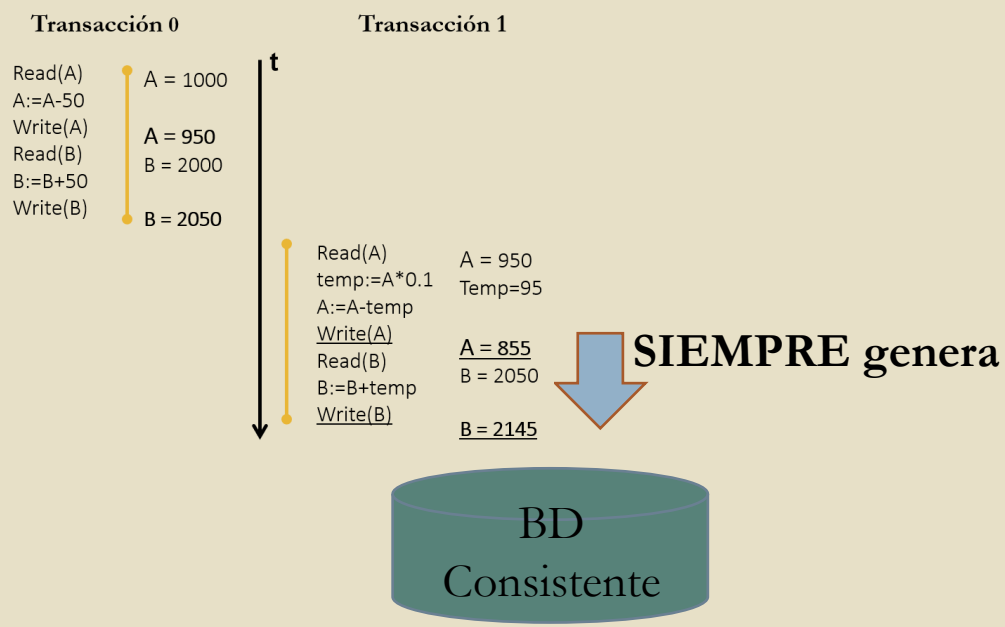


Tipos de Planificaciones

Existen **dos tipos de Planificaciones/Historias/Schedules**:

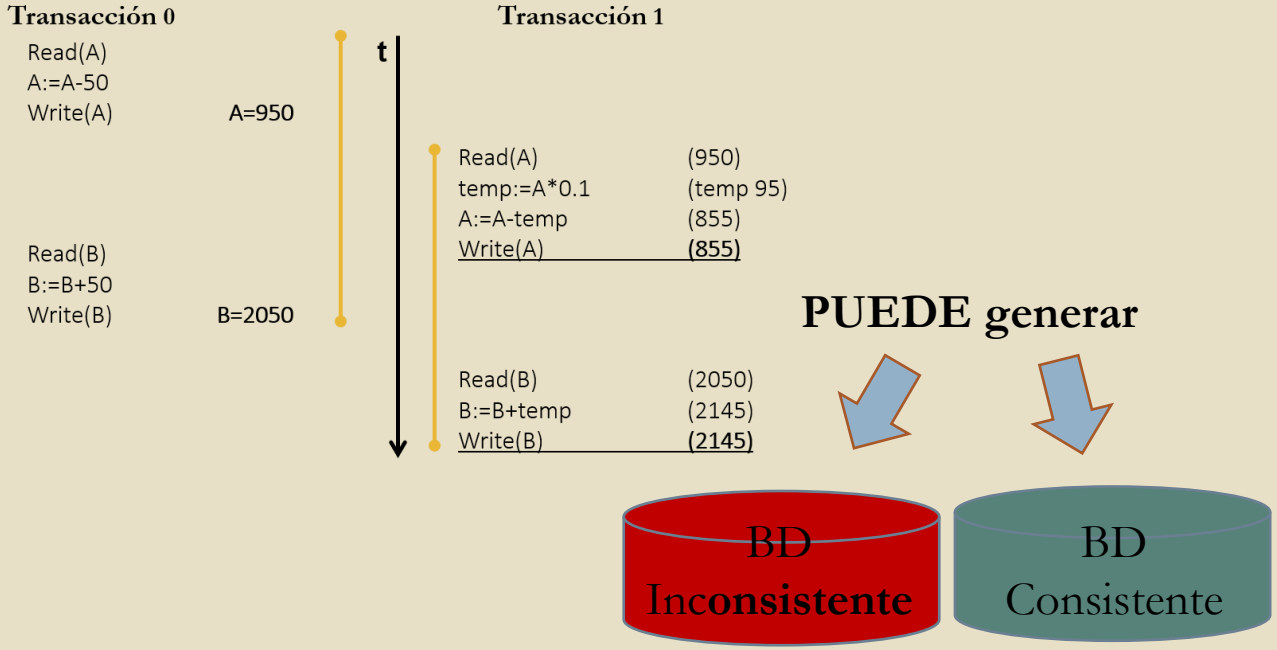
Planificaciones en Serie o Secuenciales:

Después que una transacción se ha ejecutado completamente, recién se ejecuta otra transacción.



Planificaciones Paralelas o Concurrentes:

Mientras se está ejecutando una transacción, entra otra a ejecutarse simultáneamente.



Planificaciones en Serie

Planificación1(Valores iniciales A=1000 y B=2000)**:T0 <** T1

Transacción 0

Read(A)
A:=A-50
Write(A)
Read(B)
B:=B+50
Write(B)

A = 1000

A = 950
B = 2000

B = 2050

t

Transacción 1

Read(A)
temp:=A*0.1
A:=A-temp
Write(A)
Read(B)
B:=B+temp
Write(B)

A = 950
Temp=95

A = 855
B = 2050

B = 2145

Planificación2(Valores iniciales A=1000 y B=2000)**:T1 <** T0

Transacción 0

A = 900

A = 850
B = 2100

B = 2150

Read(A)
A:=A-50
Write(A)
Read(B)
B:=B+50
Write(B)

t

Transacción 1

Read(A)
temp:=A*0.1
A:=A-temp
Write(A)
Read(B)
B:=B+temp
Write(B)

A = 1000
Temp=100

A = 900
B = 2000

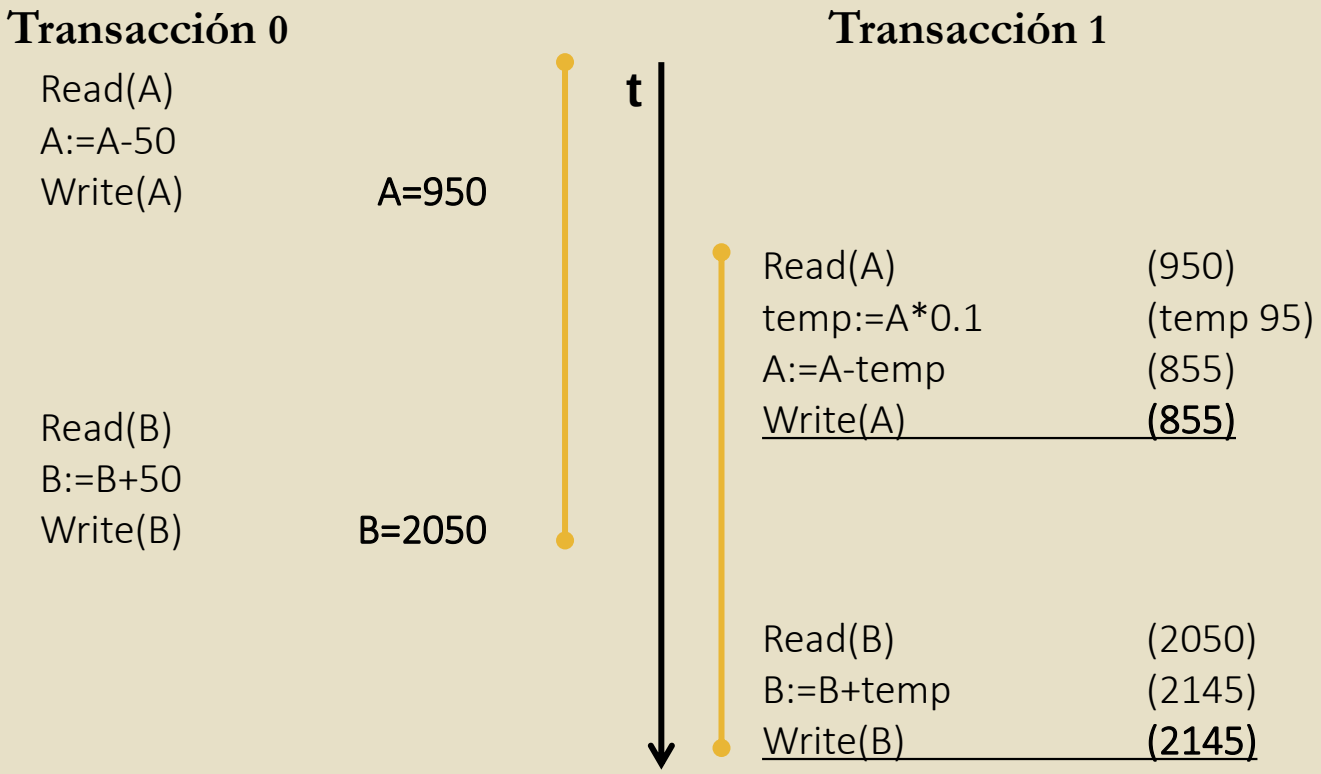
B = 2100

La **ejecución en serie** de ambas planificaciones (**T0 < T1** y **T1 < T0**), con valores iniciales de A =1000 y B=2000

A + B = 3000 **ESTADO CONSISTENTE**

Planificaciones concurrentes o en paralelo

Planificación 3: Valores iniciales de A =1000 y B=2000

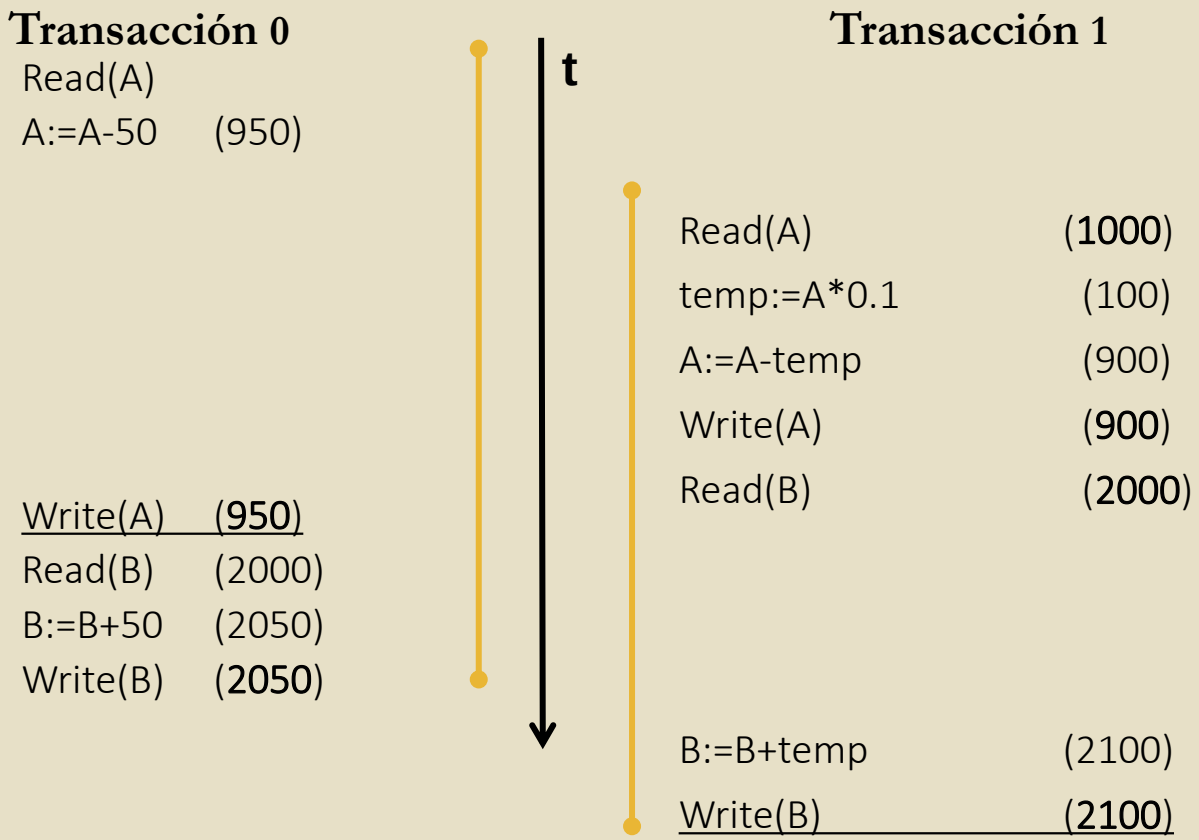


A= 855 B= 2145 **¿A+B=3000?** **SI**

Es una **planificación paralela y llega a un ESTADO CONSISTENTE**

Planificaciones concurrentes o en paralelo

Planificación 4: Valores iniciales de A = 1000 y B = 2000



A = 950 B = 2100 ¿A+B=3000? **NO, A+B = 3050**

Es una **planificación en paralelo que no llega ESTADO CONSISTENTE**

Planificaciones : Gestor de Concurrencia

- **TODA planificación EN SERIE (o secuencial) genera un ESTADO CONSISTENTE DE LA BD**

PERMITIR su ejecución (TODAS)

- **ALGUNAS planificaciones EN PARALELO generan a un ESTADO CONSISTENTE DE LA BD**

Distinguirlas

PERMITIR la ejecución SOLAMENTE de aquellas planificaciones concurrentes que generen un ESTADO CONSISTENTE de BD

¿Qué características o condiciones cumplen/tienen?

SER EQUIVALENTES a una PLANIFICACIÓN EN SERIE

1. Equivalencia según CONFLICTOS
2. Equivalencia según VISTAS

Equivalencia de Planificaciones en cuanto a Conflictos

1. **¿Cómo saber si una planificación es equivalente a otra, según conflictos?**

Pruebas de Serializabilidad/Secuencialidad

- **<u>Instrucciones/operaciones en conflicto</u>**

Dos instrucciones, **I_i e I_j están en conflicto cuando**:

- Pertenecen a diferentes transacciones.
- Operan sobre el mismo dato.
- Al menos una de ellas es un write.

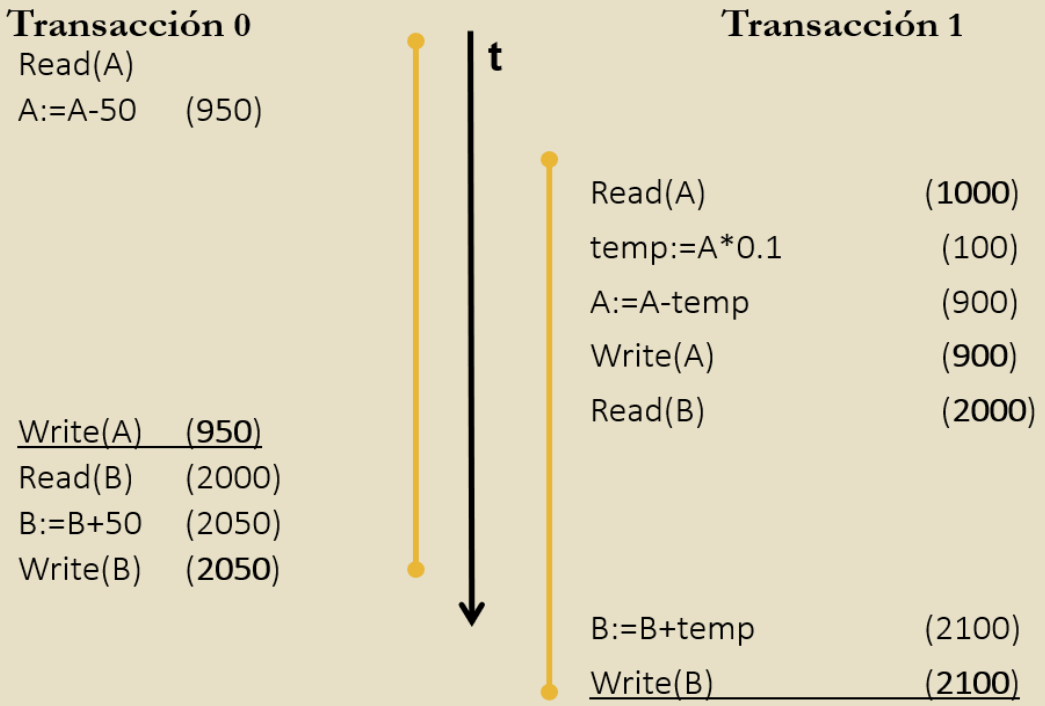
Pruebas de Serializabilidad en cuanto a CONFLICTOS

a. Intercambio de operaciones (S y S')

- Todas las instrucciones que no estén en conflicto pueden intercambiarse
- Si a través de esos intercambios en la planificación paralela, se llega a una planificación en serie, entonces la planificación es equivalente a esa planificación en serie

Analicemos las Planificaciones 3 y 4:

Planificación 4: Valores iniciales de A = 1000 y B = 2000



A = 950 B = 2100 ¿A+B=3000? NO, A+B = 3050

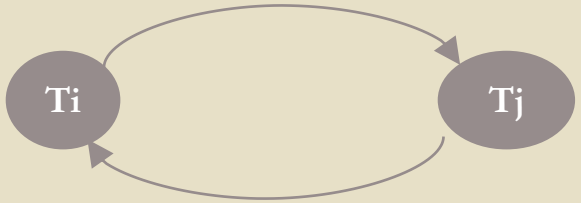
Pruebas de Serializabilidad en cuanto a CONFLICTOS

b. Grafo de precedencia dirigido

- Conjunto de vértices = las transacciones
- Conjunto de aristas *Ti*→*Tj* según se cumplan:

1. Ti ejecuta write(Q) antes de que Tj ejecute un read(Q).
2. Ti ejecuta read(Q) antes de que Tj ejecute un write(Q).
3. Ti ejecuta write(Q) antes de que Tj ejecute un write(Q).

Si el grafo presenta un ciclo, la **planificación no es serializable en cuanto a conflictos.**



La planificación no es serializable en conflictos

Analicemos las Planificaciones 3 y 4:

Planificación 4: Valores iniciales de A = 1000 y B = 2000

Transacción 0		Transacción 1	
Read(A)			
A:=A-50	(950)		
		Read(A)	(1000)
		temp:=A*0.1	(100)
		A:=A-temp	(900)
		Write(A)	(900)
		Read(B)	(2000)
Write(A)	(950)		
Read(B)	(2000)		
B:=B+50	(2050)		
Write(B)	(2050)		
		B:=B+temp	(2100)
		Write(B)	(2100)

t

A = 950 B = 2100 ¿A+B=3000? **NO, A+B = 3050**

Equivalencia de Planificaciones en cuanto a Vistas

2. **¿ Cómo saber si dos planificaciones son equivalentes en cuanto a vistas?**

Deben cumplirse las siguientes <u>3 condiciones</u>:

1. Para c/dato Q, **si la transacción Ti lee el valor inicial de Q en S** => Ti también debe leer el valor inicial de Q en S'.

2. Para c/dato Q, **si la transacción Ti ejecuta un read de Q en S y ese valor fue producido por la transacción Tj** => Ti también debe leer el valor producido por Tj de Q en S'.

3. Para c/dato Q, **la transacción que ejecute el último write de Q en S** => también debe ejecutarlo en S'.

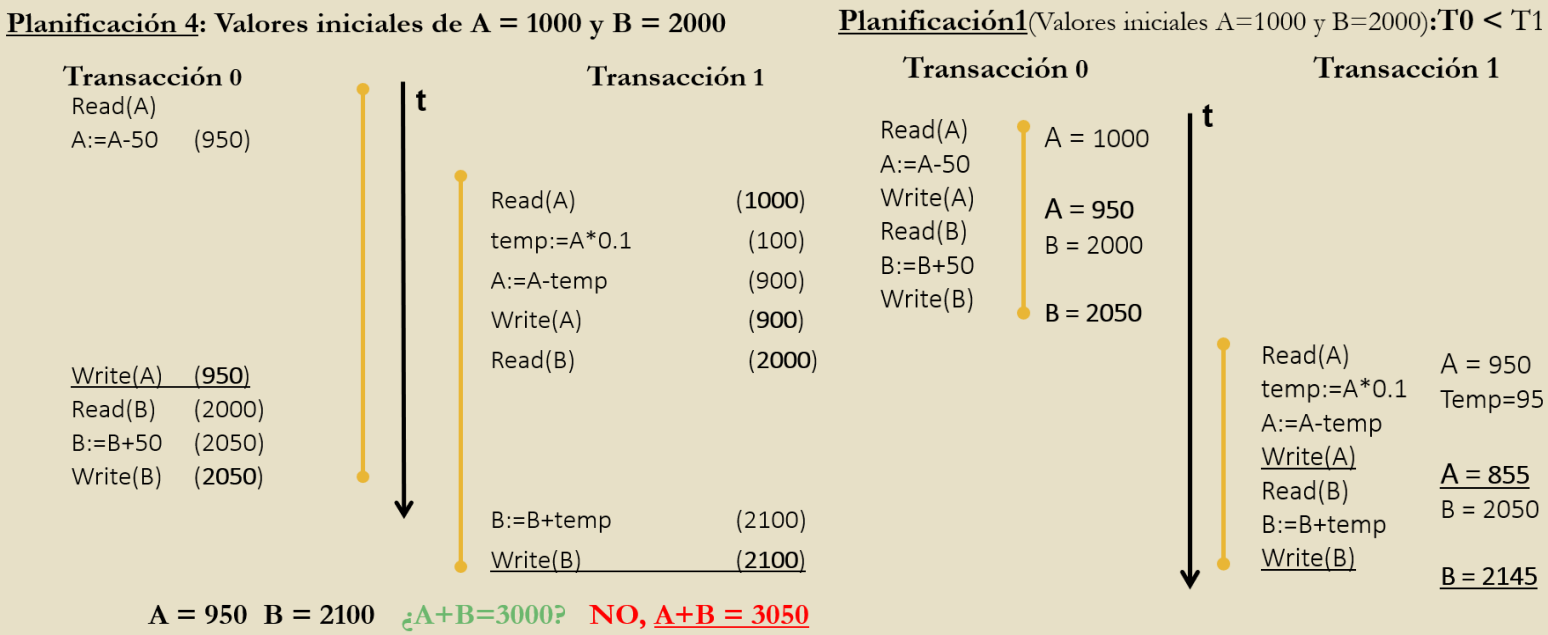
NOTA: - De 1 y 2: Lectura de los mismos valores.

- De 3, junto con 1 y 2: Las planificaciones dejan el mismo resultado.

Equivalencia de Planificaciones en cuanto a Vistas

1. **Ti lee el valor inicial de Q en S,** Ti también debe leer el valor inicial de Q en S'.
2. **Ti ejecuta un read de Q en S y ese valor fue producido por la transacción Tj,** Ti también debe leer el valor producido por Tj de Q en S'.
3. **La transacción que ejecute el último write de Q en S**, también debe ejecutarlo en S'.

Analicemos las Planificación 4:



SGBD: Gestor de Concurrencia

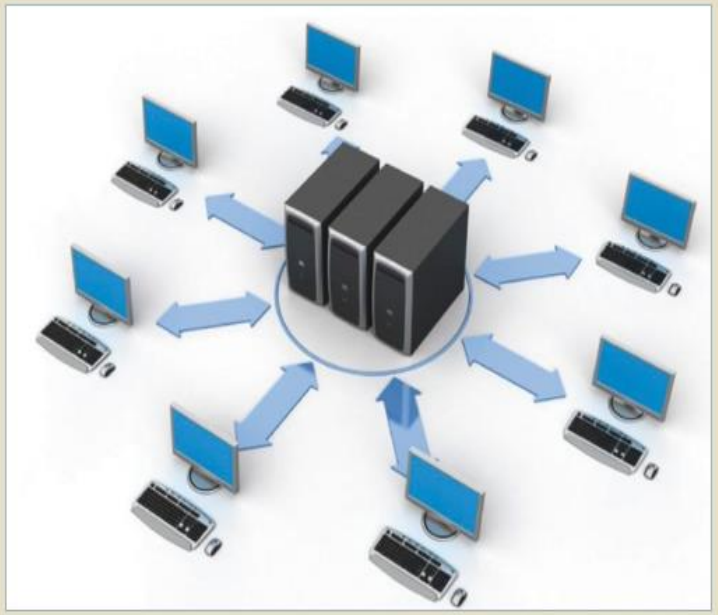
Velar por que el **entrelazado de las operaciones**, de las distintas transacciones produzcan un **estado consistente de la base de datos.**

NO PUEDE MANEJAR EL ORDEN EN EL QUE LAS

TRANSACCIONES Y OPERACIONES LLEGAN AL SISTEMA

¿Cómo hace el SGBD para “acomodar” las operaciones de manera que se generen planificaciones que lleven a la BD a un estado consistente?

PROTOCOLOS/ESQUEMAS DE CONTROL DE CONCURRENCIA



Protocolos

Conjunto de reglas de accesos a la BD

GESTOR DE CONCURRENCIA

Protocolo

Conjunto de reglas de tránsito

Protocolos de Concurrencia

A. **Protocolos basados en bloqueos (planificaciones equivalentes en conflictos)**
   - Resuelven el problema de inconsistencia
   - Pueden caer en estados de deadlock (abrazo mortal)/inanición

B. **Protocolos basados en marcas temporales (planificaciones equivalentes en vistas con Regla de Thomas)**
   - Resuelven el problema de inconsistencia
   - No caen en estados de deadlock

Protocolo de Bloqueos

En este caso, las transacciones antes de ejecutar una operación de lectura o escritura deben:

- **Solicitar el bloqueo correspondiente.**
- **Si se le concede** el bloqueo, se **ejecuta la operación.**
- **Si no se le concede** el bloqueo, la transacción queda en **espera** hasta que la transacción que está ocupando el dato lo libere, y ésta pueda obtener el bloqueo solicitado.

Protocolo de Bloqueos

Existen dos tipos de bloqueos:

- S: Compartido
- X: Exclusivo

Consideraciones:

- Si un dato esta bloqueado en forma exclusiva, no se podrá conceder ningún otro bloqueo sobre el mismo dato (ni X ni S).

	S	X
S	true	false
X	false	false

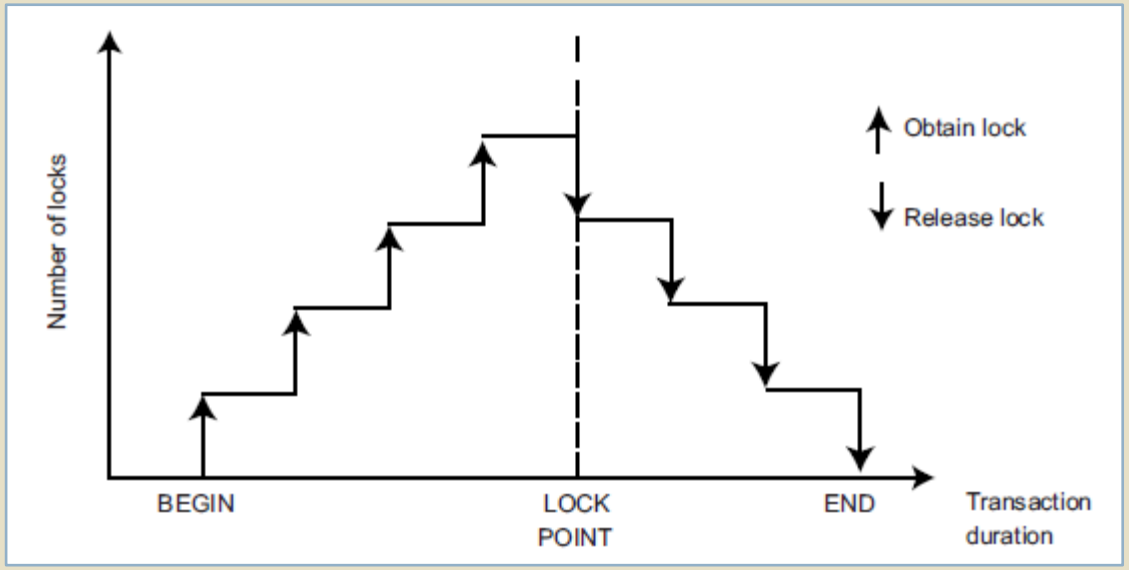
- Los bloqueos no se pueden liberar en cuanto el dato no se necesite, sólo pueden liberarse al final de una transacción (commit o rollback) o en la fase de decrecimiento (según el protocolo particular que se esté usando).
- La espera de una transacción ante la solicitud de un bloqueo puede llevar a un estado de deadlock, en cuyo caso se debe abortar alguna de las transacciones para que la otra pueda continuar.

Protocolos de Bloqueo

- Protocolo de Bloqueo simple
- Protocolo de Bloqueo de dos Fases
- Protocolo de Bloqueo basado en Grafos

Protocolo de Bloqueo de 2 Fases (2F)

Cuando una transacción desbloquea un dato, entra en la fase de decrecimiento, y allí ya no puede pedir ningún bloqueo.



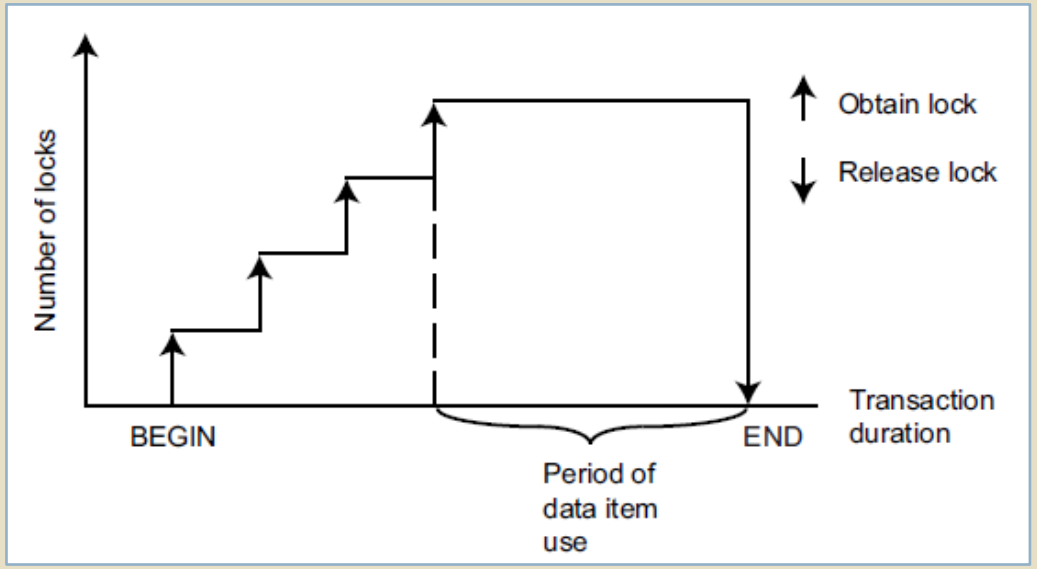
Protocolos de Bloqueo de 2 Fases Refinado

Se permiten CONVERSIONES de bloqueo:

- **Upgrade** -> Fase de Crecimiento
- **Downgrade** -> Fase de Decrecimiento

Protocolo de Bloqueos dos Fases Estricto (2FE)

Una transacción es 2FE si cumple con 2F y además **no libera ninguno de sus bloqueos exclusivos** hasta después de haber hecho **commit o abort**.



Protocolos de Concurrencia

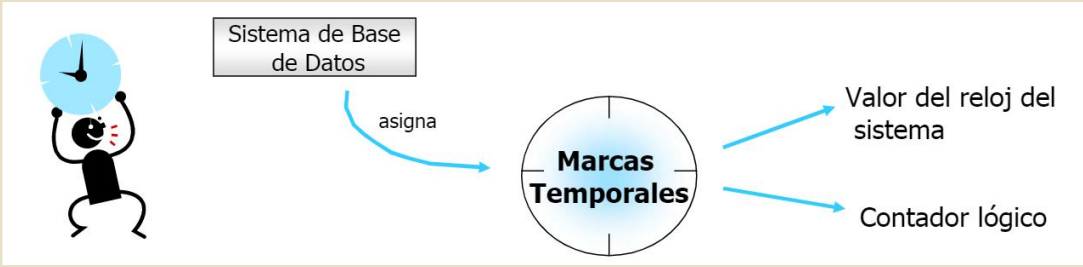
A. **Protocolos basados en bloqueos (planificaciones equivalentes en conflictos)**
   - Resuelven el problema de inconsistencia
   - Pueden caer en estados de deadlock

B. **Protocolos basados en marcas temporales (planificaciones equivalentes en vistas con Regla de Thomas)**
   - Resuelven el problema de inconsistencia
   - No caen en estados de deadlock

¿Nos tomamos un descanso?

Protocolo de Marcas Temporales/Basados en hora de entrada

- **Cada transacción Ti tiene una hora de entrada única: TS(Ti)**



- **Para cada dato Q**, se le asocia y mantiene:
  - **W-hora (Q)**: Mayor hora de entrada de cualquier transacción que ejecutó con éxito un write(Q)
  - **R-hora (Q)**: Mayor hora de entrada de cualquier transacción que ejecutó con éxito un read(Q)

- Si Ti entra antes que Tj, entonces:
  - **TS**(Ti) < **TS**(Tj)
- Se asegura la equivalencia a la planificación serie en el mismo orden de precedencia de la planificación concurrente:
  - Entonces si **TS**(Ti) < **TS**(Tj), el sistema asegura que la planificación producida es equivalente a la planificación serie Ti<Tj

Protocolo de Marcas Temporales/Basados en hora de entrada

Cada operación de lectura y escritura debe:

1. **Ser evaluada** por el sistema para definir si es posible o no su ejecución

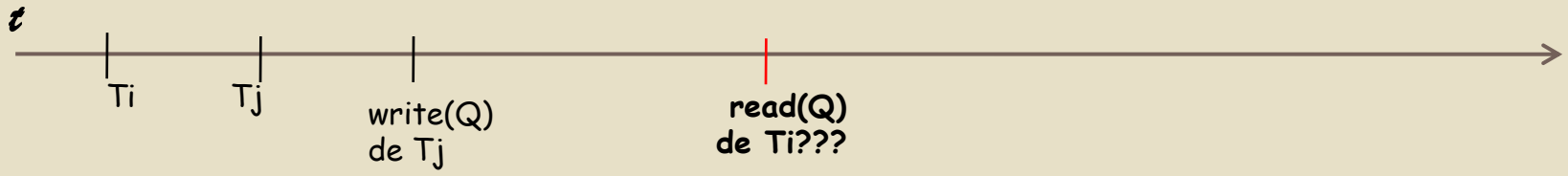
2. **En caso de poder ejecutarse**:
   - Se ejecuta efectivamente y
   - Se actualiza la marca temporal (W-hora-entrada o R-hora-entrada) del dato en cuestión según la operación realizada

3. **En caso de no poder ejecutarse**:
   - La transacción se aborta
   - Se relanza la transacción, obteniendo así obviamente una nueva marca temporal par la transacción TS(Ti)

Protocolo de Marcas Temporales/Basados en hora de entrada

i. Supongamos que **Ti** solicita un **read(Q)**:

- Si TS(Ti) < W-h(Q) -> **Ti retrocede**

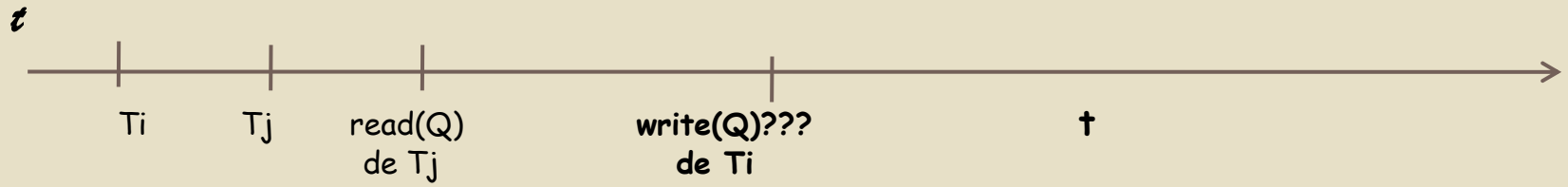


- Si TS(Ti) >= W-h(Q) -> Ti ejecuta el read y se actualiza el valor de R-h(Q)

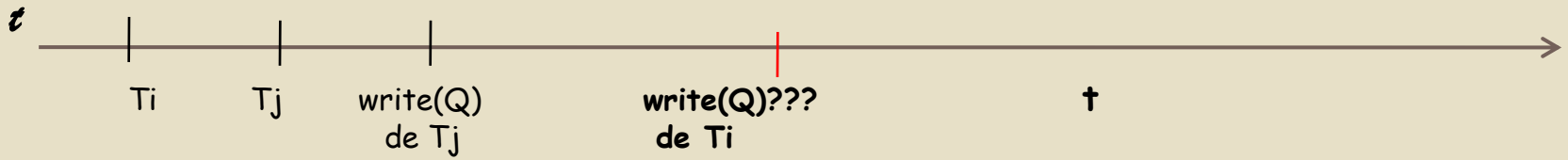
Protocolo de Marcas Temporales/Basados en hora de entrada

ii. Supongamos que **Ti** solicita un **write(Q)**:

- Si TS(Ti) < R-h(Q) ->**Ti retrocede**



- Si TS(Ti) < W-h(Q) ->**Ti retrocede**



- En los demás casos la operación **write se ejecuta** y se actualiza el valor de **W-h(Q)**

Protocolo de Marcas Temporales/Basados en hora de entrada

Veamos un ejemplo (T1< T2):

Transacción 1	Transacción 2
(1) Read(A)	
	(2) Write(A) *
(3) Write(A)	

(1) se ejecuta con éxito

(2) se ejecuta con éxito

(3) se rechaza y T1 retrocede puesto que TS(T1) < W-hora entrada(A) (=TS(T2)

Es decir, esta Planificación **no es válida en este protocolo**:

- Pero, de haberse ejecutado en serie, T1< T2 el valor que hubiese quedado sería el escrito por T1.

Nota: * La transacción T2 ejecuta un write a ciegas

Protocolo de Marcas Temporales/Basados en hora de entrada

Thomas aporta un a mejora al protocolo en relación a la evaluación de las operaciones write:

ii. Supongamos que Ti solicita un write(Q):

- Si TS(Ti) < R-h(Q) -> Ti retrocede, **igual que vimos antes**

- Si TS(Ti) < W-h(Q) -> **Ti no retrocede, sino que la operación write(Q) de Ti no se ejecuta**

  ✓ ***No permite el write*** *porque el valor de Q que quedaría no sería el mismo que en la planificación equivalente en serie Ti<Tj. Sin embargo, el write que intenta "cuidar", es un valor obsoleto.*

  ✓ *Entonces,* ***sin ejecutar el write en cuestión****, el valor de Q que quedaría sería el mismo que si se hubiese ejecutado en la planificación en serie Ti<Tj, es decir, el de Tj.*

- En los demás casos la operación write se ejecuta y se actualiza el valor de W-h(Q)

Protocolo de Marcas Temporales/Basados en hora de entrada

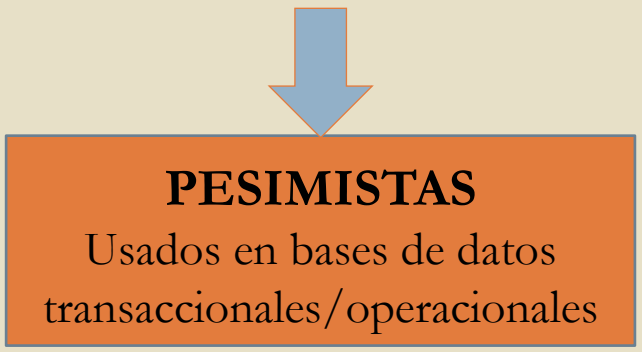
A. **Protocolos basados en bloqueos** (conflictos)
   - Obviamente, resuelven el problema de inconsistencia
   - Pueden caer en estados de deadlock

B. **Protocolos basados en marcas temporales**

   (con Regla de Thomas)
   - Obviamente, resuelven el problema de inconsistencia
   - No caen en estados de deadlock

Tipos de Protocolos de Concurrencia

Esquemas/Protocolos de Control de Concurrencia



OPTIMISTAS

¿En qué categoría incluirían los protocolos vistos, es decir, los basados en bloqueos y en marcas temporales?

Gestor de Transacciones: Propiedades ACID

Los SGBDs Relacionales aseguran las siguientes propiedades:

- **A**tomicity (Atomicidad)
- **C**onsistency (Consistencia)
- **I**solation (Aislamiento)
- **D**urability (Durabilidad)

GESTOR DE TRANSACCIONES
Componente del SGBD

Llegamos al final…

¡Qué lindo fue compartir con ustedes este año!!!

Los vamos a extrañar… nos encontraremos en el examen