



Tema 7. Procesamiento de Transacciones

1. Introducción y objetivos
2. Conceptos Básicos
3. Problemas de la Concurrencia
4. Problemas de la Recuperabilidad
5. Servicio de Transacciones
6. Métodos de Control de la Concurrencia

Tema 7

Procesamiento de transacciones

1



Introducción y objetivos

Tema 7

Procesamiento de transacciones

2



Introducción. 1

- Transacciones
 - Arquitectura cliente-servidor
 - Un servidor encapsula recursos
 - Garantía de atomicidad (unidades indivisibles)
 - Ejemplo representativo: transacciones bancarias
- Concurrencia
 - Ejemplos:
 - Servicio de ficheros
 - Tiempo y coordinación distribuida
 - Replicación

Tema 7

Procesamiento de transacciones

3



Recuperación y tolerancia a fallos

- Características
 - Tolerancia a fallos:
 - Los fallos que ocurran no deben dejar los datos en un estado inconsistente
 - Recuperación de los datos:
 - Uso de disco
- Mecanismos de implementación:
 - Lista de intenciones
 - Versiones de ficheros

Tema 7

Procesamiento de transacciones

4



Objetivos

- Establecer un sencillo modelo cliente-servidor que gestiona el acceso a datos (Transacción)
- Estudiar las características de este modelo en presencia de clientes concurrentes y ante fallos
- Estudiar las tres principales aproximaciones al control de la concurrencia usando transacciones

Tema 7

Procesamiento de transacciones

5



Conceptos básicos

Tema 7

Procesamiento de transacciones

6



Conceptos básicos. 1

- Proceso cliente-servidor. Dos exigencias:
 - los clientes deben ser capaces de usar los servidores para compartir e intercambiar información
 - las operaciones de un cliente deben estar protegidas de las interferencias de las operaciones de otros clientes
- Transacción: en algunas aplicaciones,
 - los clientes necesitan ejecutar secuencias de operaciones
 - sobre el servidor
 - como una unidad indivisible

Tema 7

Procesamiento de transacciones

7



Conceptos básicos. 2

- Si servicio dividido sobre varios servidores: *Transacción Distribuida*
- Posibilidad de que las operaciones en el servidor interfieran
 - ⇒ necesidad de *operaciones atómicas*
- *Transacción atómica*: sucesión de operaciones de un cliente:
 - ejecutadas por el servidor
 - sin interferencia de otros clientes
 - formando un solo paso lógico

Tema 7

Procesamiento de transacciones

8



Requisitos

- Una transacción atómica debe ser:
 - serializable: la ejecución concurrente de varias transacciones debe ser equivalente a su ejecución en serie
 - los efectos intermedios de una transacción no deben ser visibles a otras transacciones concurrentes
 - recuperable: todo o nada en sus efectos.
 - o la transacción se completa totalmente de forma satisfactoria y sus efectos se registran de forma permanente
 - o, si se produce algún fallo, no se registra ningún efecto
- Propiedades *ACID*:
 - Atomicidad, Consistencia, aislamiento y Durabilidad

Tema 7

Procesamiento de transacciones

9



Interfaz del Servicio de Transacciones

AbreTransaccion () → (TID)

- Arranca una nueva transacción y devuelve el *TID* como referencia a la misma

CierraTransaccion (TID) → (ResulTrans)

- Cierra *TID* y devuelve el *ResulTrans* correspondiente
 - *comprometida*: todo bien
 - *abortada*: algo mal

AbortaTransaccion (TID)

- Aborta

Tema 7

Procesamiento de transacciones

10



Problemas de la Concurrency

Tema 7

Procesamiento de transacciones

11



El problema de la Actualización Perdida. 1

| Transacción T: <i>Transferir 4 unidades de A a B</i> | Transacción U: <i>Transferir 3 unidades de C a B</i> |
|---|---|
| Leer el balance (a) de la cuenta A | Leer el balance (c) de la cuenta C |
| Leer el balance (b) de la cuenta B | Escribir el nuevo balance (c-3) de la cuenta C |
| Escribir el nuevo balance (a-4) de la cuenta A | Leer el balance (b) de la cuenta B |
| Escribir el nuevo balance (b+4) de la cuenta B | Escribir el nuevo balance (b+3) de la cuenta B |

Tema 7

Procesamiento de transacciones

12



El problema de la Actualización Perdida. 2

| Transacción T: <i>Transferir 4 unidades de A a B</i> | | Transacción U: <i>Transferir 3 unidades de C a B</i> | |
|---|-----|---|-----|
| a = A.Lee() | 100 | c = C.Lee() | 300 |
| A.Escribe(a-4) | 96 | C.Escribe(c-3) | 297 |
| b = B.Lee() | 200 | b = B.Lee() | 200 |
| B.Escribe(b+4) | 204 | B.Escribe(b+3) | 203 |

Tema 7

Procesamiento de transacciones

13



El problema de la Recuperación Inconsistente

| Transacción V: <i>Transferir 100 unidades de B a C</i> | | Transacción W: <i>Determinar el total del banco</i> | |
|---|-----|--|-----|
| b = B.Lee() | 200 | total = A.Lee() | 200 |
| B.Escribe(b-100) | 100 | total += B.Lee() | 300 |
| | | total += C.Lee() | 500 |
| | | ... | |
| c = C.Lee() | 200 | ... | |
| C.Escribe(c+100) | 300 | ... | |

Tema 7

Procesamiento de transacciones

14



La equivalencia serie

- La actualización perdida ocurre cuando dos transacciones leen un mismo dato y lo usan para calcular el valor nuevo
- La equivalencia serie requiere que todos los accesos de una transacción a un dato en particular puedan ser serializados con respecto a los accesos de otras transacciones al mismo dato
- La equivalencia serie se utiliza como criterio a la hora de desarrollar protocolos de control de concurrencia

Tema 7

Procesamiento de transacciones

15



Entrelazado equivalente serie de T y U

| Transacción T: <i>Transferir 4 unidades de A a B</i> | | Transacción U: <i>Transferir 3 unidades de C a B</i> | |
|---|-----|---|-----|
| a = A.Lee() | 100 | c = C.Lee() | 300 |
| A.Escribe(a-4) | 96 | C.Escribe(c-3) | 297 |
| b = B.Lee() | 200 | | |
| B.Escribe(b+4) | 204 | b = B.Lee() | 204 |
| | | B.Escribe(b+3) | 207 |

Tema 7

Procesamiento de transacciones

16



Entrelazado equivalente serie de V y W

| Transacción V: <i>Transferir 100 unidades de B a C</i> | | Transacción W: <i>Determinar el total del banco</i> | |
|---|-----|--|-----|
| $b = B.Lee()$ | 200 | $total = A.Lee()$ | 200 |
| $B.Escribe(b-100)$ | 100 | | |
| $c = C.Lee()$ | 200 | | |
| $C.Escribe(c+100)$ | 300 | $total += B.Lee()$ | 300 |
| | | $total += C.Lee()$ | 600 |
| | | ... | |

Tema 7

Procesamiento de transacciones

17



Problemas de la Recuperabilidad

Tema 7

Procesamiento de transacciones

18



Problemas de la Recuperabilidad

- El servicio de transacciones debe:
 - tener en cuenta que una transacción puede abortar
 - y evitar que esto afecte a otras transacciones concurrentes
- Dos problemas:
 - lectura sucia
 - escritura prematura

Tema 7

Procesamiento de transacciones

19



El problema de la *Lectura Sucia*

| Transacción T: <i>Depositar 3 unidades en A</i> | | Transacción U: <i>Depositar 5 unidades en A</i> | |
|--|-----|--|-----|
| $a = A.Lee()$ | 100 | | |
| $A.Escribe(a+3)$ | 103 | $a = A.Lee()$ | 103 |
| | | $A.Escribe(a+5)$ | 108 |
| | | $CierraTransaccion(U)$ | |
| | | → comprometida | |
| $AbortaTransaccion(T)$ | | | |

Tema 7

Procesamiento de transacciones

20



Escrituras prematuras

- Algunas sistemas de base de datos implementan la acción de abortar restaurando el *valor anterior* que tenían los datos escritos por una transacción
- Esto es un problema cuando varias transacciones escriben el mismo dato y una o varias de ellas abortan

Tema 7

Procesamiento de transacciones

21



El problema de la Escritura Prematura. 1

| Transacción T: <i>Depositar 3 unidades en A</i> | | Transacción U: <i>Depositar 5 unidades en A</i> | |
|--|-----|--|----------------|
| a = A.Lee() | 100 | a = A.Lee() | 103 |
| A.Escribe(a+3) | 103 | A.Escribe(a+5) | 108 |
| | | CierraTransaccion(U) | → comprometida |
| AbortaTransaccion(T) | 100 | | |

Tema 7

Procesamiento de transacciones

22



El problema de la Escritura Prematura. 2

| Transacción T: <i>Depositar 3 unidades en A</i> | | Transacción U: <i>Depositar 5 unidades en A</i> | |
|--|-----|--|-----|
| a = A.Lee() | 100 | a = A.Lee() | 103 |
| A.Escribe(a+3) | 103 | A.Escribe(a+5) | 108 |
| AbortaTransaccion(T) | 100 | AbortaTransaccion(U) | 103 |

Tema 7

Procesamiento de transacciones

23



Retardo del compromiso

- *Lectura sucia:*
 - una transacción se compromete después de haber visto los efectos de una transacción que más tarde abortará
 - ⇒ la situación no será recuperable
- Para evitar que ocurra esto,
 - se podrían consentir dichas lecturas, y que
 - una transacción que corra el riesgo de haber realizado una lectura sucia
 - retrase su compromiso hasta que terminen las transacciones cuyo estado no comprometido haya observado

Tema 7

Procesamiento de transacciones

24



Aborto de transacciones en cascada

- Si aborta una transacción anterior cuyas operaciones afectan a una transacción T , lo más probable es que T también aborte.
 - además, otras transacciones afectadas por las operaciones de T tendrán que abortar, y así sucesivamente
- Para evitar que suceda esto, otra opción es que:
 - una operación de lectura de un dato se retrase hasta que se comprometa o aborte toda transacción que haya aplicado una operación de escritura sobre dicho dato
- Hay mecanismos menos restrictivos

Tema 7

Procesamiento de transacciones

25



Versiones provisionales

- El servicio de transacciones debe asegurar que cualquier actualización de los datos que haya hecho una transacción no se realice si ésta aborta
- Para ello, las modificaciones de datos se registran en memoria de forma:
 - provisional: sólo se transferirán a disco si la transacción se compromete
 - aislada: cada transacción tendrá su propia versión provisional de los datos que haya modificado
- La escritura a disco se realiza en exclusión mutua

Tema 7

Procesamiento de transacciones

26



Interfaces de Servicio con soporte de Transacciones

Tema 7

Procesamiento de transacciones

27



Interfaz de servicio con soporte de Transacciones

- Interfaz que soporta transacciones atómicas sobre los objetos que maneja
- Servicio de Transacciones como extensión del servicio básico:
 - garantiza atomicidad
 - 3 pasos:
 1. *AbreTransaccion* → TID
 2. Operaciones sobre objetos (provisionales y aisladas)
 3. Finalmente:
 - o *CierraTransaccion*. Dos opciones:
 - » bien → comprometida
 - » mal → abortada por el servidor ⇒ informe al cliente
 - o *AbortaTransaccion*: explícitamente por el cliente

Tema 7

Procesamiento de transacciones

28



Ejemplo: Operaciones Modificadas para la Interfaz del Servicio de Ficheros.

Escribe(TID, UFID, Posicion, Datos)
- **INFORME (PosicionErronea, UFIDmal, Abortada)**

- Tiene el mismo efecto que *Escribe (UFID, Posicion, Datos)*, pero registra los datos de una manera provisional, pendiente de la terminación de *TID*

Lee (TID,UFID,Posicion, n) → (Datos)
- **INFORME (PosicionErronea, UFIDmal, Abortada)**

- Devuelve el valor provisional de los *Datos*, resultante de la ejecución de *TID*, si ésta ha registrado alguno.
- Si no, tiene el mismo efecto que *Lee (UFID,Posicion,n)*

Longitud (TID, UFID) → (l) - INFORME (UFIDmal, Abortada)

- Devuelve la nueva longitud provisional de *UFID*, resultante de *TID*, si ésta ha registrado alguna. Si no, tiene el mismo efecto que *Longitud (UFID)*

Trunca (TID, UFID, l) - INFORME (UFIDmal, Abortada)

- Trunca *UFID* de una manera provisional, pendiente de la terminación de *TID*

Crea (TID) → (UFID) - INFORME (Abortada)

- Crea el Fichero *UFID* de manera provisional, pendiente de la terminación de *TID*

Borra (TID, UFID) - INFORME (UFIDmal, Abortada)

- Borra *UFID* de una manera provisional, pendiente de la terminación de *TID*

Tema 7

Procesamiento de transacciones

29



Fases en un Servicio con soporte de Transacciones

• Dos fases con tres estados:

– *fase 1:*

- desde: *AbreTransaccion*
⇒ estado *provisional*
- hasta: *CierraTransaccion*

– *fase 2:*

- desde: *CierraTransaccion*
– bien ⇒ estado *comprometido*
– mal ⇒ estado *abortado*
- hasta: el final

Tema 7

Procesamiento de transacciones

30



Métodos de Control de la Concurrency

Tema 7

Procesamiento de transacciones

31



Bloqueos. 1

- Es el método más utilizado en la práctica
- Evitan inconsistencias y pérdida de actualizaciones
- Peligro: abrazo mortal (*deadlock*)
- Realización. Gestor de bloqueos que mantiene un registro para cada dato con:
 - Identificador(es) de la(s) transacción(es) que mantiene(n) el bloqueo
 - identificador del dato
 - tipo de bloqueo
 - variable de condición
- Decisión de diseño: *granularidad*

Tema 7

Procesamiento de transacciones

32



Bloqueos. 2

- Relación entre las 2 fases y los bloqueos:
 - 1ª fase: adquisición de bloqueos
 - 2ª fase: liberación una vez que la transacción se comprometa o aborte
- Bloqueo estricto:
 - antes de acceder al dato, hay que poner un cerrojo
 - ⇒ exclusión mutua sobre un recurso compartido
 - si el recurso ya está bloqueado, espera
 - si no, lo bloquea y accede
- Las operaciones sobre cerrojos son privilegiadas:
 - las ejecuta el SF como efecto lateral de sus ops.
 - el cliente no puede acceder a los cerrojos

Tema 7

Procesamiento de transacciones

33



Ejemplo de bloqueo estricto

| Transacción T: <i>Transferir 4 unidades de A a B</i> | | Transacción U: <i>Transferir 3 unidades de C a B</i> | |
|---|-----------------|---|---------------------------------------|
| <u>Operaciones</u> | <u>Cerrojos</u> | <u>Operaciones</u> | <u>Cerrojos</u> |
| AbreTransaccion a = A.Lee() | cierra A | AbreTransaccion c = C.Lee() | cierra C |
| A.Escribe(a-4) | | C.Escribe(c-3) | |
| b = B.Lee() | cierra B | b = B.Lee() | espera por el cerrojo de T sobre B |
| B.Escribe(b+4) | | ... | |
| CierraTransaccion | abre A y B | | cierra B |
| | | B.Escribe(b+3) | |
| | | CierraTransaccion | abre B y C |

Tema 7

Procesamiento de transacciones

34



Bloqueos de lectura y escritura

- Proveen el esquema
múltiples lectores/único escritor
- Dato bloqueado en escritura:
 - inaccesible desde otras transacciones
- Dato bloqueado en lectura:
 - accesible en lectura desde otras transacciones (bloqueo compartido)
 - inaccesible en escritura desde otras transacciones
 - la misma transacción puede convertirlo en bloqueo de escritura si el bloqueo no está compartido
- Mismo problema que en bloqueo estricto:
 - no evitan el *abrazo mortal* ni la *acaparación*

Tema 7

Procesamiento de transacciones

35



Compatibilidad de los bloqueos de lectura y escritura

| Bloqueo puesto | Bloqueo intentado | |
|----------------|-------------------|-----------|
| | lectura | escritura |
| ninguno | OK | OK |
| lectura | OK | espera |
| escritura | espera | espera |

Tema 7

Procesamiento de transacciones

36



Reglas de uso de los bloqueos de lectura y escritura

1. Cuando un cliente accede a un dato:
 - Si el dato no está bloqueado por otras transacciones, el servidor lo bloquea y accede al dato para el cliente
 - Si el dato tiene un bloqueo incompatible puesto por otra transacción, el cliente debe esperar a que se desbloquee
 - Si el dato tiene un bloqueo compatible puesto por otra transacción, el bloqueo se comparte y el servidor accede al dato para el cliente
 - Si una transacción intenta escribir un dato que ella, y sólo ella, ha bloqueado en lectura, el bloqueo de lectura se convierte en un bloqueo de escritura
2. Cuando una transacción se compromete o aborta:
 - el servidor desbloquea todos los datos que bloqueó para esa transacción

Tema 7

Procesamiento de transacciones

37



Abrazo mortal con bloqueos de lectura y escritura

| Transacción T: <i>Transferir 4 unidades de A a B</i> | | Transacción U: <i>Transferir 3 unidades de C a B</i> | |
|---|---|---|---|
| Operaciones | Cerrojos | Operaciones | Cerrojos |
| AbreTransaccion a = A.Lee() | cierra A en Lectura | AbreTransaccion c = C.Lee() | cierra C en Lectura |
| A.Escribe(a-4) | cierra A en Escritura | C.Escribe(c-3) | cierra C en Escritura |
| b = B.Lee() | cierra B en Lectura | b = B.Lee() | comparte el cerrojo en Lectura sobre B espera por el cerrojo en Lectura de T sobre B |
| B.Escribe(b+4) | espera por el cerrojo en Lectura de U sobre B | B.Escribe(b+3) | |
| <i>deadlock</i> | | <i>deadlock</i> | |

Tema 7

Procesamiento de transacciones

38



Solución del *abrazo mortal*

- Detección: búsqueda en un grafo de espera y aborto de una transacción para romper el ciclo
 - elección no es trivial
- Prevención. 2 métodos:
 - bloqueo previo (sobre los datos a utilizar)
 - Problemas:
 - Restricción de acceso a los recursos compartidos
 - No se sabe a priori qué datos se van a utilizar
 - bloqueo con orden predefinido
 - Problema:
 - Reducción de la concurrencia
- Híbrido: temporización (*timeout*)
 - es el más habitual
 - soluciona el *deadlock* y la *acaparación*

Tema 7

Procesamiento de transacciones

39



Solución del *deadlock* mediante temporizaciones

- Cuando vence la temporización \Rightarrow bloqueo vulnerable
 - Mientras nadie compita por el dato \Rightarrow sigue bloqueado
 - Si alguien compite \Rightarrow se rompe el cerrojo:
 - la transacción competidora continúa
 - la que tiene el bloqueo vulnerable se aborta
- Problemas:
 - Alguna transacción puede ser abortada cuando el bloqueo se hace vulnerable y hay otras transacciones esperando, aunque no haya *deadlock*
 - Si el sistema está sobrecargado, se penaliza a las transacciones que consumen mucho tiempo
 - Es difícil determinar la duración óptima de las temporizaciones

Tema 7

Procesamiento de transacciones

40

Resolución del *abrazo mortal*

| Transacción T: <i>Transferir 4 unidades de A a B</i> | | Transacción U: <i>Transferir 3 unidades de C a B</i> | |
|---|--|---|---|
| Operaciones | Cerrojos | Operaciones | Cerrojos |
| a = A.Lee() | cierra A en Lec → t(T,A) | c = C.Lee() | cierra C en Lec → t(U,C) |
| A.Escribe(a-4) | cierra A en Esc → t(T,A) | C.Escribe(c-3) | cierra C en Esc → t(U,C) |
| b = B.Lee() | cierra B en Lec → t(T,B) | b = B.Lee() | comparte el cerrojo en Lec sobre B → t(U,B) |
| B.Escribe(b+4) | espera por el cerrojo en Lectura de U sobre B | B.Escribe(b+3) | espera por el cerrojo en Lectura de T sobre B |
| ... | vence t(T,B) | ... | |
| | el cerrojo de T sobre B se hace vulnerable abre B, aborta T | | cierra B en Esc → t(U,B) |
| | | CierraTransaccion | abre B y C |

Tema 7

Procesamiento de transacciones

41



Bloqueos bi-versión

- El bloqueo de lectura impide las escrituras
 - innecesario para escrituras provisionales
- Podemos refinar el bloqueo de escritura mediante:
 - un bloqueo de *escritura*:
 - se pone cuando se solicita una escritura
 - compatible con bloqueos en lectura de otras transacciones
 - otro bloqueo de *compromiso*:
 - se pone cuando se solicita cerrar la transacción
 - sobre los datos bloqueados en escritura
 - incompatible con bloqueos en lectura de otras transacciones

Tema 7

Procesamiento de transacciones

42



Compatibilidad de los bloqueos bi-versión

| Bloqueo puesto | Bloqueo intentado | | |
|----------------|-------------------|-----------|------------|
| | lectura | escritura | compromiso |
| ninguno | OK | OK | OK |
| lectura | OK | OK | espera |
| escritura | OK | espera | ---- |
| compromiso | espera | espera | ---- |

Tema 7

Procesamiento de transacciones

43



Bloqueos bi-versión con temporizaciones

- Si una transacción tiene un bloqueo de escritura sobre un dato y
- otra transacción tiene uno de lectura sobre el mismo dato y
- la primera quiere cerrarse,
- el servidor espera hasta que:
 - la transacción lectora se cierre y libere su bloqueo
 - el servidor convierte el de escritura en uno de compromiso
 - o el bloqueo de lectura se vuelva vulnerable
 - el servidor lo rompe y convierte el de escritura en uno de compromiso
 - o se vuelva vulnerable el de escritura
 - el servidor aborta a la transacción propietaria del mismo

Tema 7

Procesamiento de transacciones

44



Ejemplo de bloqueos bi-versión

| Transacción T: <i>Transferir 4 unidades de A a B</i> | | Transacción U: <i>Transferir 3 unidades de C a B</i> | |
|---|--|---|---|
| Operaciones | Cerrojos | Operaciones | Cerrojos |
| a = A.Lee() | cierra A en Lec → t(T,A) | c = C.Lee() | cierra C en Lec → t(U,C) |
| A.Escribe(a-4) | cierra A en Esc → t(T,A) | C.Escribe(c-3) | cierra C en Esc → t(U,C) |
| b = B.Lee() | cierra B en Lec → t(T,B) | b = B.Lee() | comparte el cerrojo en Lec sobre B → t(U,B) |
| B.Escribe(b+4) | espera por el cerrojo en Escritura de U sobre B | B.Escribe(b+3) | cierra B en Esc → t(U,B) |
| ... | | CierraTransaccion | cierra C en Comp espera por el cerrojo en Lectura de T sobre B |
| ... | | ... | |
| ... | | | cierra B en Comp abre B y C |
| | vence t(T,B) el cerrojo de T sobre B se hace vulnerable abre B, aborta T | | |

Tema 7

Procesamiento de transacciones

45



Control Optimista de la Concurrency

Tema 7

Procesamiento de transacciones

46



Principales problemas del uso de bloqueos

- El mantenimiento de los bloqueos es una carga extra
- Incluso las transacciones de sólo lectura deben bloquear los datos
- El uso de bloqueos puede dar lugar al abrazo mortal
- Para evitar abortar transacciones en cascada, los bloqueos no se deben quitar hasta el final de la transacción
⇒ reducción de la concurrencia

Tema 7

Procesamiento de transacciones

47



Control Optimista de la Concurrency. 2

- Control optimista de la concurrencia:
 - es *optimista* porque supone que la probabilidad de que las transacciones concurrentes de clientes distintos accedan a los mismos datos es muy baja
 - útil sólo si dicha suposición es acertada
- La primera fase se denomina:
 - fase de *lectura*
- La segunda fase se divide en 2 sub-fases:
 - fase de *validación*
 - fase de *escritura o descarte*

Tema 7

Procesamiento de transacciones

48



Fase de Lectura

- Cada transacción que modifique datos tiene una versión provisional de los mismos
- Las operaciones de escritura guardan los nuevos valores de los datos como valores provisionales y aislados
- Las operaciones de lectura se realizan inmediatamente
- Cuando existen transacciones concurrentes accediendo a los mismos datos, pueden coexistir distintos valores provisionales de los mismos datos
- Se mantienen dos registros por cada transacción:
 - Conjunto de Lectura: guarda una relación de los datos leídos por la transacción
 - Conjunto de Escritura: guarda una relación de los datos escritos por la transacción

Tema 7

Procesamiento de transacciones

49



Sub-fases de validación y escritura o descarte

- Sub-fase de validación:
 - Al recibir la solicitud de *CierraTransaccion()*, la transacción se examinará para comprobar que sus operaciones no entran en conflicto con las operaciones de otras transacciones
- Sub-fase de escritura o descarte:
 - Si la transacción es válida, los cambios que había efectuado de forma provisional se convierten en permanentes y públicos

Tema 7

Procesamiento de transacciones

50



Proceso de validación

- Reglas de lectura/escritura para asegurar que la transacción es equivalente serie con otras transacciones con las que tenga solape en el tiempo
- A cada transacción T_i se le asigna un número de transacción (nt_i) al entrar en la sub-fase de validación
- Si la transacción se compromete, retiene el nº
- Los números de transacción son enteros secuenciales
- El número de una transacción define su posición en el tiempo: $nt_i < nt_j \Leftrightarrow T_i$ precede a T_j
- Métodos de validación:
 - Validación hacia atrás
 - Validación hacia adelante

Tema 7

Procesamiento de transacciones

51



Control Optimista. Reglas

- Para que una transacción T_{val} sea equivalente serie con respecto a otra T_i con la que se solapa en el tiempo, sus operaciones deben cumplir tres reglas:
 1. T_{val} no debe hacer escrituras permanentes durante la ejecución de T_i en datos en los que T_i haya leído
 2. T_{val} no debe leer datos en los que T_i haya hecho escrituras permanentes durante la ejecución de T_{val}
 3. Las escrituras inmediatas (sin lectura previa) deben hacerse en el orden de cierre de las transacciones
- Simplificación: sólo puede haber una transacción en la segunda fase al mismo tiempo.
 - ⇒ la regla 3 se satisfará siempre.

Tema 7

Procesamiento de transacciones

52



Validación hacia atrás

- Comparación con transacciones ya comprometidas
- Las escrituras de la transacción actual no van a ser vistas por las precedentes \Rightarrow la Regla 1 se cumple siempre
- La validación de una transacción se basa en comprobar que su conjunto de lectura no se solapa con ninguno de los conjuntos de escritura de las transacciones concurrentes precedentes (Regla 2)
 - sólo se comparan los registros de aquellas transacciones comprometidas que tengan algún solape temporal con la actual

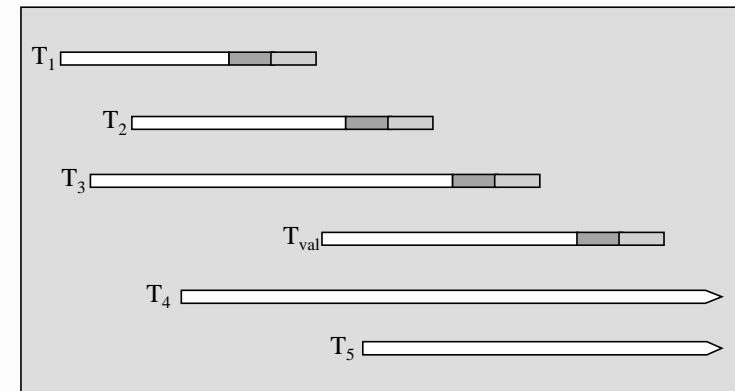
Tema 7

Procesamiento de transacciones

53



Validación en función del solapamiento



Tema 7

Procesamiento de transacciones

54



Validación hacia atrás. 2

- El único modo de resolver un conflicto es abortar la transacción que está siendo validada
- Las transacciones que sólo escriben no tienen que ser validadas
- Problema: hay que guardar los conjuntos de escritura de las transacciones más recientes
 - mientras quede alguna transacción en marcha con la que tengan algún solape temporal

Tema 7

Procesamiento de transacciones

55



Validación hacia adelante. 1

- Comparación con transacciones aún activas
- Las escrituras de las transacciones activas no han sido vistas por la actual \Rightarrow la Regla 2 se cumple siempre
- La validación de una transacción se basa en comprobar que su conjunto de escritura no se solapa con ninguno de los conjuntos de lectura de las transacciones activas (Regla 1)
 - todas las transacciones activas tienen algún solape temporal con la actual

Tema 7

Procesamiento de transacciones

56



Validación hacia adelante. 2

- Las transacciones de sólo lectura no tienen que ser validadas
- Hay dos estrategias para decidir qué transacción abortar en caso de conflicto:
 - abortar todas las transacciones activas conflictivas y comprometer la que está siendo validada
 - abortar la transacción que estamos validando
 - ¿y si después las conflictivas también tienen que abortar?

Tema 7

Procesamiento de transacciones

57



Comparación entre los métodos de validación

- Hacia adelante:
 - Flexibilidad en resolución de conflictos
 - En general los conjuntos de lectura son más grandes que los de escritura
 - Debe permitir la incorporación de nuevas transacciones y variaciones de conjuntos de transacciones activas mientras está en la sub-fase de validación
- Hacia atrás:
 - Almacenamiento de conjuntos de escritura de transacciones ya terminadas

Tema 7

Procesamiento de transacciones

58



Sellos temporales

Tema 7

Procesamiento de transacciones

59



Sellos temporales. 1

- Los sellos temporales definen la posición temporal en la secuencia de transacciones
 - sello con valor más bajo \Rightarrow transacción precedente
- Cada operación se valida cuando se ejecuta
- Reglas básicas:
 - una escritura es válida si la última lectura y la última escritura fueron hechas por transacciones precedentes
 - una lectura es válida si la última escritura fue hecha por una transacción precedente

Tema 7

Procesamiento de transacciones

60



Sellos temporales. 2

- A cada transacción se le asigna un sello:
 - cuando $AbreTransaccion()$
 - $t_i < t_j \Leftrightarrow T_i$ precede a T_j
- a cada dato se le asocia la siguiente información:
 - valor permanente (V_{perm}) y sello temporal del valor permanente (t_{perm})
 - conjunto de valores provisionales ($V_{prov}[i]$) y sellos temporales de las escrituras provisionales ($t_{prov}[i]$)
 - conjunto de sellos temporales de las lecturas ($t_{lect}[i]$)
- el sello temporal de una lectura o escritura es el sello temporal de la transacción que la generó

Tema 7

Procesamiento de transacciones

61



Sellos temporales. Escritura

- Si es válida, se crea un valor provisional
- El valor provisional:
 - se convertirá en permanente si el cliente ejecuta $CierraTransaccion()$
 - y, en consecuencia, la transacción se compromete
 - se descartará si la transacción se aborta
- No hace esperar al cliente

```
if (( $t_{val} \geq \max(t_{lect}[i])$ ) && ( $t_{val} > t_{perm}$ ))
  crear  $V_{prov}[val]$  y  $t_{prov}[val] = t_{val}$ 
else abortar  $T_{val}$ 
```

Tema 7

Procesamiento de transacciones

62



Sellos temporales. Lectura y Compromiso

- Una lectura puede tener que esperar por transacciones previas:

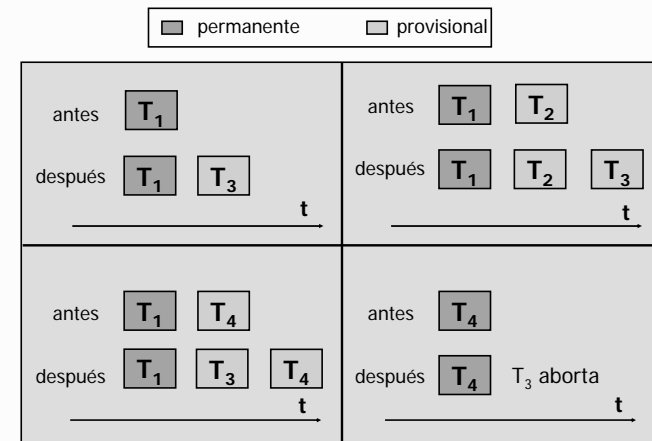
```
if ( $t_{val} > t_{perm}$ )
  while ( $\exists t_{prov}[i]$  tal que  $t_{prov}[i] < t_{val}$ )
    esperar hasta que todas las transacciones
    que pusieron los  $t_{prov}[i]$  se comprometan
    o aborten
  leer  $V_{perm}$ 
  else abortar  $T_{val}$ 
```

- Compromiso. Nunca hace esperar al cliente
 - el valor provisional y su sello pasan a ser permanentes
 - pero antes, el servidor puede tener que esperar a que se comprometan escrituras provisionales precedentes

Tema 7

Procesamiento de transacciones

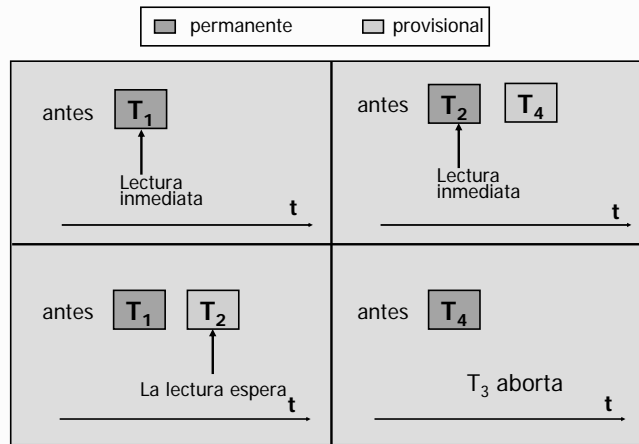
63

Ejemplo. Solicitud de escritura de T_3 

Tema 7

Procesamiento de transacciones

64

Ejemplo. Solicitud de lectura de T_3 

Tema 7

Procesamiento de transacciones

65



Comparación entre los métodos. 1

- Ordenamiento de las transacciones:
 - Bloqueos: orden de acceso a los datos conflictivos (dinámico)
 - Control optimista: orden según entrada en la fase de validación (*Cierra Transacción*)
 - Sellos temporales: orden según inicio de la transacción (*Abre Transacción*)
- Nivel de complejidad:
 - Los bloqueos son más sencillos

Tema 7

Procesamiento de transacciones

66



Comparación entre los métodos. 2

- Optimismo:
 - Los bloqueos y los sellos temporales emplean un enfoque pesimista: sólo dejan acceder al dato en ausencia de conflicto
 - El control optimista permite siempre el acceso a los datos, aunque finalmente se aborte la transacción
- Resolución de conflictos:
 - Los bloqueos hacen esperar a las transacciones
 - pueden abortar finalmente para evitar el *deadlock*
 - Los sellos temporales abortan a las transacciones *in situ*
 - Control optimista aborta al final

Tema 7

Procesamiento de transacciones

67



Comparación entre los métodos. 3

- Comportamiento según el tipo de transacciones:
 - El control optimista presenta un buen comportamiento cuando es cierta la suposición de que hay pocos conflictos
 - Si la probabilidad de conflicto no es tan baja:
 - Los bloqueos tienen un mejor comportamiento ante transacciones con una relación lecturas/escrituras moderada
 - Los sellos temporales tienen un mejor comportamiento ante transacciones con una relación lecturas/escrituras muy alta

Tema 7

Procesamiento de transacciones

68