Dpto. Ciencias e Ingeniería de la Computación
Universidad Nacional del Sur

Elementos de Bases de Datos

Segundo Cuatrimestre 2015

Clase 15:

Control de Concurrencia – Bloqueos y Estampillas de Tiempo

Mg. María Mercedes Vitturini
[mvitturi@uns.edu.ar]

Protocolos Basados en Bloqueos

Repaso

- Control de Concurrencia
- Protocolos basados en bloqueos vistos:
  - PBB
  - P2F
  - P2F Estricto
  - P2F Riguroso
  - Upgrade y downgrade
  - P2F Refinado
  - P2F Reginado + Riguroso

- Ejemplo

- Otra estrategia
  - Protocolo Árbol
  - Reglas
  - Ejemplo
  - Por qué funciona – Lectura asociada la clase

- Implementación del gestor de bloqueos

Control de Concurrencia (CC)

- Los **protocolos de control de concurrencia** son los mecanismos usados por el DBMS para garantizar aislamiento en ejecuciones concurrentes.
- Buscan:
  - Proporcionar *concurrencia*.
  - Generar *planificaciones serializables*.
  - Controlar el *acceso a los recursos compartidos*.

PCC basados en bloqueos

Protocolos de Control de Concurrencia Basados en Bloqueos (PBB): controlan el acceso concurrente a un ítem de dato usando *permisos* o bloqueos (*locks*).

- Existen dos tipos de permisos:
  - 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.

Protocolo Basado en Bloqueos

Protocolo Basado en Bloqueos (PBB)

- Imponen como única regla para acceder a un recurso obtener el *lock* apropiado antes.
- Los locks se requieren al *gestor de bloqueos* que determina si lo puede conceder o la transacción debe esperar.
- Los *locks* y *unlocks* se pueden hacer en cualquier momento.
- Este protocolo genera *planificaciones NO serializables*.

Protocolos de Bloqueos de 2 Fases

- **Protocolos de Bloqueos de Dos Fases (PB2F):** requieren que cada transacción realice sus solicitudes de bloqueo y desbloqueo en dos fases:
  - **Fase de Crecimiento**: Una transacción puede obtener nuevos bloqueos pero no puede liberar ningún bloqueo.
  - **Fase de Encogimiento**: Una transacción puede liberar bloqueos pero no puede obtener ningún bloqueo nuevo.

Variantes al PB2F

- *Protocolo de Bloqueo de Dos Fases* (**PB2F**)
- Protocolo de Bloqueo de Dos Fases *Estricto* (**PB2F Estricto**).
- Protocolo de Bloqueo de Dos Fases *Riguroso* (**PB2F Riguroso**).
- Protocolo de Bloqueo de Dos Fases *Refinado* (**PB2F Refinado**).

Ejemplo

Dadas las transacciones:

T_1 = read(A); A = A – 50; write(A); read(B); B = B + 50; write(B);

T_2 = read(A); read(B); print (A+B);

T_3 = read(C); read(B); C = B ; write(C);

Se pide

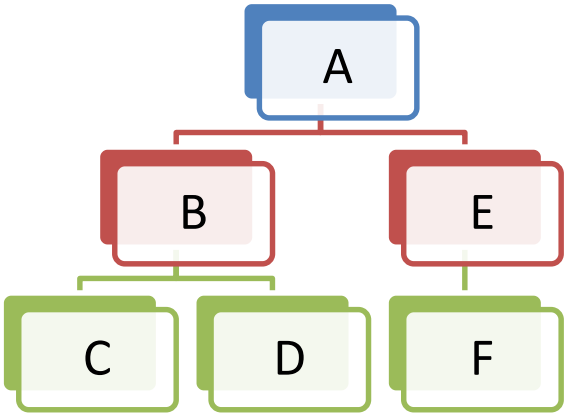
i. Generar una planificación concurrente *no serializable* bajo el PBB.

ii. Generar una planificación concurrente bajo **PB2F**. ¿Es serializable? ¿Es serializable en conflictos? Dar la serie equivalente.

iii. Generar una planificación concurrente bajo **PB2F refinado**. Dar la serie equivalente.

iv. Generar una planificación concurrente bajo **PB2F riguroso**. Dar la serie equivalente.

Protocolo de Árbol



Un protocolo de control de concurrencia basado en bloqueos libre de deadlock

Protocolos basados en grafos

- Son una alternativa de protocolos para control de concurrencia basados en bloqueos.
- Imponen un **orden parcial sobre el conjunto D de ítems de datos**
  - D = $\{d_1, d_2. \ldots, d_n\}$, forma un grafo no cíclico.
  - Si en D $d_i \rightarrow d_j$ entonces, cualquier transacción que necesite acceder a los datos d_i, d_j. debe hacerlo respetando ese orden.

Protocolo del árbol

REGLAS

- El único tipo de bloqueo permitido es *lock-X*.
- Dada una transacción T_i, su primer bloqueo puede ser sobre cualquier dato.
- Después del primer bloqueo, T_i puede bloquear un dato Q sólo si T_i bloquea actualmente al padre de Q.
- Los datos pueden desbloquearse en cualquier momento.
- Si T_i bloqueó y desbloqueó un dato Q, no puede volver a bloquearlo.

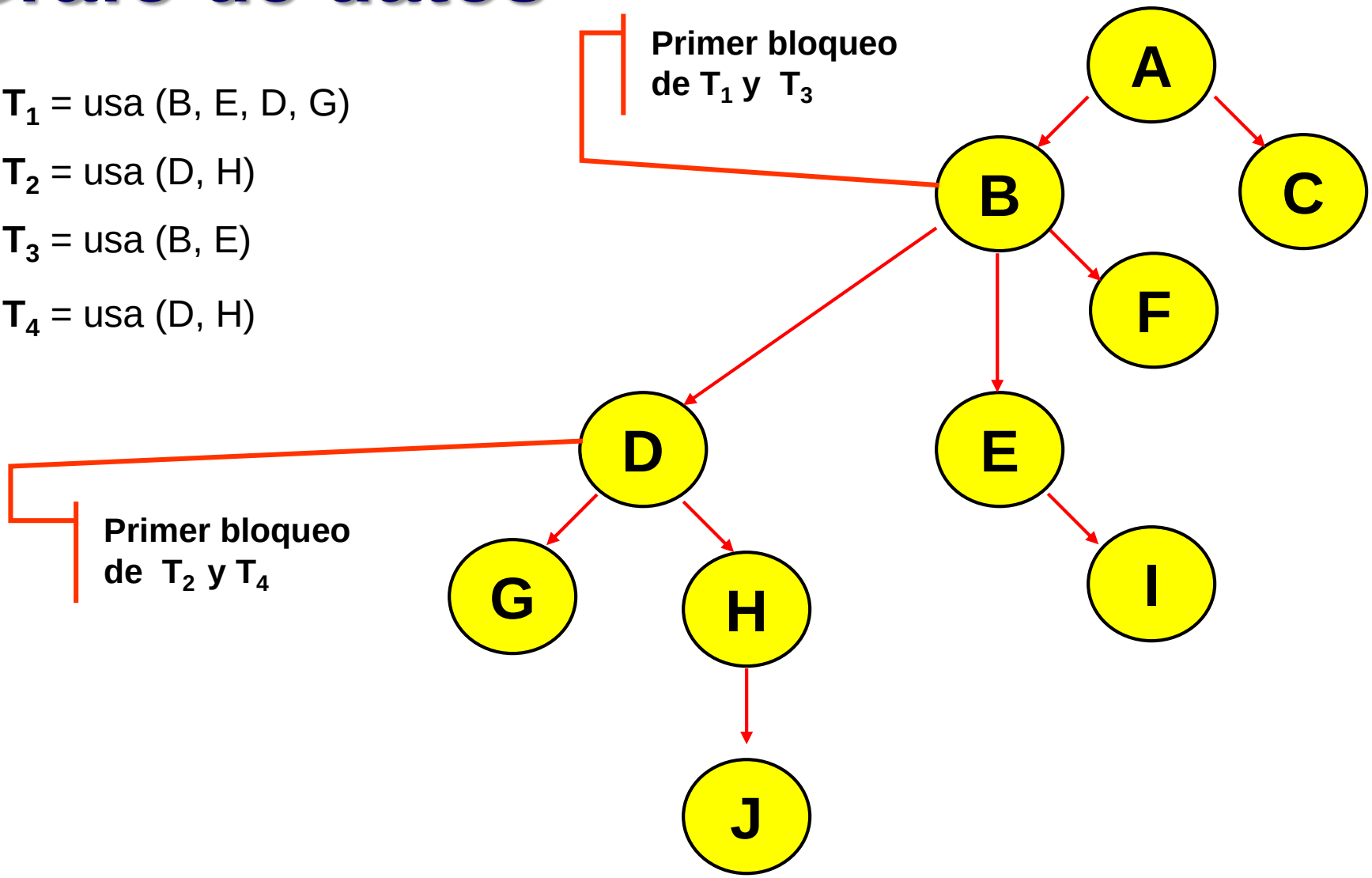
Grafo de datos

T_1 = usa (B, E, D, G)

T_2 = usa (D, H)

T_3 = usa (B, E)

T_4 = usa (D, H)



Planificación Serializable

T_1	T_2	T_3	T_4
Lock-X(B)			
	Lock-X(D)		
	Lock-X(H)		
	Unlock(D)		
Lock-X(E)			
Lock-X(D)			
Unlock(B)			
Unlock(E)			
		Lock-X(B)	
		Lock-X(E)	
	Unlock(H)		
Lock-X(G)			
Unlock(D)			
			Lock-X(D)
			Lock-X(H)
			Unlock(D)
			Unlock(H)
		Unlock(E)	
		Unlock(B)	
Unlock(G)			

Análisis del protocolo del árbol

Ventajas

☺ Asegura serializabilidad en conflictos y está libre de deadlocks.

☺ Combina secuencias de bloqueos y desbloqueos, "*aumentando la concurrencia*".

Desventajas

☹ No está libre de retrocesos en cascada y planificaciones no recuperables.

☹ Requiere que se bloqueen datos que no se necesitan, "*disminuye la concurrencia*".

Implementación de Bloqueos

- El **gestor de bloqueos** es un proceso independiente al que las transacciones envían sus *pedidos de lock*.
- Mantiene una *tabla de locks* para llevar registro de los locks cedidos y pendientes.
- La tabla generalmente se implementa como una tabla hash en memoria.
- A un pedido de bloqueo contesta con un mensaje de *'grant'*. Eventualmente ante un deadlock envía un mensaje de rollback.
- Las transacciones que solicitan un bloqueo se quedan en espera.

Protocolos basados en Estampilla de Tiempo

Protocolos de control de concurrencia basados en la estrategia “Hora de Entrada” de cada transacción

Protocolos basados en Hora de Entrada

- Protocolo basados en Hora de Entrada
- Estampillas: de transacción y de datos
- Protocolos
  - Protocolo Basado en Hora de Entrada (PBHE)
  - Ejemplos
  - PBHE + Regla de Escritura de Thomas

Protocolos basados en hora de entrada

- Los **protocolos basados en hora de entrada** *(time-stamp)* resuelven los conflictos realizando un esquema de ordenamiento según una *hora de entrada* u *hora de inicio de cada transacción*.
- Así, cuando T_i ingresa, se le asigna una hora $ts(T_i)$. Si más tarde ingresa una nueva transacción T_j entonces:

$$ts(T_i) < ts(T_j).$$

- ¿Cómo de implementar el esquema de hora?
  - Usando el *reloj del sistema*.
  - Usando un *contador lógico*.

Estrategia “Hora de Entrada”

- A cada transacción tiene su hora de entrada que es única en el sistema.
- Los conflictos se resuelven en función de la *“antigüedad”* de cada transacción.
- A diferencia de los protocolos de bloqueo, los conflictos no resuelven con esperas sino con “retrocesos”.
- Las planificaciones serializables coinciden con series que están definidas según algún *ordenamiento por hora de entrada*.

Protocolo basado en hora de entrada (PBHE)

- En los protocolos basados en hora de entrada (PBHE), se manejan las ejecuciones concurrentes de manera que *las estampillas de tiempo de las transacciones determinan el orden de serializabilidad.*

- Esto es, si ***ts*** **(T_i) < *ts* (T_j)** entonces el sistema debe asegurar que la planificación producida *sea equivalente a la planificación serie* en la cual **T_i** precede a **T_j.**

PBHE

- El protocolo PBHE requiere que para cada dato Q se mantengan **dos etiquetas de tiempo:**
  - **R-ts (Q)**, registra la *mayor etiqueta de tiempo de una transacción que ejecutó exitosamente un Read (Q)* (esto es, la lectura más reciente).
  - **W-ts (Q)**, mantiene la *mayor etiqueta de tiempo de una transacción que ejecutó exitosamente un Write (Q)* (esto es, la escritura más reciente).
- Las etiquetas de tiempo de Q: R-ts(Q) y W-ts(Q) potencialmente se pueden actualizar cada vez que se realiza un Read(Q) o un Write(Q).

PBHE – Controles de Lectura

Supongamos que T_i desea realizar un **Read(Q)**, el control consiste:

- Si ***ts*(T$_i$) < *W-ts*(Q)** entonces T_i necesita leer un valor de Q que ya fue escrito. Por lo tanto, la operación Read es rechazada, la transacción T_i

  retrocede

- Si ***ts*(T$_i$) ≥ *W-ts*(Q)** entonces la operación Read se ejecuta exitosamente. El valor de *R-ts*(Q) será el MAX (*R-ts*(Q), *ts*(T_i))

Qué ocurre en los protocolos de bloqueo?

PBHE – Controles de Escritura

Supongamos que T_i requiere un **Write(Q),** control:

- Si ***ts*(T$_i$) < *R-ts*(Q)** entonces el valor de Q que T_i está produciendo fue requerido previamente y el sistema asumió que nunca se produciría. Por lo tanto, la operación Write es rechazada y la transacción T_i *retrocede*.
- Si ***ts*(T$_i$) < *W-ts*(Q)** entonces T_i está intentando escribir un valor obsoleto de Q. Por lo tanto, la operación Write es rechazada y la transacción T_i *retrocede*.
- En otro caso, la operación Write se ejecuta exitosamente y *W-ts*(Q) toma el valor *ts*(T_i).

Planificación 1

$ts(T_1) < ts(T_2)$

T_1	T_2
1.Read(B)	
	2.Read(B)
	3.B:=B-50
	4.Write(B)
	5.Read(A)
6.Read(A)	
7.Display(A+B)	
	8.A:=A+50
	9.Write(A)
10.Display(A+B)	

<Q,*R-ts*(Q),*W-ts*(Q)>

<B,-∞,-∞>

<B,*ts*(T_1),-∞> (1)

<B,*ts*(T_2),-∞> (2)

<B,*ts*(T_2),*ts*(T_2)>(4)

<A,-∞,-∞>

<A,*ts*(T_2),-∞> (5)

<A,*ts*(T_2),-∞> (6)

<A,*ts*(T_2),*ts*(T_2)> (9)

Planificación 2

$ts(T_1) < ts(T_2)$

T_1	T_2
1.Read(B);	
	2.Read(B);
	3.B:=B-50;
	4.Write(B);
	5.Read(A);
	6.A:=A+50
	7.Write(A);
8.Read(A);	
9.Display(A+B)	
	Display(A+B)

<Q,*R-ts*(Q),*W-ts*(Q)>

<B,-∞,-∞>

<B,*ts*(T_1),-∞> (1)

<B,*ts*(T_2),-∞> (2)

<B,*ts*(T_2),*ts*(T_2)> (4)

<A,-∞,-∞>

<A,*ts*(T_2),-∞> (5)

<A,*ts*(T_2),*ts*(T_2)> (7)

En (8) se determina que T_1 debe retroceder pues necesita leer un valor que ya fue sobre-escrito.

Análisis del PBHE

- Asegura que para cualquier par de instrucciones *Read* y Write que están en conflicto se ejecuten según el orden de entrada.
- Garantiza seriabilidad de conflictos.
- Está libre de deadlocks pues las transacciones nunca esperan.
- No está libre de retrocesos en cascada y de generar planificaciones no recuperables.

PBHE

- Planificaciones posibles mediante el Protocolo de Bloqueo de Dos Fases no son posibles bajo el Protocolo de Ordenamiento por Hora de Entrada?
- y viceversa?. ¿Por qué?

- Cuando no se explicite el orden de entrada, se asignará como hora de entrada a cada transacción el instante previo a su primera instrucción.

Planificación 3

T_1	T_2
Read(A);	
	Write(A).
Write(A).	

$ts(T_1) < ts(T_2)$

<A,-∞,-∞>

<A,$ts(T_1)$,-∞>

<A,$ts(T_1)$,$ts(T_2)$>

T_1 intenta escribir un valor obsoleto de A; por lo tanto, deberá retroceder.

Usando el Protocolo de Ordenamiento por Hora de Entrada existen casos con retrocesos innecesarios.

Regla de Escritura de Thomas

Supongamos que **T_i** realiza un **Write(Q).** Control:

- Si ***ts*(T_i) < *R-ts*(Q)** entonces el valor de Q que T_i está produciendo fue requerido previamente y el sistema asumió que nunca se produciría. Por lo tanto, la operación Write es rechazada y la transacción T_i *retrocede*.

- Si ***ts*(T_i) < *W-ts*(Q)** entonces T_i está intentando escribir un valor obsoleto de Q. Por lo tanto, esta operación puede ser ignorada.

- En otro caso, la operación Write se ejecuta exitosamente y *W-ts*(Q) toma el valor *ts*(T_i).

PBHE + Regla de Escritura de Thomas

- Es una variante al PBHE.
- Las reglas para la operación Read no varían.
- Se modifican las reglas para la operación Write.
- Se eliminan retrocesos innecesarios producidos por escrituras obsoletas.
- Planificaciones serializables en este protocolo no lo son bajo otros protocolos.

Para pensar ...

- Los protocolos de estampilla de tiempo pueden tener problemas de:
  - Deadlock?
  - Retrocesos en cascada?
  - Planificaciones no recuperables?
  - Inanición?

Temas de la clase de hoy

- Protocolos de control de concurrencia
  - Basados en Bloqueos
    - Protocolos de Dos Fases y Variaciones – Ejemplos
    - Protocolo de Árbol.
  - Basados en Hora de Entrada
    - Time Stamping,
    - Time Stamping + Regla de Escritura de Thomas.
- **Bibliografía**
  - *Database System Concepts*– A. Silberschatz. Capítulo 15.
  - *DataBase System – The Complete Book* – H. Molina, J. Ullman. Capítulo 18.