Tema 3.2:
Control de la Concurrencia

Control de la Concurrencia

- Propiedad de aislamiento en las transacciones.
- Controlar la interacción de las transacciones $\Rightarrow$ Esquemas de control de la concurrencia
- Basados en secuencialidad.
- Tipos:
  - Basados en bloqueos.
  - Basados en marcas temporales.
  - Basados en validación.

Protocolos basados en Bloqueo

- Bloqueo $\Rightarrow$ mecanismo de control concurrente para el acceso a un elemento de datos.
- Dos modos de bloqueo:
  - Compartido (C o S) : transacción Ti bloqueo de sobre Q en modo C $\Rightarrow$ Ti puede leer Q pero no escribir.
  - Exclusivo (X) : transacción Ti bloqueo de sobre Q en modo X $\Rightarrow$ Ti puede leer y escribir Q.

- Gestor control de la concurrencia. La transacción procede después de la concesión del bloqueo.

Protocolos basados en Bloqueo

- Función de Compatibilidad ⇒ Matriz de Compatibilidad

	C	X
C	cierto	falso
X	falso	falso

- Un bloqueo se concede sobre un elemento si es compatible con los bloqueos actuales del elemento concedidos a otras transacciones.
- Cualquier número de transacciones pueden mantener bloqueos compartidos.
- Si cualquier transacción mantiene un bloqueo exclusivo, ninguna transacción puede mantener ningún otro bloqueo.
- Si una transacción transacción espera no puede acceder a un bloqueo ⇒ la hasta que todos los bloqueos incompatibles desaparezcan.

Protocolos basados en Bloqueo

Bloquear-X(B)
leer(B);
$B := B - 50$;
escribir(B);
desbloquear(B);
bloquear-X(A);
leer(A);
$A := A + 50$;
escribir(A);
desbloquear

Transacción T_1.

T_2: bloquear-C(A);
leer(A);
desbloquear(A);
bloquear-C(B);
leer(B);
desbloquear(B);
visualizar($A + B$).

Transacción T_2.

Protocolos basados en Bloqueo

T_1	T_2	Gestor de control de concurrencia
bloquear-X(B)		
		conceder-X(B, T_1)
leer(B)		
$B := B - 50$		
escribir(B)		
desbloquear(B)		
	bloquear-C(A)	
		conceder-C(A, T_2)
	leer(A)	
	desbloquear(A)	
	bloquear-C(B)	
		conceder-C(B, T_2)
	leer(B)	
	desbloquear(B)	
	visualizar($A + B$)	
bloquear-X(A)		
		conceder-X(A, T_1)
leer(A)		
$A := A + 50$		
escribir(A)		
desbloquear(A)		

Planificación 1.

Protocolos basados en Bloqueo

- Considerar

T_3: bloquear-X(B);
leer(B);
$B := B - 50$;
escribir(B);
bloquear-X(A);
leer(A);
$A := A + 50$;
escribir(A);
desbloquear(B);
desbloquear(A).

Transacción T_3.

T_4: bloquear-C(A);
leer(A);
bloquear-C(B);
leer(B);
visualizar($A + B$).
desbloquear(A);
desbloquear(B).

Transacción T_4.

T_3	T_4
bloquear-X(B)	
leer(B)	
$B := B - 50$	
escribir(B)	
	bloquear-C(A)
	leer(A)
	bloquear-C(B)
bloquear-X(A)	

. Planificación 2.

- Interbloqueo (deadlock). Sistema debe de retroceder una de las dos transacciones.
- Si no se usan bloqueos $\Rightarrow$ estados inconsistentes.
- Protocolo de Bloqueo $\Rightarrow$ conjunto de reglas seguido por todas las transacciones mientras de solicitan y sueltan bloqueos. Restringen el número de planificaciones posibles. (legales)

Protocolos basados en Bloqueo

- Ti precede a Tj: Ti $\rightarrow$ Tj,
  - Existe un elemento de datos Q , Ti ha obtenido bloqueo en modo A
  - Tj ha obtenido bloqueo en modo B
  - Comp(A,b)=falso.
- Si Ti $\rightarrow$ Tj , cualquier planificación secuencial equivalente, Ti debe aparecer antes que Tj.
- Inanición $\Rightarrow$ Transacción nunca progresa debido a bloqueos sucesivos de otras transacciones
- Se puede evitar:
  - No exista una transacción que posea un bloqueo sobre Q que esté en conflicto con el modo M
  - No haya otra transacción que esté esperando un bloqueo sobre Q y que lo haya solicitado antes.

Protocolos de Bloqueo de dos fases

- Protocolo que asegura la secuencialidad
- Dos fases:
  - Fase de crecimiento ⇒ Puede obtener bloqueos. No liberarlos
  - Fase de decrecimiento ⇒ Puede liberar bloqueos. No obtenerlos
- Transacciones T3 y T4.
- Transacciones T1 y T2 NO.
- Asegura la secuencialidad en cuanto a conflictos.
- Punto de bloqueo ⇒ punto donde la transacción adquiere el último bloqueo. ⇒ Ordenar transacciones por punto bloqueo.
- No asegura la ausencia de interbloqueos (Planificación 2)
- Puede ocurrir retroceso en cascada (Rollback en cascada)

Protocolos de Bloqueo de dos fases

- Ejemplo:

T_5	T_6	T_7
bloquear-X(A)		
leer(A)		
bloquear-C(B)		
leer(B)		
escribir(A)		
desbloquear(A)		
	bloquear-X(A)	
	leer(A)	
	escribir(A)	
	desbloquear(A)	
		bloquear-C(A)
		leer(A)

Protocolos de Bloqueo de dos fases

- Protocolo de bloqueo estricto de dos fases ⇒ evita retrocesos en cascada.
  - Dos fases + poseer todos bloqueos exclusivos hasta que termina.
- Protocolo de bloqueo riguroso de dos fases ⇒ poseer todos los bloqueos hasta comprometer la transacción.

T_8: leer(a_1);
leer(a_2);
...
leer(a_n);
escribir(a_1).

T_9: leer(a_1);
leer(a_2);
visualizar($a_1 + a_2$).

Protocolos de Bloqueo de dos fases

- Conversiones de bloqueo (refinado) ⇒ aumentar concurrencia
  - Fase Crecimiento:
    - Adquirir bloqueo C o Bloqueo X
    - Modo compartido ⇒ modo exclusivo (subir)
  - Fase Decrecimiento
    - Soltar bloqueo C o Bloqueo X
    - Modo exclusivo ⇒ modo compartido (bajar) (fase de decrecimiento)

T_8	T_9
bloquear-C(a_1)	
	bloquear-C(a_1)
bloquear-C(a_2)	
	bloquear-C(a_2)
bloquear-C(a_3)	
bloquear-C(a_4)	
	desbloquear(a_1)
	desbloquear(a_2)
bloquear-C(a_n)	
subir(a_1)	

Adquisición Automática de Bloqueos

- Generar automáticamente las instrucciones de bloqueo y desbloqueo para una transacción basándose en peticiones de lectura y escritura.

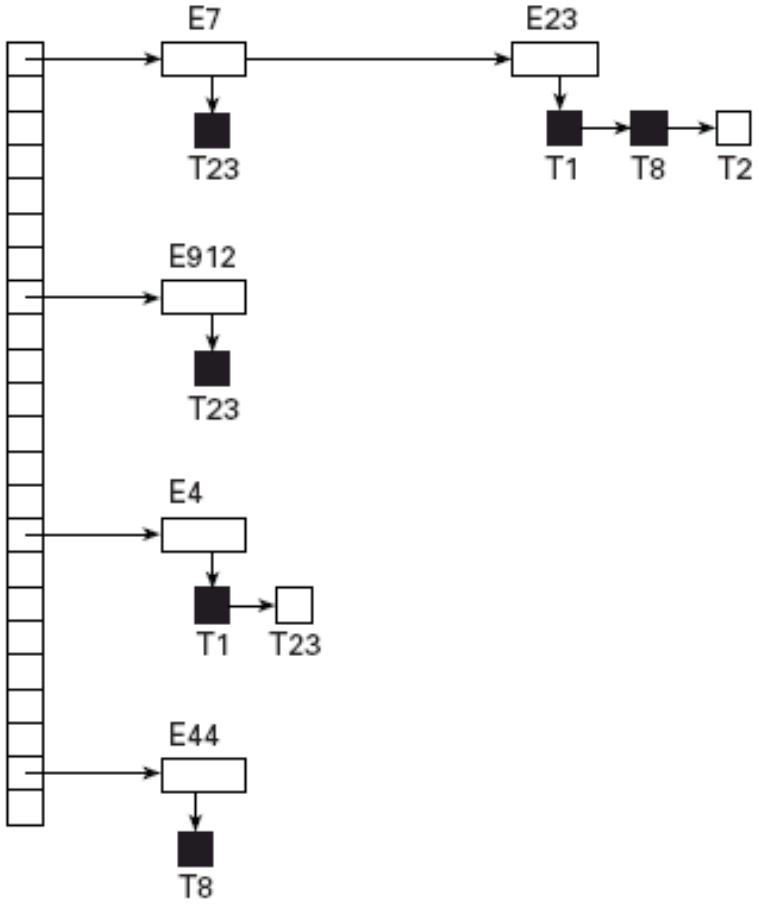
If T_i tiene un bloqueo en D
then
leer(D)
else
begin
if necesario esperar hasta ninguna transacción tenga un **bloqueo-X** sobre D
Permitir T_i un **bloqueo-C** sobre D;
leer(D)
end

Adquisición Automática de Bloqueos

if T_i tiene un **bloqueo-X** sobre *D*
then
escribir(*D*)
else
begin
if necesario esperar hasta ninguna transacción tenga cualquier bloqueo sobre *D*,
if T_i tiene un bloqueo**-C** sobre *D*
then
subir bloqueo sobre *D* a **bloqueo-X**
else
permitir T_i un **bloqueo-X** sobre *D*
escribir(*D*)
end;

Implementación de Bloqueos

- Gestor de Bloqueos



Tratamiento de Bloqueos

- Sistema está bloqueado si hay un conjunto de transacciones tal que cada transacción en el conjunto está esperando a otra transacción del conjunto.
- Protocolos de Prevención de Interbloqueos
- Estrategias:
  - Cada transacción bloquee todos sus elementos de datos antes de comenzar la ejecución (predeclaración)
  - Imponer un orden parcial a todos los elementos de datos y requerir que una transacción puede bloquear elementos en el orden especificado por el orden parcial. (protocolos basados en grafos)
  - Uso de expropiaciones y retrocesos de transacciones. Se utilizan marcas temporales a la transacción que se le ha expropiado y se sigue utilizando bloqueos para el control de la concurrencia.

Tratamiento de Bloqueos

- Esquema Esperar-Morir ⇒ Sin expropiación.
  - Transacciones más viejas pueden esperar a que las más jóvenes suelten los elementos. Las más jóvenes nunca esperan a las más viejas, estas se deshacen.
  - Una transacción puede morir varias veces antes de obtener los datos solicitados.
- Esquema Herir-Esperar ⇒ Con expropiación.
  - Transacciones más viejas hieren (fuerzan ROLL-BACK) de las transacciones más jóvenes en vez de esperar. Las más jóvenes pueden esperar a las más viejas
  - Puede tener menos roll-back que el esquema anterior.
- En ambos esquemas, las transacciones de reinician con su marca temporal original.
- Las más viejas tienen precedencia sobre las más nuevas ⇒ se evita la inanición.

Tratamiento de Bloqueos

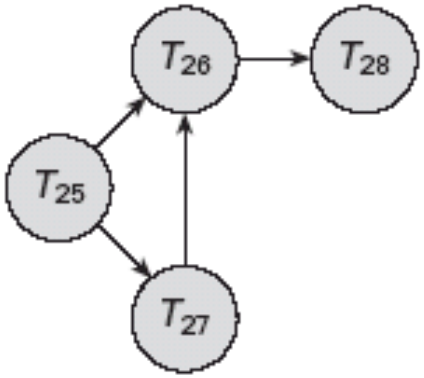
- Esquemas basados en límites de tiempo
  - Una transacción espera a un bloqueo una cantidad de tiempo. Después de ese tiempo se deshace.
  - No se dan interbloqueos
  - Simple de implementar.
  - Es posible la inanición.
  - Difícil de determinar el intervalo de tiempo.

Detección de Bloqueos

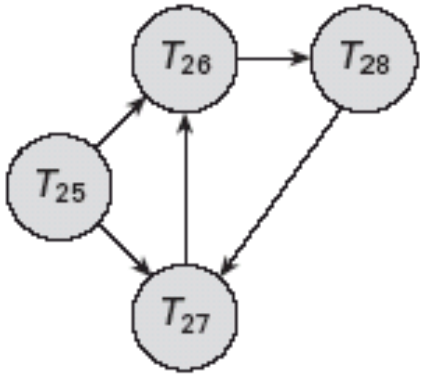
- Interbloqueos se pueden describir como grafos de espera.
- Consisten de un par $G = (V,E)$,
  - V es un conjunto de vértices (todas las transacciones)
  - E es un conjunto de arcos; par ordenado $T_i \rightarrow T_j$.
- Si $T_i \rightarrow T_j$ está en $E \Rightarrow$ hay un arco de T_i a T_j, donde T_i espera a que T_j suelte el elemento de datos .
- Cuando T_i pide un elemento de datos que tiene $T_j \Rightarrow$ un arco entre T_i T_j se inserta en el grafo. Se elimina cuando T_j suelta el elemento necesitado por T_i.
- El sistema está en interbloqueo si y solo si el grafo tiene un ciclo.
- Se debe de utilizar un algoritmo periódicamente para detectar los ciclos.

Detección de Bloqueos

$$T_{26} \rightarrow T_{28} \rightarrow T_{27} \rightarrow T_{26}$$



Grafo de espera sin ciclos.



Grafo de espera con un ciclo.

Recuperación de Bloqueos

- Cuando se detecta un interbloqueo:
  - Alguna transacción (víctima) será deshecha para romper interbloqueo. Seleccionar la transacción víctima que involucre menor coste.
  - Rollback → determinar cuanto hay que deshacer la transacción.
    - Total: Abortar y reiniciar.
    - Deshacer lo necesario para romper el interbloqueo.
  - Inanición sucede si siempre se elige la misma transacción como víctima. Incluir el número de rollbacks en el factor de coste.

Operaciones de Inserción y Borrado

- SI se usa un algoritmo de bloqueo dos fases:
  - Un borrado sólo se puede realizar si la transacción que borra la tupla tiene un bloqueo-X sobre la tupla a ser borrada.
  - Una transacción que inserta una nueva tupla se le da un bloqueo-X sobre la tupla.
- Inserciones y borrados pueden producir el fenómeno fantasma
  - Una transacción que lee una relación y una transacción que inserta una tupla en la relación pueden causar conflictos a pesar de no acceder a la tupla en común.
  - Si se utilizan sólo bloqueos de tuplas, pueden resultar planificaciones no secuenciables: la relación que lee puede no ver la nueva tupla, pudiendo ser secuenciable antes de la transacción que inserta.

Operaciones de Inserción y Borrado

- T1 lee la relación y T2 inserta información ⇒ información debería bloquearse.
- Solución:
  - Asociar elementos de datos con la relación.
  - T que lee, adquirir un bloqueo-C.
  - T que escribe, adquirir bloqueo-X.

- Producen una concurrencia baja para insertar/borrar

- Protocolos de bloqueo de índices proporcionan mayor concurrencia, previniendo efecto fantasma, proporcionando bloqueos sobre ciertos nodos del índice.