Transacciones concurrentes

jueves, 26 de octubre de 2023 14:36

Planificación: Orden de ejecución de transacciones.

En monousuario → se planifica en nivel transacciones. → no transacción A, después B.

En multiusuario se mezclan.

Entornos concurrentes

1.	READ(A)	
2.	A := A – 50	
3.	WRITE(A)	
4.		READ(A)
5.		TEMP := A * 0.1
6.		A := A – TEMP
7.		WRITE(A)
8.	READ(B)	
9.	B := B + 50	
10.	WRITE(B)	
11.		READ(B)
12.		B := B + TEMP
13.		WRITE(B)

BD A = 1000 3. 950 7. 945
B =2000 10.2050 13. 2145

To 1, A =1000
2 A = 950
8 B = 2000
9 B – 2050

T1 4 A = 950
5 temp = 95
6 A = 945
11 B = 2050
12 B = 2145

A + B se conserva

1.	READ(A)	
2.	A := A – 50	
3.		READ(A)
4.		TEMP := A * 0.1
5.		A := A – TEMP
6.		WRITE(A)
7.		READ(B)
8.	WRITE(A)	
9.	READ(B)	
10.	B := B + 50	
11.	WRITE(B)	
12.		B := B + TEMP
13.		WRITE(B)

BD A = 1000 6. 900 8.
B =2000 11.2050 13. 2

To 1, A =1000
2 A = 950
9 B = 2000
10 B = 2050

T1 3 A = 1000
4 temp = 100
5 A = 900
7 B = 2000
12 B = 2100

A + B no se conserva

7

Conservar consistencia implica q' suma cte = a suma antes de operar

- I1, I2 está en conflicto si actúan sobre el mismo dato y al menos una es un write.

Formas de controlar Concurrencia

Bloqueo → Compartido → cualquiera puede leer el dato.
Exclusivo → Solo lo uso yo mientras.

- Compartido *Lock_c*(dato)(solo lectura)
- Exclusivo *Lock_e*(dato) (lectura/escritura)
- Las transacciones piden lo que necesitan.
- Los bloqueos pueden ser compatibles y existir simultáneamente (compartidos)

Puede haber interbloqueo si se pide bloqueo compartido para q' terminen volver a pedir.

Deadlock

- situación en la que una transacción espera un recurso de otra y viceversa

- Si lo datos se liberan pronto → se evitan posibles deadlock
- Si los datos se mantienen bloqueados → se evita inconsistencia.

Protocolos de bloqueo

Depende de qué bloqueo primero.

- Dos fases
  - Requiere que las transacciones hagan bloqueos en dos fases:
    - Crecimiento: se obtienen datos
    - Decrecimiento: se liberan los datos
  - Garantiza seriabilidad en conflictos, pero no evita situaciones de deadlock.
  - Como se consideran operaciones
    - Fase crecimiento: se piden bloqueos en orden: compartido, exclusivo
    - Fase decrecimiento: se liberan datos o se pasa de exclusivo a compartido.

Protocolo basado en hora de entrada

- El orden de ejecución se determina por adelantado, no depende de quien llega primero
- C/transacción recibe una HDE
  - Hora del servidor
  - Un contador
- Si HDE(Ti) < HDE(Tj), Ti es anterior
- C/Dato
  - Hora en que se ejecutó el último WRITE
  - Hora en que se ejecutó el último READ
  - Las operaciones READ y WRITE que pueden entrar en conflicto se ejecutan y eventualmente fallan por HDE.

- **Ti Solicita READ(Q)**
  - HDE(Ti) < HW(Q): rechazo (solicita un dato que fue escrito por una transacción posterior)
  - HDE(Ti) ≥ HW(Q): ejecuta y se establece HR(Q)=Max{HDE(Ti), HR(Ti)}
- **Ti solicita WRITE(Q)**
  - HDE(Ti) < HR(Q): rechazo (Q fue utilizado por otra transaccion anteriomente y suposu que no cambiaba)
  - HDE(Ti) < HW(Q): rechazo (se intenta escribir un valor viejo, obsoleto)
  - HDE(Ti) > [HW(Q) y HR(Q)]: ejecuta y HW(Q) se establece con HDE(Ti).
- **Si Ti falla, y se rechaza entonces puede recomenzar con una nueva hora de entrada.**

→ nuevo pedido.

Bitácora

- Similar sistemas monousuarios
- Como proceder con checkpoint
  - Colocarlo cuando ninguna transacción esté activa. Puede que no exista el momento.
  - Checkpoint<L> L lista de transacciones activa al momento del checkpoint.
- Ante un fallo
  - UNDO y REDO según el caso.
  - Debemos buscar antes del Checkpoint solo aquellas transacciones que estén en la lista.