Controllo della concorrenza (ripasso generale)

- Supponiamo di operare su una macchina con una sola CPU e che questa CPU sia single core. Se vogliamo eseguire più operazioni queste dovranno essere schedulate per avere tempi di risposta accettabili.

Operazioni-Transazioni-Schedule

Una transazione è un agglomerato di operazioni ATOMICHE, cioè che devono essere eseguite senza interruzioni. Chiamiamo schedule un insieme di transazioni che devono essere eseguite dalle CPU.

- Le transazioni devono rispettare 3 PROPRIETÀ

DURABILITÀ: ogni Transazione deve avere un effetto sulla base di dati che si mantenga nel tempo

CONSISTENZA: le Transazioni operano su basi di dati in stato legale e le lasciano in stato legale

ISOLAMENTO: ogni Transazione è indipendente dalle altre

- Le operazioni delle transazioni possono essere interfogliate ma l'interfogliamento deve rispettare l'ordine interno di ogni transazione

- ESEMPIO:

$T_1: i_1, i_2, i_3$
$T_2: o_1, o_2, o_3$

$schedule(T_1, T_2) = (i_1, i_2, i_3, o_1, o_2, o_3)$ SCHEDULE VALIDO.
$schedule(T_1, T_2) = (i_1, o_2, i_2, i_3, o_1, o_3)$ SCHEDULE NON VALIDO.
$schedule(T_1, T_2) = (i_1, o_1, i_2, o_2, o_3, i_3)$ SCHEDULE VALIDO

in particolare il primo esempio è detto SCHEDULE SERIALE, un interfogliamento è corretto se lo SCHEDULE È SERIALIZZABILE (vedremo in seguito cosa significa.

Errori più comuni sui dati.

Aggiornamento mancato

T_1	T_2
read(X)	
X=X-N	
	read(X)
	X=X+M
write(X)	
read(Y)	
	write(X)
Y=Y+N	
write(Y)	

il Dato X modificato da T_1 non viene scritto in memoria quando T_2 lo va a leggere, T_1 scrive dopo che T_2 ha modificato X e T_2 sovrascrive X appena scritto da T_1

Dato sporco

T_1	T_2
read(X)	
X=X-N	
write(X)	
	read(X)
	X=X+M
read(Y)	
ERRORE	
	write(X)

In questo caso il dato X viene modificato da T_1, Questa transazione però viene abortita ma x è stato modificato

Aggregato sbagliato

T_1	T_2'
	S=0
read(X)	
X=X-N	
write(X)	read(X)
	S=S+X
	read(Y)
	S=S+Y
read(Y)	
Y=Y+N	
write(Y)	

Con lo schedule (T_1, T_2) si vuole avere $S = (Y+X)$ ma, a causa di un interfoglia-mento errato si è avuto un aggregato errato.

Lock a due stati

per implementare una lock a due stati, ogni variabile è dotata di un item binario questo item può essere lock o unlock.
quando una transazione deve lavorare su un dato esegue una lock su quel dato, così le altre transazioni non potranno usare quel dato né in lettura, né in scrittura.

possiamo associare ad una coppia lock-unlock una funzione con stati in ingresso i stati su cui la transazione ha eseguito una lock.

T_1	
lock(X)	
unlock(X)	$\rightarrow f_1(X)$
lock(Y)	
unlock(Y)	$\rightarrow f_2(X,Y)$

come capire se uno schedule è serializzabile.

T_1	T_2
lock(X)	
unlock(X)	
	lock(Y)
	unlock(Y)
lock(Y)	
unlock(Y)	
	lock(X)
	unlock(X)

$X = f_1(X_0)$

$Y = f_3(Y_0)$

$Y = f_4(f_3(Y_0), X_0)$

$X = f_2(Y_0, f_1(X_0))$

$X = f_4(f_3(Y_0), X_0)$

$Y = f_2(Y_0, f_1(X_0))$

se esiste uno schedule seriale con le stesse espressioni $\Rightarrow$
$\Rightarrow$ lo schedule in questione è serializzabile altrimenti NO.

c'è un modo più veloce e meno dispendioso per verificare ciò.

© GRAFO DI SERIALIZZAZIONE

le transazioni sono rappresentate con dei pallini e ponte una freccetta da una transazione ad un'altra se quest'ultima fa richiesta per una lock su un dato che le prima ha unlockato.

se il grafo in questione presenta un ciclo ⇒ lo schedule non è serializzabile.

PROTOCOLLO A DUE FASI

Una transazione rispetta il protocollo a due fasi se prima esegue tutte le operazioni di lock e poi tutte le operazioni di unlock

- Se uno schedule ha solo transazioni che rispettano il protocollo a due fasi ⇒ è serializzabile.

Dimostrazione:

Supponiamo, per assurdo che uno schedule sia composto da solo transazioni che rispettano il protocollo a due fasi e che nel grafo di serializzazione ci sia un ciclo. Per formare un ciclo, una transazione T_j deve fare richieste di lock su un dato che T_i ha unlockato e T_i deve fare richieste di lock su un dato che T_j ha unlockato ma questo entra in contraddizione con l'ipotesi di avere solo transazioni che rispettano il protocollo a due fasi ⇒ lo schedule è serializzabile.

Lock a tre stati

per implementare le lock a 3 stati, ogni dato avrà un item che potrà assumere i seguenti stati: 1) lock in lettura. 2) lock in scrittura. 3) unlock.

le transazioni possono compiere operazioni di rlock, unlock e rlock. Se un dato è wlockato allora questo non potrà essere né letto né scritto. Se invece è rlockato potrà essere letto dalle altre transazioni ma non modificato.

NUOVO GRAFO DI SERIALIZZAZIONE

"$\xrightarrow{x}$" che collega 2 transazioni si scrive se:

1) una transazione T_i esegue una r-lock o una w-lock su x e un'altra transazione T_j esegue una w-lock su x
2) una transazione T_i esegue una w-lock su x, una transazione T_j esegue una r-lock su x prima di eseguire una w-lock su x.

Deadlock - Liverlock/starvation

- **Deadlock**

Si si trova in una situazione di deadlock quando quando delle Transazioni sono bloccate in fase di stallo perché ognuna di queste sta aspettando che un dato venga unlockato.

- PER CAPIRE CHÉ SI È IN UNA SITUAZIONE DI DEADLOCK basta costruire un grafo dove le transazioni sono dei pallini e le queste (T_i, T_j) sono collegate da una freccia se T_i richiede una lock su un dato in uso da T_j, se troviamo un ciclo su questo grafo ⇒ lo schedule presenta deadlock.

Prevenire il deadlock

Bolinare i dati disponibili.

Liverlock

Una transazione può "trovarsi" in fase di starvation, cioè può richiedere una lock all'infinito senza che le sue richieste sia mai soddisfatte, per risolvere tale problema basta implementare una coda (FIFO) o un meccanismo basato sulle priorità (TEMPO o ESTERNA per esempio)

dati sporchi, Roll back e punti di commit

- un Roll-Back può avvenire per vari motivi
1) errore Hardware o Software.
2) Dead-lock.
3) operazione non corretta (DIVISIONE PER 0).

si cerca di evitare il più possibile di eseguire un Roll-Back, perché molto DISPENDIOSO.
evitabile in casi 3.
in caso 2 è evitabile con il punto di commit.
una Base di dati si trova nel suo puto di commit quando tutte le sue transazioni hanno richiesto le loro lock.

DATI SPORCHI

Un dato è detto sporco quando viene modificato e scritto da una transazione prima che la base di dati si trovi nel suo punto di commit.

PROTOCOLLO A 2 FASI STRETTO

PROTOCOLLO A 2 FASI PIÙ RESTRITTIVO

Una transazione rispetta il protocollo a due fasi stretto quando:
1) non esegue operazioni finché la base di dati non è in PUNTO DI COMMIT.
2) non rilascia i dati finché non ha finito di scrivere sulle base di dati