Lezione 27– Lock a tre valori

Prof.ssa Maria De Marsico
demarsico@di.uniroma1.it

SAPIENZA
UNIVERSITÀ DI ROMA

Parte del materiale è tratto dal materiale preparato dalla prof.ssa Moscarini per il canale A-L

Lock a tre valori

Una transazione può accedere ad un item **solo per leggerlo**, senza modificarlo

- Se una transazione desidera **solo leggere** un item *X* effettua una *rlock(X)* che **impedisce a qualsiasi altra transazione di modificare** *X*, ma **non di leggere** *X*
- Se una transazione **desidera modificare** un item *X* effettua un *wlock(X)*; in tal caso **nessuna altra transazione può leggere o modificare *X***
- **Entrambi** i lock sono rilasciati mediante una *unlock(X)*

Lock a tre valori
rlocked, wlocked, unlocked

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

Valore lock su X	T vuole fare un	Risultato
unlocked	*rlock(X)*	T ottiene il lock in **lettura**, valore variabile rlocked
rlocked	*rlock(X)*	T ottiene il lock in lettura
wlocked	*rlock(X)*	T **aspetta**
unlocked	*wlock(X)*	T ottiene il lock in scrittura, valore variabile wlocked
rlocked	*wlock(X)*	T **aspetta**
wlocked	*wlock(X)*	T **aspetta**

Osservazione

- Il modello a tre valori consente un grado **più alto** di **concorrenza**

- Un **qualsiasi** numero di transazioni può ottenere **contemporaneamente** un lock di **lettura** su *X*.

- Una transazione che mantiene un lock di **lettura** su un certo item **può richiedere un lock in scrittura** su quello **stesso** item (così impedisce ad altri di **leggere**, mentre altri scrittori erano già bloccati)

Modello per le transazioni

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

Una transazione è una sequenza di operazioni di *rlock, wlock* e *unlock*

- ogni *rlock(X)* e ogni *wlock(X)* implicano la **lettura** di *X*
- ogni *unlock(X)* **associato** a una ***wlock(X)*** implica la **scrittura** di *X*

- l'insieme degli item **scritti** da una transazione è **contenuto** nell'insieme degli item **letti** dalla stessa transazione.

Modello per le transazioni

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

T_1
rlock(X)
unlock(X)
wlock(Y)
unlock(Y) $f_1(X,Y)$

Il nuovo valore di un item viene calcolato da una funzione che

- è associata in modo univoco ad ogni **coppia wlock-unlock**
- ha per argomenti **tutti** gli item **letti** (**rlocked o wlocked**) dalla transazione prima dell'operazione di unlock

Equivalenza

Poiché in questo modello si assume che una transazione possa leggere un item **senza modificarlo**, la definizione di **equivalenza** di schedule deve essere **modificata** per tener conto di **tale eventualità**.

Due schedule sono ***equivalenti*** se

- producono lo stesso valore **per ogni item** su cui viene effettuato un ***wlock*** *(*le formule che danno i valori finali per ciascun item sono le stesse)
- **ogni** operazione ***rlock(X)*** legge **lo stesso valore di *X*** nei due schedule

Serializzabilità

- Uno schedule è serializzabile se è equivalente ad uno schedule seriale

Osservazioni

- Supponiamo che in uno schedule S una transazione T^1 effettui un'operazione ***wlock*** su un item *X* e che una transazione T^2 effettui un'operazione *rlock* su *X* **prima** che una **terza** transazione T^3 esegua **la successiva operazione di *wlock*** su *X* (in altre parole, T^1 **modifica** il valore di *X* e T^2 **legge** il valore di *X* prodotto da T^1 **prima che *X* venga nuovamente modificato** da T^3). Allora **in qualsiasi** schedule **seriale equivalente** ad S **T^1 deve precedere T^2 e T^2 deve precedere T^3.** D'altra parte, se due transazioni T^1 e T^2 **leggono entrambe il valore di un item *X*** prodotto da una transazione **non è lecito stabilire nessuna precedenza tra T^1 e T^2 .**
- Per rappresentare le **precedenze** tra le transazioni è possibile usare, come per il modello precedente, un **grafo diretto** che ha per nodi le transazioni e ha un arco da una transazione T^i a una transazione T^j **se la semantica delle transazioni impone che T^i debba precedere T^j.**

Testare la serializzabilità

Algoritmo

Dato uno schedule *S*

- ***Passo 1***
- crea un grafo diretto *G* (*grafo di serializzazione*)

  nodi: transazioni

  archi: T_i --> T_j (con etichetta X) **se** in *S*

  - T_i esegue una ***rlock(X)*** **o una *wlock(X)*** e T_j è la transazione **che esegue la successiva *wlock(X)***

  - T_i esegue una ***wlock(X)*** e T_j esegue **una *rlock(X)*** **dopo** che T_i ha eseguito **la *wlock(X)*** e **prima** che **un'altra** transazione esegua **una *wlock(X)***.

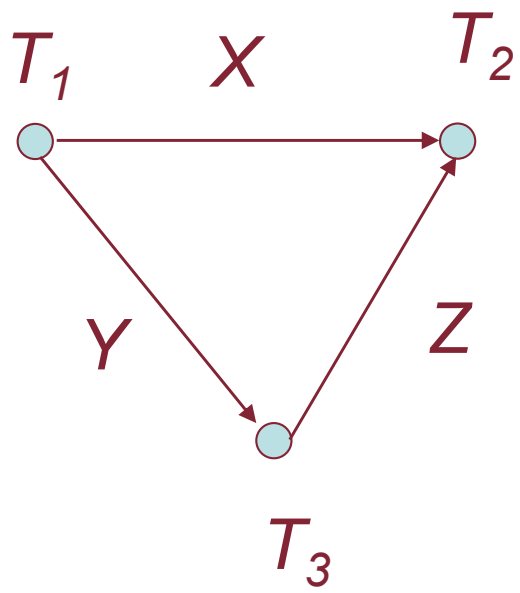
O T_j esegue la successiva **wlock** (dopo una rlock o una wlock, è indifferente) oppure esegue la **rlock tra due wlock (potrebbero esserci più rlock tra due wlock e quindi più archi che partono da** T_i)

Testare la serializzabilità

- ***Passo 2***
- Se *G* ha un ciclo allora *S* non è serializzabile; altrimenti applicando a *G* l'*ordinamento topologico* si ottiene uno schedule seriale *S'* equivalente ad *S*

T_1	T_2	T_3
rlock(X)		
unlock(X)		
	wlock(X)	
	unlock(X)	
wlock(Y)		
unlock(Y)		
		rlock(Y)
		unlock(Y)
		wlock(Z)
		unlock(Z)
	rlock(Z)	
	unlock(Z)	

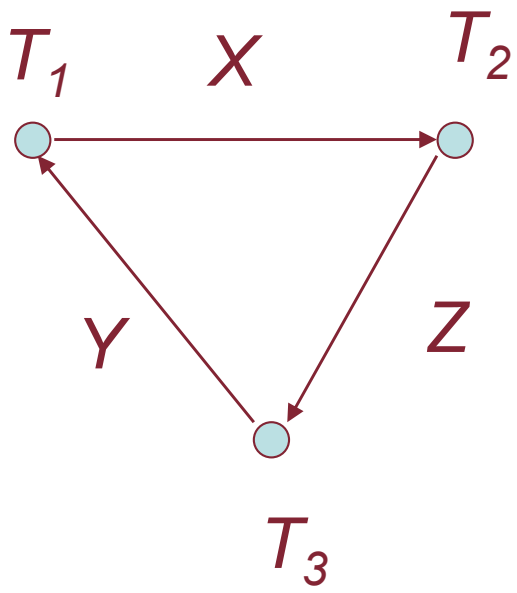
T_i --> T_j (con etichetta X) **se** in *S*
- T_i esegue una ***rlock(X)*** **o una** ***wlock(X)*** e T_j è la transazione **che esegue la successiva** ***wlock(X)***
- T_i esegue una ***wlock(X)*** e T_j esegue **una** ***rlock(X)*** **dopo** che T_i ha eseguito **la** ***wlock(X)*** e **prima** che **un'altra** transazione esegua **una** ***wlock(X)***.



È serializzabile (equivalente allo schedule seriale T_1 T_3 T_2)

T_1	T_2	T_3
rlock(X)		
unlock(X)		
	wlock(X)	
	unlock(X)	
		rlock(Y)
		unlock(Y)
	rlock(Z)	
	unlock(Z)	
		wlock(Z)
		unlock(Z)
wlock(Y)		
unlock(Y)		

il seguente schedule delle **stesse** transazioni



non è serializzabile

Risultati

- Una transazione nel modello a tre valori è a due fasi se **nessuna** operazione di **lock** (**rlock o wlock**) **segue** una operazione di **unlock**

- Se **ogni** transazione in un insieme T **è a due fasi** allora **ogni** schedule di T è **serializzabile**
- Solo se **tutte le transazioni sono a due fasi** possiamo avere la certezza che **ogni** schedule è serializzabile