Lezione 29– Deadlock e livelock - Protocollo di locking a due fasi stretto

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

Il deadlock (stallo)

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

Un *deadlock* si verifica quando

- **ogni transazione in un insieme** $\mathcal{T}$ è in attesa di ottenere un lock su un item sul quale qualche altra transazione nell'insieme $\mathcal{T}$ mantiene un lock e quindi
- **rimane bloccata** e quindi
- **non rilascia i lock** e quindi
- **può bloccare anche transazioni che non sono in** $\mathcal{T}$

Soluzioni per il deadlock (approcci risolutivi)

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

Quando si verifica una situazione di stallo, questa viene risolta

Per verificare il sussistere di una situazione di stallo si mantiene il ***grafo di attesa***

nodi: le transazioni

archi: c'è un arco $T1 \rightarrow T2$ se la transazione *T1* è in attesa di ottenere un lock su un item sul quale *T2* mantiene un lock

se in tale grafo c'è un ciclo si sta verificando una situazione di stallo che coinvolge le transazioni nel ciclo

Soluzioni per il deadlock (approcci risolutivi)

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

Per risolvere il sussistere di una situazione di stallo una transazione nel ciclo viene *rolled-back* e successivamente viene fatta ripartire

roll-back:

a) la transazione è abortita
b) i suoi effetti sulla base di dati vengono annullati ripristinando i valori dei dati precedenti l'inizio della sua esecuzione
c) tutti i lock mantenuti dalla transazione vengono rilasciati

Soluzioni per il deadlock (approcci preventivi)

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

Si cerca di evitare il verificarsi di situazioni di stallo adottando opportuni protocolli

Soluzioni per il deadlock (approcci preventivi)

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

Esempio

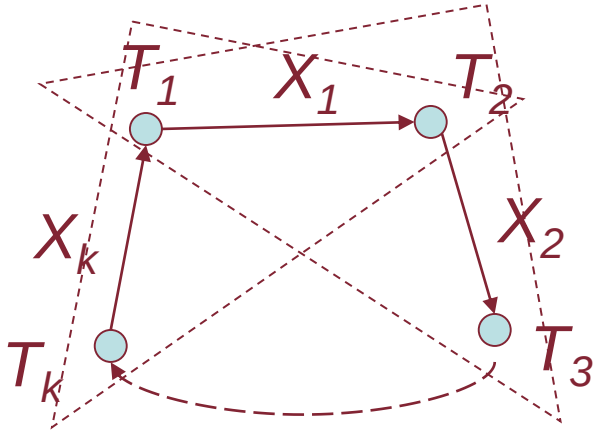
Si ordinano gli item e si impone alle transazioni di richiedere i lock necessari seguendo tale ordine

Soluzioni per il deadlock (approcci preventivi)

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

- In tal modo non ci possono essere cicli nel grafo di attesa (e quindi non si può verificare un deadlock).

Per assurdo: le transazioni richiedono gli item seguendo l'ordine fissato e nel grafo di attesa c'è un ciclo



$X_1 \prec X_2$ (X_1 precede X_2 nell'ordinamento)
$X_2 \prec X_3$
...
$X_{k-1} \prec X_k$
da cui segue $\boldsymbol{X_1 \prec X_k}$

$\boldsymbol{X_k \prec X_1}$
(contraddizione)

Il livelock (attesa indefinita)

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

Un *livelock* si verifica quando

una transazione aspetta indefinitamente che gli venga garantito un lock su un certo item

Soluzioni per il livelock

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

- Il problema dell'attesa indefinita, può essere risolto
  - con una strategia first came-first served
  - eseguendo le transazioni in base alle loro priorità e aumentando la priorità di una transazione all'aumentare del tempo in cui rimane in attesa

Abort di una transazione

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

1. La transazione esegue un operazione non corretta (divisione per 0, accesso non consentito)
2. Lo scheduler rileva un deadlock
3. Lo scheduler fa abortire la transazione per garantire la serializzabilità (timestamp)
4. Si verifica un malfunzionamento hardware o software

Punto di commit

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

punto di commit di una transazione è il punto in cui la transazione

ha ottenuto tutti i lock che gli sono necessari

ha effettuato tutti i calcoli nell'area di lavoro

e quindi non può più essere abortita a causa di 1-3

Dati sporchi

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

dati sporchi: dati scritti da una transazione sulla base di dati prima che abbia raggiunto il punto di commit

Roll-back a cascata

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

Quando una transazione T viene abortita devono essere annullati gli effetti sulla base di dati prodotti:

- da T
- da qualsiasi transazione che abbia letto dati sporchi

Riprendiamo in esame i problemi legati all'esecuzione concorrente delle transazioni visti inizialmente

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)	

L'**aggiornamento** di *X* prodotto da T_1 viene **perso**

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

Il valore di *X* letto da T_2 è un **dato sporco** (temporaneo) in quanto prodotto da una transazione fallita

T_1	T_3
	somma:=0
read(X)	
X:=X-N	
write(X)	
	read(X)
	somma:=somma+X
	read(Y)
	somma:=somma+Y
read(Y)	
Y:=Y+N	
write(Y)	

Il valore di *somma* è un dato **aggregato non corretto**

T_1	T_2
wlock(X)	
read(X)	
X:=X-N	
write(X)	
unlock(X)	
	wlock(X)
	read(X)
	X:=X+M
	commit
	write(X)
	unlock(X)
wlock(Y)	
read(Y)	
Y:=Y+N	
commit	
write(Y)	
unlock(Y)	

L'uso di lock consente di risolvere il problema dell'aggiornamento perso ma non quello della lettura di un dato sporco …

T_1	T_3
	somma:=0
wlock(X)	
read(X)	
X:=X-N	
write(X)	
unlock(X)	
	rlock(X)
	rlock(Y)
	read(X)
	somma:=somma+X
	read(Y)
	somma:=somma+Y
	commit
	unlock(X)
	unlock(Y)
wlock(Y)	
read(Y)	
Y:=Y+N	
commit	
write(Y)	
unlock(Y)	

… né quello dell'aggregato non corretto

T_1	T_3
	somma:=0
wlock(X)	
read(X)	
X:=X-N	
write(X)	
wlock(Y)	
unlock(X)	
	rlock(X)
	read(X)
	somma:=somma+X
read(Y)	
Y:=Y+N	
commit	
write(Y)	
unlock(Y)	
	rlock(Y)
	read(Y)
	somma:=somma+Y
	commit
	unlock(X)
	unlock(Y)

Il protocollo di locking a due fasi risolve il problema dell'aggregato non corretto …

T_1	T_2
wlock(X)	
read(X)	
X:=X-N	
write(X)	
wlock(Y)	
unlock(X)	
	wlock(X)
	read(X)
	X:=X+M
	commit
	write(X)
	unlock(X)
read(Y)	
Y:=Y+N	
commit	
write(Y)	
unlock(Y)	

… ma non quello della lettura di un dato sporco

Per risolvere il problema della lettura di dati sporchi occorre che le transazioni obbediscano a regole più restrittive del protocollo di locking a due fasi

Protocollo a due fasi stretto

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

Una transazione soddisfa il *protocollo di locking a due fasi stretto* se:

1. **non scrive** sulla base di dati **fino a quando** non ha raggiunto il suo **punto di commit**

   se una transazione **è abortita** allora **non ha modificato** nessun item nella base di dati
2. **non rilascia un lock finchè non ha finito di scrivere** sulla base di dati.

   se una transazione legge un item scritto da un'altra transazione quest'ultima non può essere abortita

T_1	T_3
	somma:=0
wlock(X)	
read(X)	
X:=X-N	
wlock(Y)	
read(Y)	
Y:=Y+N	
commit	
write(X)	
write(Y)	
unlock(X)	
unlock(Y)	
	rlock(X)
	read(X)
	somma:=somma+X
	rlock(Y)
	read(Y)
	somma:=somma+Y
	commit
	unlock(X)
	unlock(Y)

T_1	T_2
wlock(X)	
read(X)	
X:=X-N	
wlock(Y)	
read(Y)	
Y:=Y+N	
commit	
write(X)	
write(Y)	
unlock(X)	
unlock(Y)	
	wlock(X)
	read(X)
	X:=X+M
	commit
	write(X)
	unlock(X)

Classificazione dei protocolli

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

- conservativi

  cercano di **evitare** il verificarsi di situazioni di stallo

- aggressivi

  cercano di processare le transazioni **il più rapidamente** possibile anche se ciò **può portare a situazioni di stallo**

Protocolli conservativi

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

Versione più conservativa:

Una transazione T richiede tutti i lock che servono **all'inizio** e li ottiene se e solo se:

- **tutti i lock** sono disponibili

se non li può ottenere tutti viene messa in una coda di attesa

Si evita il deadlock, ma non il livelock.

Protocolli conservativi

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

Per evitare il verificarsi sia del dealdock che del livelock:

Una transazione T richiede tutti i lock che servono all'inizio e li ottiene se e solo se:

- tutti i lock sono disponibili
- nessuna transazione **che precede *T* nella coda** è in attesa di un lock richiesto da *T*

Protocolli conservativi

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

VANTAGGI:

- si evita il verificarsi sia del dealdock che del livelock

SVANTAGGI:

- l'esecuzione di una transazione può essere **ritardata**
- una transazione è costretta a richiedere un lock **su ogni item che potrebbe essergli necessario** anche se poi di fatto non l'utlizza

Protocolli aggressivi

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

Versione più aggressiva:

una transazione deve richiedere un lock su un item **immediatamente** prima di leggerlo o scriverlo

Può verificarsi un deadlock

Protocolli conservativi vs protocolli aggressivi

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

Se la probabilità che due transazioni richiedano un lock su uno stesso item è:

- alta

  è conveniente un protocollo conservativo in quanto evita al sistema il sovraccarico dovuto alla gestione dei deadlock (rilevare e risolvere situazioni di stallo, eseguire parzialmente transazioni che poi vengono abortite, rilascio dei lock mantenuti da transazioni abortite)

- bassa

  è conveniente un protocollo aggressivo in quanto evita al sistema il sovraccarico dovuto alla gestione dei lock (decidere se garantire un lock su un dato item ad una data transazione, gestire la tavola dei lock, mettere le transazioni in una coda o prelevarle da essa)