Esecuzione concorrente di transazioni

ALBERTO BELUSSI

PARTE I

ANNO ACCADEMICO 2010-2011

Osservazione

- Per gestire con prestazione accettabili il carico di lavoro tipico delle applicazioni gestionali (100 o 1000 tps) un DBMS deve eseguire le transazioni in modo concorrente.
- L'esecuzione concorrente di transazioni senza controllo può generare anomalie (come vedremo in seguito).
- È quindi necessario introdurre dei meccanismi di controllo nell'esecuzione delle transazioni per evitare tali anomalie.

Anomalie di esecuzione concorrente

Anomalie tipiche

- Lettura sporca
- Lettura inconsistente
- Perdita di update
- Aggiornamento fantasma

Notazione:

- Indichiamo con $\mathbf{t_i}$ una transazione
- $\mathbf{r_i(x)}$: è una operazione di lettura eseguita dalla transazione $\mathbf{t_i}$ sulla risorsa **x**;
- $\mathbf{w_i(x)}$: è una operazione di scrittura eseguita dalla transazione $\mathbf{t_i}$ sulla risorsa **x**;

Anomalie di esecuzione concorrente

Esempi alla lavagna

Concetti di base

Schedule

Sequenza di operazioni di lettura e scrittura eseguite su risorse della base di dati da diverse transazioni concorrenti

Esso rappresenta una possibile esecuzione concorrente di diverse transazioni.

Quali schedule accettare per evitare le anomalie?

Si stabilisce di accettare SOLO gli schedule “equivalenti” ad uno schedule seriale.

Concetti di base

Schedule seriale

E' uno schedule dove le operazioni di ogni transazione compaiono in sequenza senza essere inframezzate da operazioni di altre transazioni.

s_1: r1(x) r2(x) w2(x) w1(x) NON è SERIALE

s_2: r1(x) w1(x) r2(x) w2(x) è SERIALE

Concetti di base

Schedule serializzabile

E' uno schedule "equivalente" ad uno schedule seriale

A questo punto occorre definire il concetto di EQUIVALENZA tra schedule.

L'idea è che si vogliono considerare equivalenti due schedule che producono gli stessi effetti sulla base di dati.

Quindi accettare schedule serializzabili significa accettare schedule che hanno gli stessi effetti di uno schedule seriale.

Equivalenza tra schedule

Ipotesi di commit-proiezione

Si suppone che le transazioni abbiano esito noto. Quindi si tolgono dagli schedule tutte le operazioni delle transazioni che non vanno a buon fine.

Richiedono questa ipotesi le seguenti tecniche di gestione della concorrenza:

- Gestione della concorrenza basata sulla view-equivalenza
- Gestione delle concorrenza basata sulla conflict-equivalenza

View-equivalenza

Definizioni preliminari

RELAZIONE "LEGGE_DA"

Dato uno schedule S si dice che un'operazioni di lettura $\mathbf{r_i(x)}$, che compare in S, *LEGGE_DA* un'operazione di scrittura $\mathbf{w_j(x)}$, che compare in S, se $\mathbf{w_j(x)}$ precede $\mathbf{r_i(x)}$ in S e non vi è alcuna operazione $\mathbf{w_k(x)}$ tra le due.

Esempio

S1: $r_1(x)\ r_2(x)\ w_2(x)\ w_1(x)$ LEGGE_DA(S1) = ∅

S2: $r_1(x)\ w_1(x)\ r_2(x)\ w_2(x)$ LEGGE_DA(S2) = $\{(r_2(x), w_1(x))\}$

View-equivalenza

Definizioni preliminari

SCRITTURE FINALI

Dato uno schedule S si dice che un'operazione di scrittura **$w_i(x)$**, che compare in S, è una SCRITTURA FINALE se è l'ultima operazione di scrittura della risorsa x in S.

Esempio

S1: $r_1(x)\ r_2(x)\ w_2(x)\ w_1(x)$ SCRITTURE_FINALI(S1) = $\{w_1(x)\}$

S2: $r_1(x)\ w_1(x)\ r_2(x)\ w_2(x)$ SCRITTURE_FINALI(S2) = $\{w_2(x)\}$

View-equivalenza

View-equivalenza

Due schedule S_1 e S_2 sono view-equivalenti ($S_1 \approx_V S_2$) se possiedono le stesse relazioni LEGGE_DA e le stesse scritture finali.

View-serializzabilità

Uno schedule S è view-serializzabile (VSR) se esiste uno schedule seriale S' tale che S' $\approx_V$ S.

View-equivalenza

Osservazioni

- L'algoritmo per il test di view-equivalenza tra due schedule è di complessità linere.
- L'algoritmo per il test di view-serializzabilità di uno schedule è di complessità esponenziale.
- In conclusione, la VSR richiede algoritmi di complessità troppo elevata e l'ipotesi di commit-proiezione $\Rightarrow$ non è applicabile nei sistemi reali

Conflict-equivalenza

Definizioni preliminari

CONFLITTO

Dato uno schedule S si dice che una coppia di operazioni ($\mathbf{a_i}$, $\mathbf{a_j}$), dove $\mathbf{a_i}$ e $\mathbf{a_j}$ compaiono nello schedule S, rappresentano un conflitto se:

- $i \neq j$
- entrambe operano sulla stessa risorsa
- almeno una di esse è una operazione di scrittura
- $\mathbf{a_i}$ compare in S prima di $\mathbf{a_j}$

Conflict-equivalenza

Conflict-equivalenza

Due schedule S_1 e S_2 sono conflict-equivalenti ($S_1 \approx_C S_2$) se possiedono le stesse operazioni e gli stessi conflitti.

Conflict-serializzabilità

Uno schedule S è conflict-serializzabile (CSR) se esiste uno schedule seriale S' tale che S' $\approx_C$ S.

Conflict-equivalenza

Osservazioni

- L'algoritmo per il test di conflict-serializzabilità di uno schedule è di complessità lineare.
- Algoritmo (dato uno schedule S):
  - Si costruisce il grafo G(N,A) dove N={t_1, …, t_n} con t1, …, tn transazioni di S; (t_i, t_j) ∈ A se esiste almeno un conflitto (a_i, a_j) in S.
  - Se il grafo così costruito è ACICLICO allora S è conflict-serializzabile.

Conflict-equivalenza

Verifica algoritmo sul grafo

CSR ⇒ grafo ACICLICO

Se uno schedule S è CSR allora è $\approx_C$ ad uno schedule seriale. Supponiamo che le transazioni nello schedule seriale siano ordinate secondo il T_{ID}: t1, t2, …, tn . Poiché lo schedule seriale ha tutti i conflitti nello stesso ordine dello schedule S, nel grafo di S ci possono essere solo archi (i,j) con i<j e quindi il grafo non può avere cicli, perché un ciclo richiede almeno un arco (i,j) con i>j.

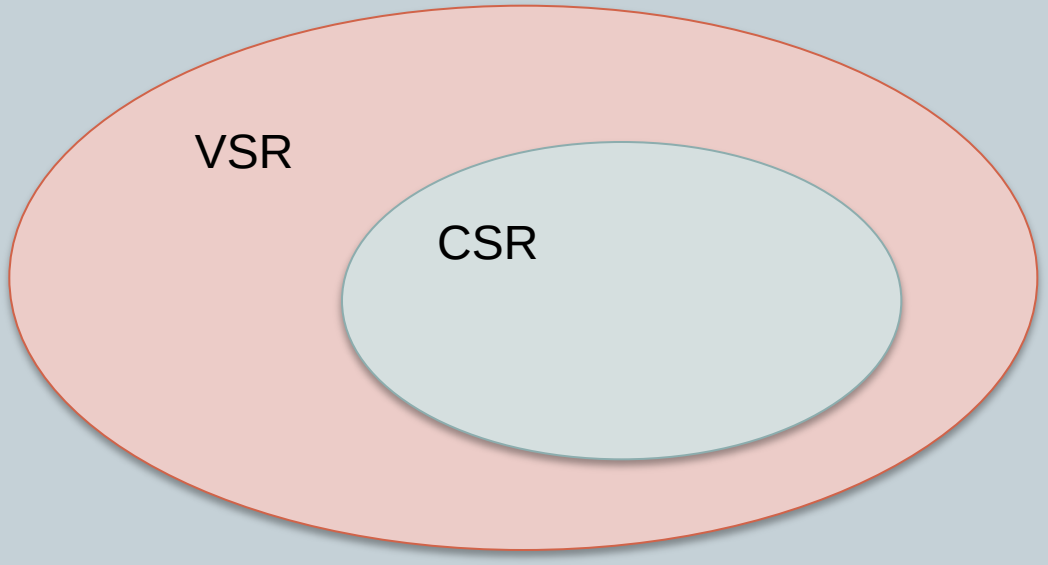
grafo ACICLICO ⇒ CSR

Se il grafo di S è aciclico, allora esiste fra i nodi un "ordinamento topologico" (cioè una numerazione dei nodi tale che il grafo contiene solo archi (i,j) con i<j). Lo schedule seriale le cui transazioni sono ordinate secondo l'ordinamento topologico è equivalente a S, perché per tutti i conflitti (i,j) si ha sempre i<j.

CSR vs VSR

Osservazioni

- La conflict-serializzabilità è condizione sufficiente ma non necessaria per la view-serializzabilità.



CSR ⊂ VSR

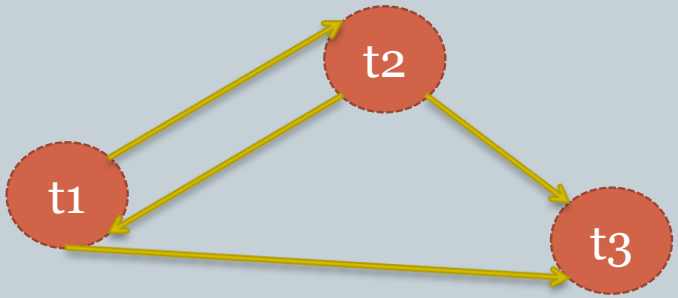
Controesempio per la non necessità:

- $r_1(x)\ w_2(x)\ w_1(x)\ w_3(x)$ LEGGE_DA=∅ SCR_FIN={$w_3(x)$}

  view-serializzabile: in quanto view-equivalente allo schedule seriale

  r1(x)w1(x)w2(x)w3(x) LEGGE_DA= ∅ SCR_FIN={$w_3(x)$}

  non conflict-serializzabile: in quanto il grafo dei conflitti è ciclico.



CSR $\Rightarrow$ VSR

Supponiamo S1 $\approx_C$ S2 e dimostriamo che S1 $\approx_V$ S2:

Osserviamo che poiché S1 $\approx_C$ S2 i due schedule hanno:

– stesse scritture finali: se così non fosse, ci sarebbero almeno due scritture sulla stessa risorsa in ordine diverso e poiché due scritture sono in conflitto i due schedule non sarebbero $\approx_C$

– stessa relazione "legge_da": se così non fosse, ci sarebbero scritture in ordine diverso o coppie lettura-scrittura in ordine diverso e quindi, come sopra sarebbe violata la $\approx_C$

Esercizi

ALLA LAVAGNA