:: Basi di Dati - Complementi - Esercizi sugli schedule ::

Torna alla pagina di Basi di Dati - Complementi

Esercizio 1 (fatto in classe)

S: w0(x) r2(x) r1(x) w2(x) w2(z)

Vediamo se è serializzabile.

Leggi-da:
- r2(x) **legge-da** w0(x)
- r1(x) **legge-da** w0(x)

Già da qui si deduce che t0 deve venire prima di t1 e t2.

Scritture finali:
- w2(x)
- w2(z)

Quindi, la t2 è quella che ha l'ultima parola in fatto di scrittura.

Se t0 deve essere la prima, e t2 possibilmente l'ultima, è logico aspettarsi che uno schedule seriale equivalente al nostro abbia, in ordine, **t0 t1 t2**. Proviamo:

S: w0(x) r1(x) r2(x) w2(x) w2(z)

Leggi-da:
- r1(x) legge da w0(x)
- r2(x) legge da w0(x)

Scritture finali:
- w2(x)
- w2(z)

Le proprietà coincidono con quelle del mio schedule iniziale, quindi quello è serializzabile.

Esercizio 2 (fatto in classe)

S: w0(x) r1(x) w1(x) r2(x) w1(z)

Leggi-da:
- r1(x) **legge-da** w0(x)
- r2(x) **legge-da** w1(x)

La prima **leggi-da** mi dice che t0 precede t1. La seconda invece che t1 precede t2.

Scritture finali:
- w1(x)
- w1(z)

Prendiamo un seriale basato sulle nostre ipotesi **S: w0(x) r1(x) w1(x) w1(z) r2(x)**

Leggi-da:
- r1(x) **legge-da** w0(x)
- r2(x) **legge-da** w1(x)

Scritture finali:
- w1(x)
- w1(z)

Coincidono, quindi il nostro schedule iniziale è serializzabile. Non siete felici?

Esercizio 3 (fatto in classe)

S: r1(x) r2(x) w2(x) w1(x)

Leggi-da:
- r1(x) **legge-da** niente
- r2(x) **legge-da** niente

Queste due relazioni ci dicono che niente precede t1 e t2.

Scritture finali:
- w1(x)

Ci accorgiamo che questo schedule non è serializzabile. Infatti, dalle **leggi-da** scopriamo che sia t1 che t2 devono stare per prime. Ma se t1 sta per prima, allora la scrittura finale w1(x) non può essere rispettata, e nemmeno la r2(x) che legge da niente. Se faccio il contrario, è errore ugualmente.

L'errore che si verifica, in questo caso, è una **perdita di aggiornamenti**. Riprendendo la definizione, vuol dire che le modifiche apportate da una t vengono cancellate da un'altra t, e si perdono.

Esercizio 4 (fatto in classe)

S: r1(x) r1(y) r2(z) r2(y) w2(y) w2(z) r1(z)

Leggi-da:
- r1(x) **legge-da** niente
- r1(y) **legge-da** niente
- r2(z) **legge-da** niente
- r2(y) **legge-da** niente
- r1(z) **legge-da** w2(z)

Scritture finali:
- w2(y)
- w2(z)

Qui il problema è che se metto t1 all'inizio, perderei *r1(z)* **legge-da** *w2(z)*. Se invece metto t2 all'inizio, perderei *r1(y)* **legge-da** *niente*. Quindi non è serializzabile.

Appello gennaio 2007

Esercizio 2

```
r1(x) r2(y) w1(x) w3(z) r4(t) w2(t) r1(y) r4(z) w4(x) r3(t) w2(z)
```

È VSR?

Leggi-da:
```
r1(x) <- niente
r2(y) <- niente
r4(t) <- niente
r1(y) <- niente
r4(z) <- w3(z)
r3(t) <- w2(t)
```

Scritture

	finali	altre
x	4	1
y	-	-
z	2	3
t	2	-

Dalle **leggi-da** che vengon da qualcosa, derivo
```
t3 -> t4
t2 -> t3
```

Dalle **scritture finali** derivo
```
t1 -> t4
t3 -> t2
```

C'è un contrasto tra t2 -> t3 (dato dalle **leggi-da**) e tra t3 -> t2 (dato dalle **scritture finali**). Quindi, non è VSR. E quindi nemmeno CSR o 2PL o quello che è, che sono tutti sottinsiemi.

Altro esercizio

```
w3(x) r2(y) w2(x) w3(y) r3(z) r2(z) w1(x) r1(y) w3(z) r1(x)
```

È VSR?

Leggi-da:

```
r2(y) <- niente
r3(z) <- niente
r2(z) <- niente
r1(y) <- w3(y)   => 3 -> 1
r1(x) <- w1(x)
```

Scritture

	finali	**altre**
x	1	2, 3
y	3	-
z	3	-

Dalle **scritture** derivo che *t2 -> t1*, *t3 -> t1* Metto insieme, e ottengo *t2 -> t3 -> t1*: è VSR.

CSR

Stesso schedule di prima

```
w3(x) r2(y) w2(x) w3(y) r3(z) r2(z) w1(x) r1(y) w3(z) r1(x)
```

Per vedere se è Conflict-serializzabile, devo vedere i conflitti e tracciare un grafo delle dipendenze. Per vedere i conflitti, punto alle scritture, e devo guardare sia avanti che indietro alle letture sulla stessa variabile.

```
w3(x): in avanti: w2(x) => 3 -> 2
                  w1(x) => 3 -> 1
                  r1(x) => 3 -> 1

w2(x): indietro:  w3(x) => 3 -> 2
       in avanti: w1(x) => 2 -> 1
                  r1(x) => 2 -> 1

w3(y): indietro:  r2(y) => 2 -> 3
       in avanti: r1(y) => 3 -> 1

w1(x): indietro:  w3(x) => 3 -> 1
                  w2(x) => 2 -> 1
       in avanti: r1(x)
```

```
w3(z): indietro:  r2(x) => 2 -> 3
```

Trovo subito che ho in generale *3* -> *2*, ma in un caso *2* -> *3*. Creerei un ciclo nel grafico (disegnatelo, seguendo le frecce:)) e quindi non è CSR. Occhio: i CSR sono inclusi in VSR, ma sono un sottinsieme, ecco perché questo schedule è VSR ma non CSR.

Febbraio 2008

```
r1(x) r2(y) r3(z) w1(y) r4(x) w3(z) r3(x)  w2(y) w1(z) r4(z) r3(y) w4(x)
```

Leggi-da:

```
r1(x) <- niente
r2(x) <- niente
r3(z) <- niente
r4(x) <- niente
r3(x) <- niente
r4(z) <- w1(z)
r3(y) <- w2(y)
```

Scritture

	finali	altre
x	4	-
y	2	1
z	1	3

Le relazioni che ricavo sono che *t1* -> *t4*, *t2* -> *t3*, *t1* -> *t2*, *t3* -> *t1*. Si vede che c'è un ciclo: t1 -> t2 -> t3 -> t4. Non è VSR.

Gennaio 2008

Ocio, questo è bastardo.

```
r1(x) r2(y) w1(z) r2(x) w3(y) r3(z) w1(t) r3(y) w2(z) r2(t) r3(x)
```

Leggi-da:

```
r1(x) <- niente
r2(y) <- niente
r2(x) <- niente
r3(z) <- w1(z) => t1 -> t3
r3(y) <- w3(y)
r2(t) <- w1(t) => t1 -> t2
r3(x) <- niente
```

Scritture

	finali	altre
x	-	-

y	3	-
z	2	1
t	1	

Le mie relazioni sono: *t1* -> *t2*, *t2* -> *t3*, *t1* -> *t3* => l'ordinamento topologico che le rispetta sarebbe *t1* -> *t2* -> *t3*.

Bene, peccato che non sia così. Infatti, dopo aver fatto uno di questi robi, occorre scrivere il mio bello schedule seriale e verificarlo, ovvero controllare che le **leggi-da** e le **scritture finali** del seriale siano equivalenti a quelle del mio schedule, per dire che siano VSR.

Non sto a scriverle tutte, ma possiamo verificare che
`r3(z) <- w2(z)`
che NON corrisponde a prima, dove avevo *r3(z) <- w1(z)*. Come mai tutto ciò? Perché il grafico ha "mentito"?

Il motivo è che se considero attentamente lo schedule e le relazioni **leggi-da**, scopro che la relazione *r3(z) <- w1(z)* implica che NON ci sia nessun'altra scrittura di z tra *t1* e *t3*. Se poi controllo la relazione *t2*, vedo che essa in realtà *contiene* una scrittura di z! Ecco perché sono stato imbrogliato! Quindi, ricordiamoci di trovare sempre le **leggi-da** e le **scritture** anche dal seriale, che magari ci sono queste cose misteriose nascoste in agguato:)

2PL base

`r1(x) r2(x) w2(y) w2(x) w1(z) r3(x) r2(y) w3(x) w3(z) r3(x) w2(z)`

È possibile che sto coso sia stato generato da un 2PL base?

Ricordo che la regola del 2PL base è che se rilascio un lock, non posso più acquisirne di nuovi, su nessuna variabile (non solo quella rilasciata).

Per scoprirlo, uso uno schema di mia invenzione che ancora non è stato approvato dalla professoressa, ma penso di sottoporglielo al più presto. Le righe sono le **transazioni**, e ogni colonna rappresenta la variabile su cui quella trans sta lavorando.

```
   0 1 2 3 4 5 6 7 8 9 10
1: x       z
2:   x y x     y       z
3:           x   x z x
```

E che me ne faccio? Dovrei disegnare delle freccette, ma non lo faccio per motivi ASCII.

Considero la transazione 1: può benissimo prendere i lock su *x* e *z* all'inizio, quindi è OK.

Considero la transazione 2: lavora su tutte e tre le variabili, quindi deve acquisire il lock su *z* PRIMA di cedere *x* alla transazione 3, all'istante 5. Infatti, una volta ceduto un lock, non può riprenderne nessun altro.