

# Gestione dei dati

## Parte 2

# Esercitazione sulla gestione della concorrenza

Maurizio Lenzerini, Riccardo Rosati

Facoltà di Ingegneria  
Sapienza Università di Roma  
Anno Accademico 2011/2012

<http://www.dis.uniroma1.it/~rosati/gd/>



SAPIENZA  
UNIVERSITÀ DI ROMA

# Esercizio 1

Per ognuno dei seguenti schedule:

S1 :  $w_0(x), r_2(x), r_1(x), w_2(x), w_2(z)$

S2 :  $w_0(x), r_1(x), r_2(x), w_2(x), w_2(z)$

S3 :  $r_1(x), r_2(x), w_2(x), w_1(x)$

S4 :  $r_1(x), r_2(x), w_2(x), r_1(x)$

dire se è view-serializzabile o no. Per quelli per i quali si risponde che sono view-serializzabili, motivare la risposta positiva. Per quelli per i quali si risponde che non sono view-serializzabili, illustrare quale anomalia presentano.

# Soluzione esercizio 1 (1)

- S2 è uno schedule seriale. In quanto tale è banalmente view-serializzabile (banalmente view-equivalente a se stesso).
- S1 è view-serializzabile in quanto è view-equivalente ad S2 (che è uno schedule seriale). In effetti S1 ed S2 hanno:
  - le stesse scritture finali:  $w_2(x)$  e  $w_2(z)$ ;
  - la stessa relazione legge-da:  $\{(r_2(x), w_0(x)), (r_1(x), w_0(x))\}$ .

## Soluzione esercizio 1 (2)

S3 non è view-serializzabile in quanto, per esserlo, dovrebbe esistere una schedule view-equivalente a S3 tale da avere come scrittura finale  $w1(x)$  e come relazione legge-da l'insieme vuoto (poiché nessuna lettura su  $x$  è preceduta da una scrittura su  $x$ ).

Supponiamo per assurdo che una schedule  $S$  siffatta esista. Perché  $w1(x)$  sia la scrittura finale di  $S$ ,  $T2$  deve essere la prima transazione a comparire in  $S$ .  $S$  non può quindi che essere il seguente schedule:

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

Ma allora la relazione legge-da di  $S$  è costituita dalla coppia  $(r1(x), w2(x))$ . Il che porta ad un assurdo.

# Soluzione esercizio 1 (3)

Si ha in effetti che S3 presenta un'anomalia da **perdita di aggiornamento**, in quanto T1 e T2 leggono lo stesso valore iniziale per x, e, dopo che T2 scrive il nuovo valore, tale valore è soprascritto da T1, in modo tale che l'aggiornamento per opera di T2 sia "perso".

## Soluzione esercizio 1 (4)

S4 non è view-serializzabile, in quanto, per esserlo, uno fra i seguenti schedule seriali dovrebbe essere view-equivalente ad S4, dove indichiamo con  $r'1(x)$  la seconda lettura di  $x$  per opera di T1:

S':  $r2(x), w2(x), r1(x), r'1(x)$

→ LEGGE-DA =  $\{(r1(x), w2(x)), (r'1(x), w2(x))\}$

S'':  $r1(x), r'1(x), r2(x), w2(x)$

→ LEGGE-DA =  $\{\}$

Poiché la relazione legge-da di S4 è tale che:

$$\text{LEGGE-DA} = \{(r'1(x), w2(x))\}$$

→ non esiste nessuna schedule seriale view-equivalente ad S4!

S4 presenta un'anomalia da **lettura non ripetibile**, in quanto T1 legge due valori diversi per  $x$ , pur non avendo effettuato nessuna modifica su di esso.

## Esercizio 2

Si analizzi la seguente sequenza di azioni di un insieme di transazioni, e si dica se essa costituisca uno schedule view-serializzabile oppure no, se costituisca uno schedule conflict-serializzabile oppure no, e se essa possa o no dare luogo ad uno schedule che segue il protocollo del 2PL. Motivare tutte e tre le risposte.

S:  $r1(x)$ ,  $w2(x)$ ,  $r3(x)$ ,  $r1(y)$ ,  $r4(z)$ ,  $w2(y)$ ,  $r1(v)$ ,  $w3(v)$ ,  $r4(v)$ ,  $w4(y)$ ,  $w5(y)$ ,  $w5(z)$

# Soluzione esercizio 2 (1)

Ricordiamo che:

- se uno schedule è eseguibile nel 2PL, allora tale schedule è conflict-serializzabile
- se uno schedule è conflict-serializzabile, allora tale schedule è view-serializzabile
- se uno schedule non è view-serializzabile, allora non è conflict-serializzabile
- se uno schedule non è conflict-serializzabile, allora non è eseguibile nel 2PL

Iniziamo a risolvere l'esercizio provando a verificare se S1 è eseguibile nel 2PL

## Soluzione esercizio 2 (2)

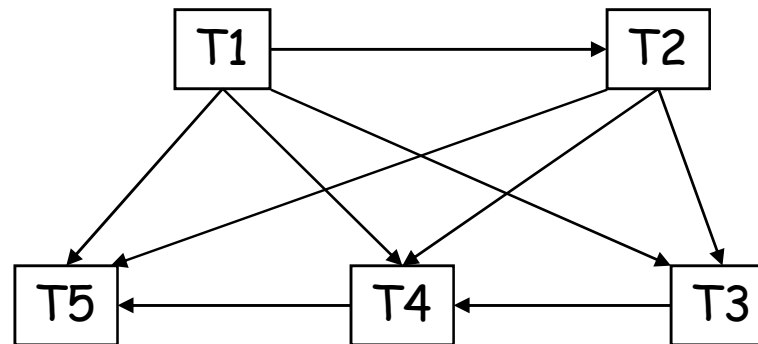
Analizzando S ci accorgiamo che non esiste alcun modo per le transazioni di fare le operazioni di lock e unlock delle risorse in modo tale da rispettare il protocollo del 2PL. Infatti:

- perché T3 possa procedere con la lettura di x, è necessario che la transazione T2 abbia rilasciato il lock su x (poiché T2 ha ottenuto il lock esclusivo in scrittura su x)
  - T2 entra pertanto in fase calante (di rilascio di lock) quando T3 legge x
- perché T1 possa procedere con la lettura di y è necessario che nessuna transazione detenga in quel momento il lock esclusivo in lettura su y
  - T2 non può detenerlo
- perché T2 possa procedere con la scrittura di y, è necessario che detenga il lock esclusivo su y
  - T2 dovrebbe richiedere il lock su y dopo che T1 ha letto y

**IMPOSSIBILE:** T2 è già in fase calante!

## Soluzione esercizio 2 (3)

Proviamo ora a costruire il grafo delle precedenze  $G(S)$ .



Tale grafo è aciclico.  $S$  è pertanto conflict-serializzabile (cf. teorema secondo il quale uno schedule è conflict-serializzabile se e solo se il suo grafo delle precedenze è aciclico)

→  $S$  è anche view-serializzabile!

In particolare, uno schedule  $S'$  seriale conflict-equivalente (e quindi view-equivalente) ad  $S$  (corrispondente ad un ordine topologico del grafo delle precedenze) è il seguente:

$r1(x), r1(y), r1(v), w2(x), w2(y), r3(x), w3(v), r4(z), r4(v), w4(y), w5(y), w5(z)$

## Esercizio 3

Siano date le transazioni T1 e T2 che seguono:

T1: r1(y), r1(z), w1(x) c1

T2: r2(z), r2(x), w2(y), w2(x) c2

1. Dire se esiste uno schedule S su T1 e T2 tale che:
  1. S non è seriale;
  2. S segue il protocollo dello strict 2PL;
  3. la lettura di x da parte di T2 precede la lettura di y da parte di T1.
2. Dire se esiste uno schedule S' su T1 e T2 tale che:
  1. S non è seriale;
  2. S segue il protocollo dello strict 2PL;
  3. la lettura di x da parte di T2 segue la lettura di y da parte di T1.

Motivare le risposte.

# Soluzione esercizio 3 (1)

- (1) Non esiste alcuna schedule tale da verificare le prime tre condizioni. Infatti:
- affinché lo schedule non sia seriale, è necessario inframmezzare la sequenza delle azioni di una transazione con una o più azioni dell'altra.
  - affinché lo schedule segua il protocollo dello strict 2PL, ogni transazione che compare in esso deve rilasciare i lock solo dopo aver effettuato correttamente l'operazione di abort/commit.
  - perché sia soddisfatta la condizione 3),  $r2(z)$ ,  $r2(x)$  devono essere le prime azioni dello schedule.

## Soluzione esercizio 3 (2)

Poiché la successiva azione di T2 è  $w2(y)$  e la prima azione di T1 è  $r1(y)$ , si hanno due casi:

- $w2(y)$  viene prima di  $r1(y)$ : in questo caso, T2 deve fare unlock su  $y$  prima di  $r1(y)$ , e quindi anche  $c2$  prima di  $r1(y)$  (al fine di assicurare che lo schedule sia stretto), e questo significa che S è seriale
- $w2(y)$  viene dopo  $r1(y)$ : in questo caso T1 deve fare unlock su  $y$  prima di  $w2(y)$ , e quindi anche  $c1$  prima di  $w2(y)$  (al fine di assicurare che lo schedule sia stretto), e questo significa che T1 esegue  $w1(x)$  e quindi ottiene l'uso esclusivo di  $x$  prima che T2 possa fare  $c2$ , cosa che esclude che lo schedule sia stretto.

## Soluzione esercizio 3 (2)

(2) Se sostituiamo la condizione 3) con la condizione 3bis), seguendo un ragionamento analogo a quello fatto precedentemente, si ha che uno schedule non seriale che soddisfa le 3 condizioni richieste è il seguente :

S: r2(z), r1(y), r1(z), w1(x), c1, r2(x), w2(y), w2(x), c2

Una possibile assegnazione delle operazioni di lock e unlock è la seguente :

sl2(z), r2(z), sl1(y), r1(y), sl1(z), r1(z), xl1(x), w1(x), c1, ul1(y), ul1(z), ul1(x), ul2(x), r2(x), xl2(y), w2(y), w2(x), c2, ul2(z), ul2(x), ul2(y)

## Esercizio 4

Si considerino due risorse A e B sulle quali opera un controllo di concorrenza basato su timestamp. Si assuma che i timestamp in lettura e scrittura di ogni risorsa siano pari a 0, e che  $cb(A)=cb(B)=true$ . Si assuma inoltre che il clock del sistema sia anche esso pari a 0 e che il sistema assegni ad ogni transazione il valore del clock al momento della prima azione della transazione (una transazione inizia quando esegue la sua prima operazione). Indicare le azioni dello scheduler a fronte del seguente input:

$r1(A), r2(B), r3(A), r2(A), w1(A), w3(A)$

e dire se lo schedule accettato è o no uno schedule 2PL stretto (con lock esclusivi e condivisi).

# Soluzione esercizio 4 (1)

- Prima che lo scheduler riceva l'input, si ha:  
 $wts(A) = wts(B) = rts(A) = rts(B) = 0$  e  $cb(A)=cb(B)=true$
- A fronte dell'input dato, il sistema risponde come segue (richieste  $\rightarrow$  risposte  $\rightarrow$  aggiornamenti):
  - $r1(A) \rightarrow ok \rightarrow ts(T1)=1, rts(A)=1$ , perché  $ts(T1) \geq wts(A)$  e ché  $cb(A)=true$
  - $r2(B) \rightarrow ok \rightarrow ts(T2)=2, rts(B)=2$ , perché  $ts(T2) \geq wts(B)$  e ché  $cb(B)=true$
  - $r3(A) \rightarrow ok \rightarrow ts(T3)=3, rts(A)=3$ , perché  $ts(T3) \geq wts(A)$  e ché  $cb(A)=true$
  - $r2(A) \rightarrow ok \rightarrow rts(A)=3$ , perché  $ts(T2) \geq wts(A)$ , ché  $cb(A)=true$  e ché  $\max(ts(T2), rts(A))=3$
  - $w1(A) \rightarrow no$ : T1 viene uccisa, perché  $ts(T1) < rts(A)=3$
  - $w3(A) \rightarrow ok \rightarrow wts(A)=3$  e  $cb(A)=false$ , perché  $ts(T3)=rts(A)$  e  $ts(T3) \geq wts(A)$

## Soluzione esercizio 4 (2)

Poiché T1 viene uccisa, lo schedule accettato (ignorando le azioni delle transazioni abortite) è il seguente:

$r_2(B), r_3(A), r_2(A), w_3(A)$

Ora, è questo uno schedule 2PL stretto? Per poter rispondere a questa domanda bisogna ovviamente fissare la posizione delle commit.

Assumiamo che ogni commit segua immediatamente l'ultima istruzione della rispettiva transazione: in tal caso, possiamo concludere che il precedente è uno schedule 2PL stretto, il che è dimostrato da:

$sl_2(A), sl_2(B), r_2(B), sl_3(A), r_3(A), r_2(A), c_2, u_2(A), xl_3(A), w_3(A), c_3, u_3(A), u_3(B)$

## Esercizio 5

Si considerino due risorse A e B sulle quali opera un controllo di concorrenza basato su timestamp. Si assuma che i timestamp in lettura e scrittura di ogni risorsa siano pari a 0. Si assuma inoltre che il clock del sistema sia anche esso pari a 0 e che il sistema assegni ad ogni transazione il valore del clock al momento della prima azione della transazione (una transazione inizia quando esegue la sua prima operazione). Indicare le azioni dello scheduler a fronte del seguente input:

$r1(B), w1(A), w2(B), w1(B), r2(A)$

# Soluzione esercizio 5

Prima che lo scheduler riceva l'input, si ha:

$$wts(A) = wts(B) = rts(A) = rts(B) = 0$$

A fronte dell'input dato, il sistema risponde come segue (richieste → risposte → aggiornamenti):

- r1(B) → ok →  $ts(T1)=1$ ,  $rts(B)=1$ , perché  $ts(T1) \geq wts(B)$  e che  $rts(B) = \max(ts(T1), rts(B))$
- w1(A) → ok →  $wts(A)=1$  e  $cb(A)=false$ , perché  $ts(T1) \geq wts(A)$  e che  $ts(T1) \geq rts(A)$
- w2(B) → ok →  $ts(T2)=3$ ,  $wts(B)=3$  e  $cb(B)=false$ , perché  $ts(T2) \geq wts(B)$  e che  $ts(T2) \geq rts(B)$
- w1(B) → T1 messa in attesa di  $cb(B)=true$  o del rollback di T2, perché  $ts(T1) = rts(B)$  e  $ts(T1) < wts(B)$
- r2(A) → T2 messa in attesa di  $cb(a)=true$  o del rollback di T1

**DEADLOCK!**

## Esercizio 6

Si considerino due risorse  $x$  e  $y$  sulle quali opera un controllo di concorrenza basato su **timestamp multi-versione**. Si assuma che i timestamp in lettura e scrittura di ogni risorsa siano pari a 0. Si assuma inoltre che il clock del sistema sia anche esso pari a 0 e che il sistema assegni ad ogni transazione il valore del clock al momento della prima azione della transazione (una transazione inizia quando esegue la sua prima operazione). Indicare le azioni dello scheduler a fronte del seguente input:

$r1(x), r2(y), w2(x), w1(y)$

# Soluzione esercizio 6 (1)

- Prima che lo scheduler riceva l'input, esiste una sola copia di ogni risorsa che è la copia dell'oggetto corrispondente nella base di dati (ricordiamo che in  $[x,y]$   $x$  è wts e  $y$  è rts):

$$\text{intervalli}(x) = \{\text{intervallo}(x_0) = [0,0]\}$$

$$\text{Intervalli}(y) = \{\text{intervallo}(y_0) = [0,0]\}$$

- A fronte dell'input dato, il sistema risponde come segue (richieste  $\rightarrow$  risposte  $\rightarrow$  aggiornamenti):

$r1(x) \rightarrow \text{ok} \rightarrow \text{ts}(T1)=1$ ;  $x_0$  è l'unica versione con  $\text{wts}^{x_0} \leq \text{ts}(T1)$ , quindi T1 legge  $x_0$  e poiché  $\text{ts}(T1) > \text{rts}^{x_0}$  lo scheduler pone  $\text{rts}^{x_0} = \text{ts}(T1) = 1$ , quindi si ha

$$\text{Intervalli}(x) = \{\text{intervallo}(x_0) = [0,1]\}$$

$r2(y) \rightarrow \text{ok} \rightarrow \text{ts}(T2)=2$ ;  $y_0$  è l'unica versione con  $\text{wts}^{y_0} \leq \text{ts}(T2)$ , quindi T2 legge  $y_0$  e poiché  $\text{ts}(T2) > \text{rts}^{y_0}$  lo scheduler pone  $\text{rts}^{y_0} = \text{ts}(T2) = 2$ , quindi si ha:

$$\text{Intervalli}(y) = \{\text{intervallo}(y_0) = [0,2]\}$$

## Soluzione esercizio 6 (2)

$w2(x) \rightarrow ok \rightarrow x0$  è l'unica versione con  $wts^{x0} \leq ts(T2)$  e poiché  $ts(T2) \geq rts^{x0}$ , lo scheduler accetta la scrittura e crea la versione  $x2$  con:

$Intervalli(x) = \{intervallo(x0) = [0, 1], intervallo(x2) = [2, 2]\}$

$w1(y) \rightarrow no \rightarrow y0$  è l'unica versione con  $wts^{y0} \leq ts(T1)$ , ma dato che  $ts(T1) < rts^{y0}$  lo scheduler rifiuta la scrittura ed esegue il rollback di  $T1$

# Esercizio proposto

Dato lo schedule

$S = r1(A), r2(B), r3(B), w4(A), r2(A), w5(A), r4(A), r3(A), r5(A)$ ,  
mostrare le differenze tra l'esecuzione di  $S$  nel metodo dei timestamp universione e l'esecuzione di  $S$  nel metodo dei timestamp multiversione.