Esercizio 1

a. Si considerino i due seguenti schedule e si verifichi se siano view serializzabili.

1. r1(x), w1(x), w3(x), r2(y), r3(y), w3(y), w1(y), r2(x)

2. r1(y), r1(y), w2(z), w1(z), w3(z), w3(x), w1(x)

Soluzione

a.1. r1(x), w1(x), w3(x), r2(y), r3(y), w3(y), w1(y), r2(x)
Questo schedule è NonVSR. In uno schedule seriale view-equivalente a questo schedule la transazione 1 dovrebbe seguire la transazione 3 a causa delle SCRITTURE FINALI su y, ma dovrebbe anche precedere la transazione 3 a causa della relazione LEGGE – DA su x.

a.2. r1(y), w2(z), w1(z), w3(z), w3(x), w1(x)
La transazione 1 ha due scritture, una su z ed un'altra su x, ma anche la transazione 3 ha due scritture, una su z ed un'altra su x. Nello schedule di partenza, le scritture finali su x e z sono originate rispettivamente dalle transazioni 1 e 3. Nessuno schedule seriale potrà esibire le stesse scritture finali. Questo schedule non è quindi VSR.

b. Si considerino i seguenti 2 schedule e si verifichi se siano conflict-serializzabili.

1. r1(x), r1(t), r3(z), r4(z), w2(z), r4(x), r3(x), w4(x), w4(y), w3(y), w1(y), w2(t)
2. r1(x), r4(x), w4(x), r1(y), r4(z), w4(z), w3(y), w3(z), w1(t), w2(z), w2(t)

b.1. r1(x), r1(t), r3(z), r4(z), w2(z), r4(x), r3(x), w4(x), w4(y), w3(y), w1(y), w2(t)
Per classificare questo schedule si deve realizzare un grafo dei conflitti. Consideriamo le operazioni relative a ogni risorsa separatamente:

t: r1, w2
x: r1, r4, r3, w4
y: w4, w3, w1
z: r3, r4, w2

Questo schedule non è CSR perchè il suo grafo dei conflitti è ciclico. Questo schedule è anche NonVSR; la transazione 1 ha la scrittura finale su y, perciò dovrebbe seguire la transazione 4. La transazione 4 scrive su x, e dovrebbe seguire la transazione 1 che legge lo stato di x prima della scrittura della transazione 4.

b.2. r1(x), r4(x), w4(x), r1(y), r4(z), w4(z), w3(y), w3(z), w1(t), w2(z), w2(t)
Visto che il grafo dei conflitti è aciclico lo schedule è CSR, quindi anche VSR.
Lo schedule seriale equivalente è:

r1(x), r1(y), w1(t), r4(x), w4(x), r4(z), w4(z), w3(y), w3(z), w2(z), w2(t)

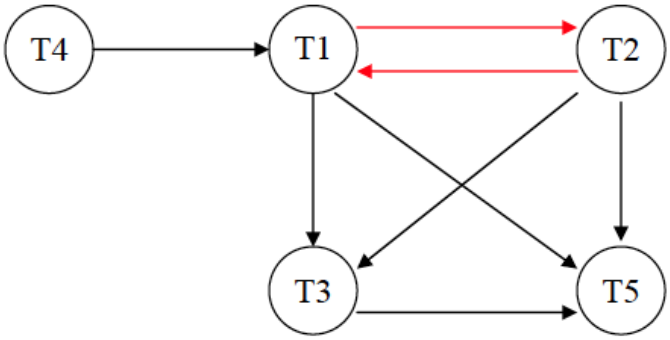
Esercizio 2

Classificare i seguenti schedule (come: NonSR, VSR, CSR); nel caso uno schedule sia VSR oppure CSR, indicare tutti gli schedule seriali e esso equivalenti.
1. r1(x), w1(x), r2(z), r1(y), w1(y), r2(x), w2(x), w2(z)
2. r1(x), w1(x), w3(x), r2(y), r3(y), w3(y), w1(y), r2(x)
3. r1(x), r2(x), w2(x), r2(x), r4(z), w1(x), w3(y), w3(x), w1(y), w5(x), w1(z), w5(y), r5(z)
4. r1(x), r3(y), w1(y), w4(x), w1(t), w5(x), r2(z), r3(z), w2(z), w5(z), r4(t), r5(t)
5. r1(x), r2(x), w2(x), r3(x), r4(z), w1(x), r3(y), r3(x), w1(y), w5(x), w1(z), r5(y), r5(z)

Soluzione:
1) r1(x), w1(x), r2(z), r1(y), w1(y), r2(x), w2(x), w2(z)
In questo schedule non ci sono conflitti, quindi è sia VSR che CSR, ed è conflict-equivalente a:
S: r1(x), w1(x), r1(y), w1(y), r2(z), r2(x), w2(x), w2(z)

2) r1(x), w1(x), w3(x), r2(y), r3(y), w3(y), w1(y), r2(x)
Questo schedule è NonSR. In uno schedule seriale view-equivalente a questo schedule la transazione 1 dovrebbe seguire la transazione 3 a causa delle SCRITTURE FINALI su Y, ma dovrebbe anche precedere la transazione 3 a causa della relazione LEGGE – DA su X.

3) r1(x), r2(x), w2(x), r3(x), r4(z), w1(x), w3(y), w3(x), w1(y), w5(x), w1(z), w5(y), r5(z)
Per classificare questo schedule si deve realizzare un grafo dei conflitti.
Consideriamo le operazioni relative a ogni risorsa separatamente:

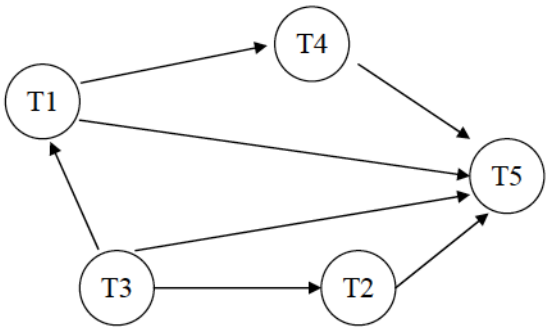


X: r1, r2, w2, r3, w1, w3, w5
Y: w3, w1, w5
Z: r4, w1, r5

Questo schedule non è CSR perchè il suo grafo dei confliti è cliclico.
Questo schedule è anche NonSR, perché, considerando la risorsa X, le transazioni 1 e 2 leggono X prima di qualsiasi altra operazione, ma entrambe scrivono anche su X.

4) r1(x), r3(y), w1(y), w4(x), w1(t), w5(x), r2(z), r3(z), w2(z), w5(z), r4(t), r5(t)
Per classificare questo schedule si deve realizzare un grafo dei conflitti.
Consideriamo le operazioni relative a ogni risorsa separatamente:
T: w1, r4, r5

X: r1, w4, w5
Y: r3, w1
Z: r2, r3, w2, w5



Questo schedule è sia CSR che VSR.
Gli schedule seriali equivalenti sono:
S1: r3(y), r3(z), r1(x), w1(y), w1(t), r2(z), w2(z), w4(x), r4(t), w5(x), w5(z), r5(t)
S2: r3(y), r3(z), r2(z), w2(z), r1(x), w1(y), w1(t), w4(x), r4(t), w5(x), w5(z), r5(t)

5) r1(x), r2(x), w2(x), r3(x), r4(z), w1(x), r3(y), r3(x), w1(y), w5(x), w1(z), r5(y), r5(z)
Lo schedule è NonSR ; le transazioni 1 e 2 leggono e scrivono X. Leggono X prima di ogni altra operazione, e così nessuna sequenza di schedule con queste transazioni potrà verificare la relazione LEGGE – DA

Esercizio 3

1) Se i seguenti schedule si presentassero a uno scheduler che usa il locking a due fasi, quali transazioni verrebbero messe in attesa? Una volta posta in attesa una transazione, le sue successive azioni non vanno più considerate. Considerare che venga sempre preso un lock esclusivo oppure prima un lock condiviso e poi uno esclusivo.

Soluzione:

1) r1(x), w1(x), r2(z), r1(y), w1(y), r2(x), w2(x), w2(z)

Nessuna transazione in attesa

2) r1(x), w1(x), w3(x), r2(y), r3(y), w3(y), w1(y), r2(x)

Le transazioni 3 e 1 e poi 2 vengono messe in attesa, producendo una situazione di deadlock; infatti, l'azione r2(x) deve aspettare l'oggetto x, messo in lock dalla transazione 1, e la transazione 1 è in attesa su w1(y) dell'oggetto y, messo in lock dalla transazione 2.

3) r1(x), r2(x), w2(x), r3(x), r4(z), w1(x), w3(y), w3(x), w1(y), w5(x), w1(z), w5(y), r5(z)
Le transazioni 2, 1, 3 e 5 sono messe in attesa. Si crea una situazione di deadlock dovuta alla richiesta di lock in scrittura su x da parte delle transazioni 1 e 2, entrambe con un lock sulla stessa risorsa x.

Esercizio 4

Se gli schedule seguenti si presentassero a uno scheduler basato su timestamp, quali transazioni verrebbero abortite?
Soluzione:

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

Operazione	Risposta	Nuovo Valore
read(x,1)	Ok	RTM(x)=1
write(x,1)	Ok	WTM(x)=1
read(z,2)	Ok	RTM(z)=2
read(y,1)	Ok	RTM(y)=1
write(y,1)	Ok	WTM(y)=1
read(x,2)	Ok	RTM(x)=2
write(z,2)	Ok	WTM(z)=2

Non viene abortita nessuna transazione.

r1(x), w1(x), w3(x), r2(y), r3(y), w3(y), w1(y), r2(x)

Operazione	Risposta	Nuovo Valore
read(x,1)	Ok	RTM(x)=1
write(x,1)	Ok	WTM(x)=1
write(x,3)	Ok	WTM(x)=3
read(y,2)	Ok	RTM(y)=2
read(y,3)	Ok	RTM(y)=3
write(y,3)	Ok	WTM(y)=3
write(y,1)	t1 aborted	
read(x,2)	t2 aborted	