Laurea in Informatica
A.A. 2021-2022

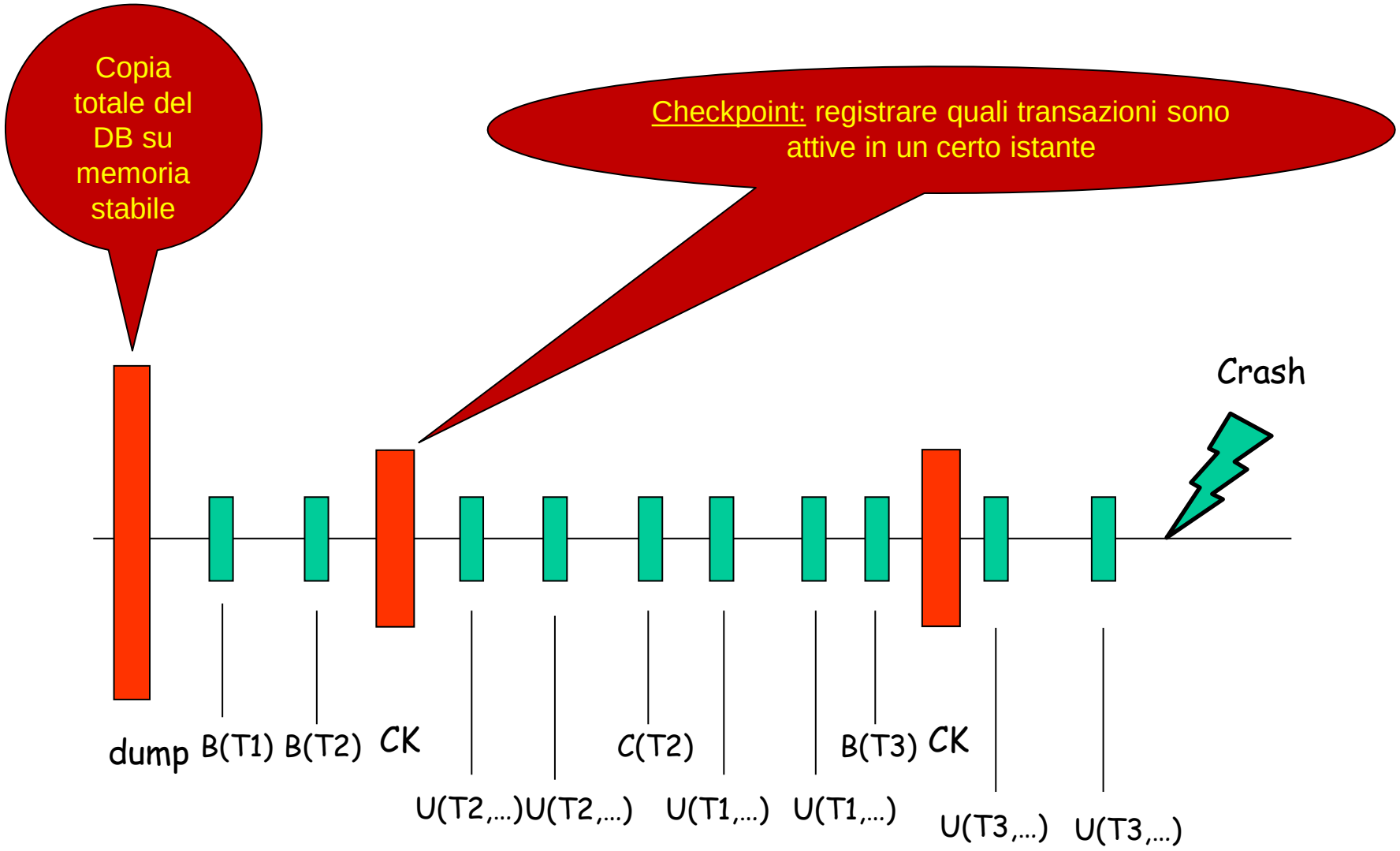
Corso "Base di Dati"

Esercitazione Transazioni

Prof. Massimiliano de Leoni

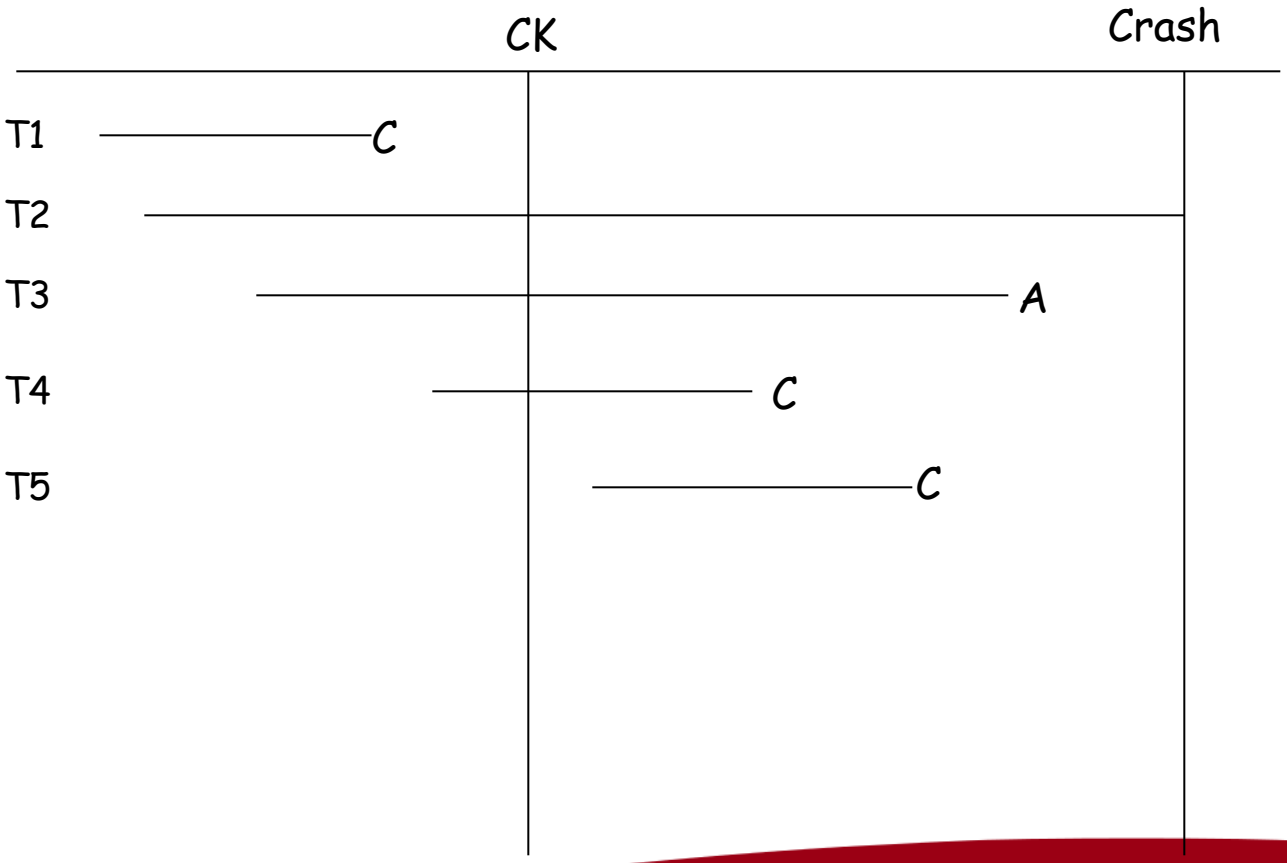
UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Log per guasti dispositivo



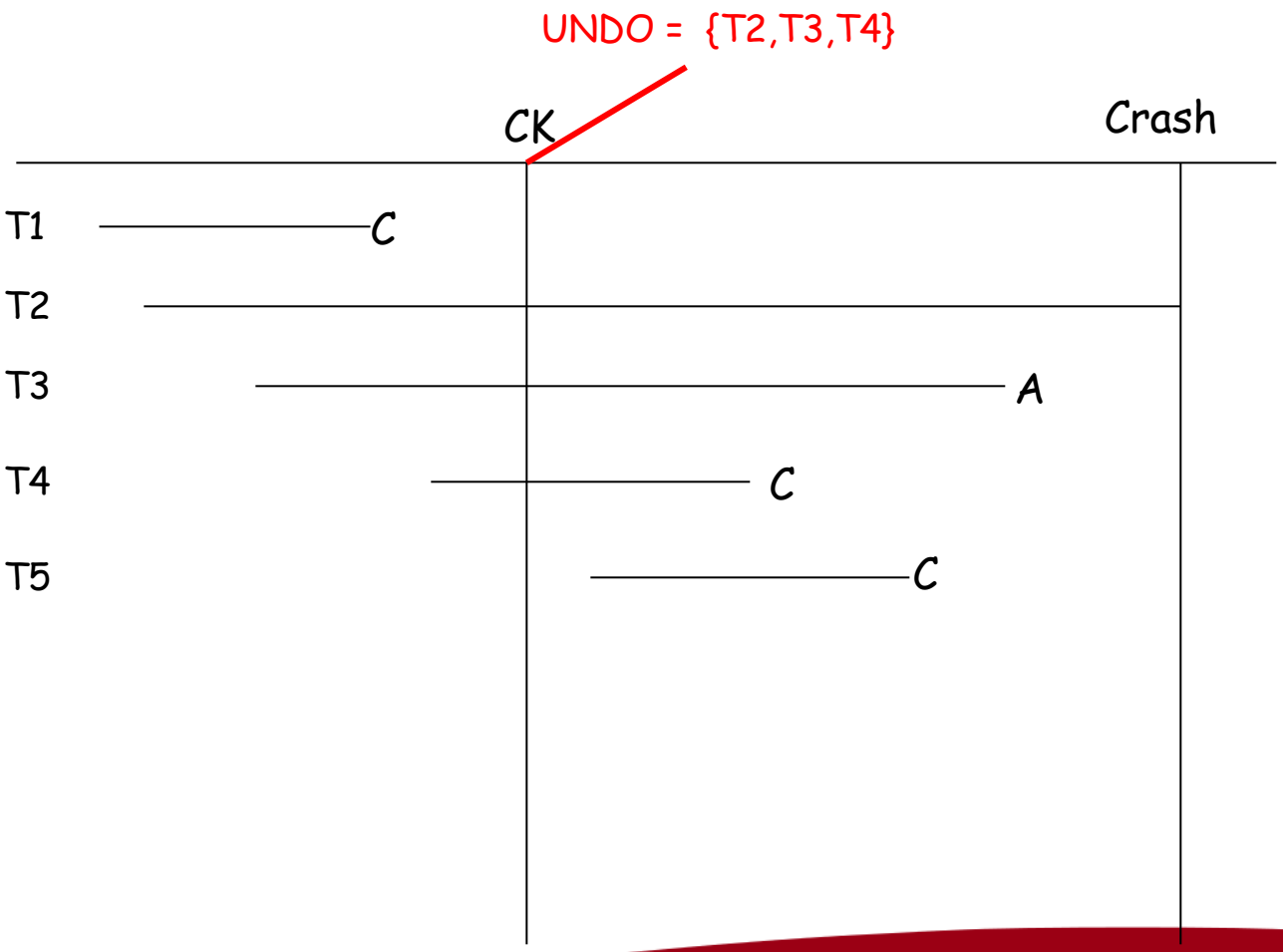
Esempio di warm restart

B(T1)
B(T2)
U(T2, O1, B1, A1)
I(T1, O2, A2)
B(T3)
C(T1)
B(T4)
U(T3,O2,B3,A3)
U(T4,O3,B4,A4)
CK(T2,T3,T4)
C(T4)
B(T5)
U(T3,O3,B5,A5)
U(T5,O4,B6,A6)
D(T3,O5,B7)
A(T3)
C(T5)
I(t2,O6,A8)



1. Ricerca dell'ultimo checkpoint

B(T1)
B(T2)
U(T2, O1, B1, A1)
I(T1, O2, A2)
B(T3)
C(T1)
B(T4)
U(T3,O2,B3,A3)
U(T4,O3,B4,A4)
CK(T2,T3,T4)
C(T4)
B(T5)
U(T3,O3,B5,A5)
U(T5,O4,B6,A6)
D(T3,O5,B7)
A(T3)
C(T5)
I(T2,O6,A8)



2. Costruzione degli insiemi UNDO e REDO

B(T1)
B(T2)
8. U(T2, O1, B1, A1)
I(T1, O2, A2)
B(T3)
C(T1)
B(T4)
7. U(T3,O2,B3,A3)
9. U(T4,O3,B4,A4)
CK(T2,T3,T4)
1. C(T4)
2. B(T5)
6. U(T3,O3,B5,A5)
10. U(T5,O4,B6,A6)
5. D(T3,O5,B7)
A(T3)
3. C(T5)
4. I(T2,O6,A8)

0. UNDO = {T2,T3,T4}. REDO = {}

1. C(T4) → UNDO = {T2, T3}. REDO = {T4}

2. B(T5) → UNDO = {T2,T3,T5}. REDO = {T4}

3. C(T5) → UNDO = {T2,T3}. REDO = {T4, T5}

3. Fase UNDO

B(T1)
B(T2)
8. U(T2, O1, B1, A1)
I(T1, O2, A2)
B(T3)
C(T1)
B(T4)
7. U(T3,O2,B3,A3)
9. U(T4,O3,B4,A4)
CK(T2,T3,T4)
1. C(T4)
2. B(T5)
6. U(T3,O3,B5,A5)
10. U(T5,O4,B6,A6)
5. D(T3,O5,B7)
A(T3)
3. C(T5)
4. I(T2,O6,A8)

0. UNDO = {T2,T3,T4}. REDO = {}

---

1. C(T4) → UNDO = {T2, T3}. REDO = {T4}
2. B(T5) → UNDO = {T2,T3,T5}. REDO = {T4}
3. C(T5) → UNDO = {T2,T3}. REDO = {T4, T5}

Setup

---

4. D(O6)
5. O5 =B7
6. O3 = B5
7. O2 =B3
8. O1=B1

Undo

4. Fase REDO

B(T1)
B(T2)
8. U(T2, O1, B1, A1)
I(T1, O2, A2)
B(T3)
C(T1)
B(T4)
7. U(T3,O2,B3,A3)
9. U(T4,O3,B4,A4)
CK(T2,T3,T4)
1. C(T4)
2. B(T5)
6. U(T3,O3,B5,A5)
10. U(T5,O4,B6,A6)
5. D(T3,O5,B7)
A(T3)
3. C(T5)
4. I(T2,O6,A8)

0. UNDO = {T2,T3,T4}. REDO = {}

1. C(T4) → UNDO = {T2, T3}. REDO = {T4}

2. B(T5) → UNDO = {T2,T3,T5}. REDO = {T4}

3. C(T5) → UNDO = {T2,T3}. REDO = {T4, T5}

Setup

4. D(O6)

5. O5 =B7

6. O3 = B5

7. O2 =B3

8. O1=B1

Undo

9. O3 = A4

10. O4 = A6

Redo

Esercizio 1

Descrivere la ripresa a caldo, indicando la costituzione progressiva degli insiemi di UNDO e REDO e le azioni di recovery, a fronte del seguente log:

DUMP, B(T1), B(T2), B(T3), I(T1, O1, A1), D(T2, O2, B2), B(T4),
U(T4, O3, B3, A3), U(T1, O4, B4, A4), C(T2), CK(T1, T3, T4), B(T5), B(T6),
U(T5, O5, B5, A5), A(T3), CK(T1, T4, T5, T6), B(T7), A(T4),
U(T7, O6, B6, A6), U(T6, O3, B7, A7), B(T8), A(T7), guasto

Esercizio 1

Soluzione:

1) Per prima cosa bisogna percorrere il log a ritroso fino al più recente record di check-point:

CK(T1,T4,T5,T6)

Si costruiscono gli insiemi di UNDO e di REDO:

UNDO= { T1, T4, T5, T6 } REDO={}

Esercizio 1

Soluzione:

2) Il log viene percorso in avanti, aggiornando i due insiemi:

B(T7) UNDO= { T1, T4, T5, T6, T7 } REDO={}

A(T4) UNDO= { T1, T4, T5, T6, T7 } REDO={}

B(T8) UNDO= { T1, T4, T5, T6, T7, T8 } REDO={}

A(T7) UNDO= { T1, T4, T5, T6, T7 } REDO={}

Esercizio 1

Soluzione:

3) Il log viene ripercorso ancora a ritroso, fino all'operazione I(T1,O1,A1) inclusa, eseguendo le seguenti operazioni:

O3=B7
O6=B6
O5=B5
O4=B4
O3=B3
Delete O1

4) Il log viene ripercorso in avanti per rieseguire le operazioni di REDO, ma essendo vuoto questo insieme, nessuna operazione verrà eseguita.

Esercizio 2

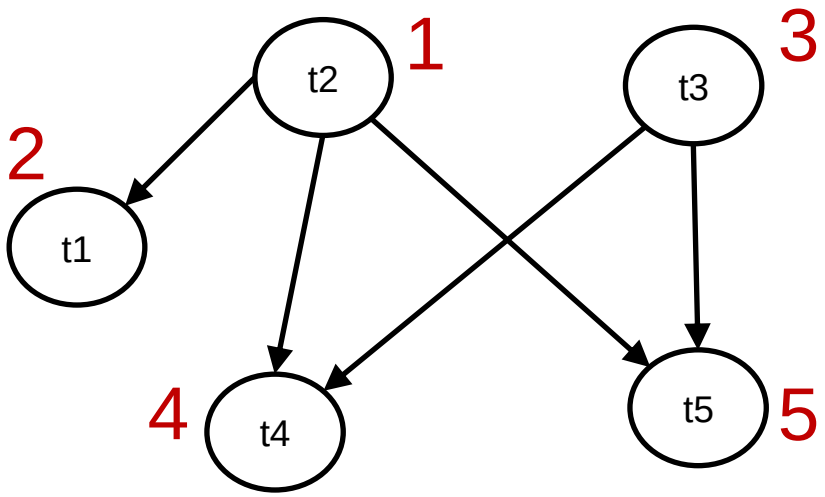
Considera il seguente schedule:

S = r2(x) r1(x) w3(t) w1(x) r3(y) r4(t) r2(y) w2(z) w5(y) w4(z)

S è conflict serializable? Se sì, mostrare uno schedule che è conflict-equivalente

Esercizio 2: Conflict Serializzabile?

S = r2(x) r1(x) w3(t) w1(x) r3(y) r4(t) r2(y) w2(z) w5(y) w4(z)



Schedule seriale conflict-equivalente:

r2(x) r2(y) w2(z) r1(x) w1(x) w3(t) r3(y) r4(t) w4(z) w5(y)

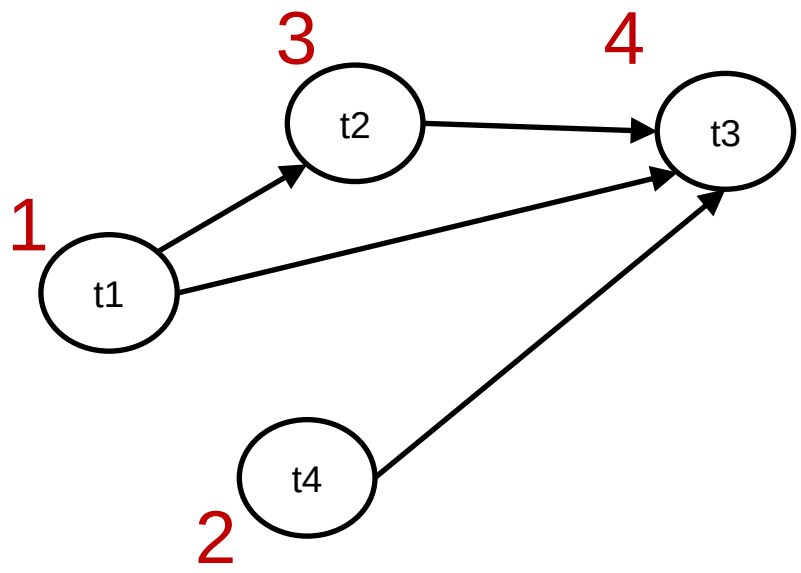
Esercizio 3

Dato lo schedule
S = r1(x) w2(x) r3(x) w1(u) w3(v) r3(y) r2(y) w3(u) w4(t) w3(t).

Dire se è confict-serializzabile e trovare uno schedule seriale conflict-equivalente

Esercizio 3

S = r1(x) w2(x) r3(x) w1(u) w3(v) r3(y) r2(y) w3(u) w4(t) w3(t).

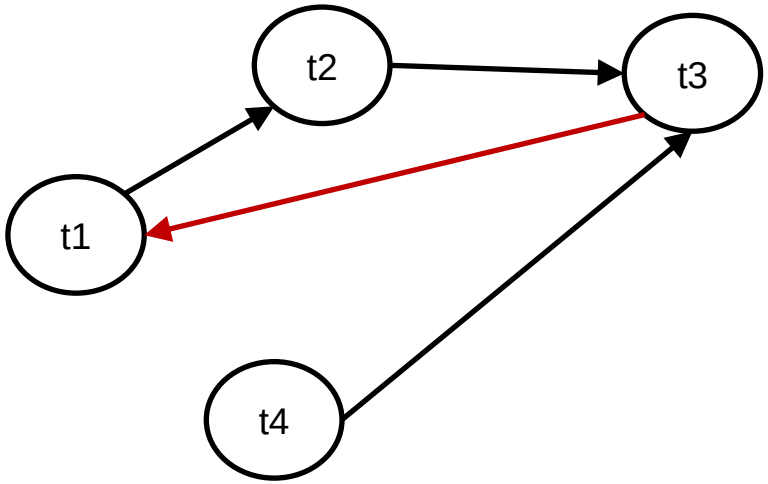


Schedule seriale conflict-equivalente:

r1(x) w1(u) w4(t) w2(x) r2(y) r3(x) w3(v) r3(y) w3(u) w3(t)

Esercizio 3 bis: spostiamo operaz. w1(u) nel punto in rosso

S = r1(x) w2(x) r3(x) w3(v) r3(y) r2(y) w3(u) w1(u) w4(t) w3(t).



Ciclo <t1, t2, t3, t1> ➔ non più conflict serializzabile

- L'operazione non può essere effettuata in quel punto
- Prima di effettuarla, occorre "rompere" il ciclo, cioè fare il commit o abort di t2 o t3 per togliere il nodo delle transazioni attive dal grafo

Esercizio 4

Indicare se i seguenti schedule sono VSR.

1. r1(x), r2(y), w1(y), r2(x) w2,(x)
2. r1(x), r2(y), w1(x), w1(y), r2(x) w2,(x)
3. r1(x), r1(y), r2(y), w2(z), w1(z), w3(z), w3(x)
4. r1(y), r1(y), w2(z), w1(z), w3(z), w3(x), w1(x)

Esercizio 4

Soluzione:  = Elemento Relazione LEGGE - DA

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

Questo schedule non è VSR, perché i due schedule seriali:

S1: r1(x), w1(y), r2(y), r2(x), w2(x) e 

S2: r2(y), r2(x), w2(x), r1(x), w1(y) 

non sono view-equivalenti con lo schedule dato. Hanno entrambi una differente relazione LEGGE - DA

Esercizio 4

Soluzione:

2. r1(x), r2(y), w1(x), w1(y), r2(x) w2,(x)

Questo schedule non è VSR perché gli schedule

S1: r1(x), w1(x), w1(y), r2(y), r2(x), w2(x)

S2: r2(y), r2(x), w2(x), r1(x), w1(x),w1(y)

hanno entrambi una differente relazione LEGGE – DA

Esercizio 4

Soluzione:

3. r1(x), r1(y), r2(y), w2(z), w1(z), w3(z), w3(x)

Questo schedule è VSR e view-equivalente allo schedule seriale, in quanto sono caratterizzati dalle stesse scritture finali e dalle stesse relazioni LEGGI –DA.

S: r2(y), w2(z), r1(x), r1(y), w1(z), w3(z), w3(x)

Esercizio 4

Soluzione:

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

Si noti che 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.

Esercizio 5

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)

Esercizio 5

Soluzione:

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

Questo schedule è sia VSR che CSR, ed è conflict-equivalente (e view-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.