Basi di Dati: Complementi

Docente: Prof. Pierangela Samarati

Appello di Maggio online – 22 Maggio 2010

Tempo a disposizione 2:00h

Soluzioni

Domanda 1)
Elencare e descrivere in modo completo le *proprietà ACIDe* delle transazioni.
Dire quali delle proprietà possono essere compromesse dalla *distribuzione dei dati* e quali soluzioni si possono utilizzare per garantire il loro soddisfacimento.

Domanda 2)
Rispondere in modo preciso e completo alle seguenti domande.

1. Elencare e descrivere brevemente le *principali primitive* del *gestore del buffer*.
2. llustrare la tecnica di *lock gerarchico* indicando i diversi tipi di lock ed il loro significato e le regole che devono essere rispettate dal protocollo di locking.
3. Spiegare cosa significa il livello *repeatable read* in SQL e dire cosa sono i *lock di predicato* e a cosa servono.

Domanda 3)
Dato un insieme di transazioni, uno schedule $S1$ che è stato generato da uno scheduler basato su *timestamp monoversione*, uno schedule $S2$ che è stato prodotto da uno scheduler *2PL*, e uno schedule $S3$ che è garantito essere *VSR ma non CSR*, si indichi per ciascuna affermazione riportata nella tabella allegata se è sicuramente vera (*Vero*), sicuramente falsa (*Falso*), o se *Non è possibile determinarlo* (potrebbe essere vera oppure falsa).

Foglio per l'Domanda 3)

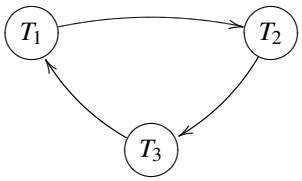
Dato un insieme di transazioni, uno schedule $S1$ che è stato generato da uno scheduler basato su *timestamp monoversione*, uno schedule $S2$ che è stato prodotto da uno scheduler *2PL*, e uno schedule $S3$ che è garantito essere *VSR ma non CSR*, si indichi per ciascuna affermazione riportata nella tabella allegata se è sicuramente vera (*Vero*), sicuramente falsa (*Falso*), o se *Non è possibile determinarlo* (potrebbe essere vera oppure falsa).

1. $S1$ è CSR
   - ☑ Vero
   - ☐ Falso
   - ☐ Non è possibile determinarlo
2. $S1$ sarebbe stato accettato da uno scheduler basato su timestamp multiversione.
   - ☑ Vero
   - ☐ Falso
   - ☐ Non è possibile determinarlo
3. $S2$ è stato generato da uno scheduler basato su timestamp monoversione
   - ☐ Vero
   - ☑ Falso
   - ☐ Non è possibile determinarlo
4. $S2$ non è VSR
   - ☐ Vero
   - ☑ Falso
   - ☐ Non è possibile determinarlo
5. $S3$ è 2PL
   - ☐ Vero
   - ☑ Falso
   - ☐ Non è possibile determinarlo
6. $S3$ non può essere stato generato da uno scheduler basato su timestamp
   - ☑ Vero
   - ☐ Falso
   - ☐ Non è possibile determinarlo

Domanda 4)
Si considerino i seguenti schemi relazionali:

IMPIEGATO(Matricola, Stipendio, Contributi, CodiceUfficio)
UFFICIO(Codice, MatricolaResponsabile, Palazzina, Piano, TtotaleContributiUfficio)

Scrivere un sistema di trigger *terminante* che abbia il seguente grafo di attivazione:




```
create trigger T1
after update of Stipendio on Impiegato
for each row
when 3000 < (select Stipendio
              from   Impiegato
              where  Matricola = new.Matricola)
begin
     update Impiegato
     set    Contributi = Stipendio * 0.5
     where  Matricola = new.Matricola
end;

create trigger T2
after update of Contributi on Impiegato
for each row
begin
     update Ufficio
     set    TotaleContributiUfficio = (select sum(Contributi)
                                       from   Impiegato
                                       where CodiceUfficio = new.CodiceUfficio)
     where  Codice = new.CodiceUfficio
end;

create trigger T3
after update of TotaleContributiUfficio on Ufficio
for each row
when 50000 < (select TotaleContributiUfficio
               from   Ufficio
               where  Codice = new.Codice)
begin
     update Impiegato
     set    Stipendio = 0.95 * Stipendio
     where  CodiceUfficio = new.Codice
end;
```

Esercizio 1)
Avendo le seguenti informazioni riguardo una **ripresa a caldo**, si indichi per ciascuna affermazione riportata nella tabella allegata se è sicuramente vera (*Vero*), sicuramente falsa (*Falso*), o se *non è possibile determinarlo* (potrebbe essere vera oppure falsa).

- record di checkpoint: CK(T1, T3, T5);
- insieme di UNDO: {T3, T6};
- insieme di REDO: {T1, T5, T7};
- operazioni di UNDO:
  - U(T6, O6, B6, A6) → O6 := B6
  - D(T3, O3, B3) → INSERT(O3), O3 := B3
  - I(T3, O4, A4) → DELETEO4
- operazioni di REDO:
  - U(T1, O1, B1, A1) → O1 := A1
  - I(T5, O5, A5) → INSERT(O5), O5 := A5
  - D(T7, O7, B7) → DELETE(O7)

Foglio per l'Esercizio 1)

Si indichi per ciascuna affermazione riportata nella tabella allegata se è sicuramente vera (*Vero*), sicuramente falsa (*Falso*), o se *non è possibile determinarlo* (potrebbe essere vera oppure falsa).

1. l'operazione di *update* di O1 da parte di T1 segue nel log l'operazione di *insert* di O5 da parte di T5
   - ☐ Vero
   - ☑ Falso
   - ☐ Non è possibile determinarlo
2. nel log esistono almeno due operazioni da parte della transazione T3
   - ☑ Vero
   - ☐ Falso
   - ☐ Non è possibile determinarlo
3. il log contiene un record di *abort* per la transazione T3
   - ☐ Vero
   - ☐ Falso
   - ☑ Non è possibile determinarlo
4. l'operazione di *insert* di O5 da parte di T5 segue nel log l'operazione di *insert* di O4 da parte di T3
   - ☐ Vero
   - ☐ Falso
   - ☑ Non è possibile determinarlo
5. supponendo che nel log esista una transazione T2, la transazione T2 è iniziata (*begin transaction*) prima del *checkpoint* e ha fatto *abort* dopo il *checkpoint*
   - ☐ Vero
   - ☑ Falso
   - ☐ Non è possibile determinarlo
6. la transazione T6 è iniziata (*begin transaction*) dopo che la transazione T3 è iniziata (*begin transaction*)
   - ☑ Vero
   - ☐ Falso
   - ☐ Non è possibile determinarlo
7. la transazione T5 è iniziata (*begin transaction*) dopo il *checkpoint*
   - ☐ Vero
   - ☑ Falso
   - ☐ Non è possibile determinarlo
8. la transazione T6 ha fatto *commit* dopo il *checkpoint*
   - ☐ Vero
   - ☑ Falso
   - ☐ Non è possibile determinarlo
9. l'operazione di *update* di O6 da parte di T6 segue nel log l'operazione di *insert* di O4 da parte di T3
   - ☑ Vero
   - ☐ Falso
   - ☐ Non è possibile determinarlo
10. la transazione T5 è iniziata (*begin transaction*) e ha fatto *commit* dopo che la transazione T3 è inziata (*begin transaction*)
    - ☐ Vero
    - ☐ Falso
    - ☑ Non è possibile determinarlo

Esercizio 2)

Dati i seguenti schedule:

1. $r_1(z)\ r_2(z)\ r_1(t)\ r_4(y)\ w_4(y)\ w_3(t)\ w_3(x)\ r_3(x)\ w_2(y)\ r_1(y)\ w_3(z)$
2. $r_1(t)\ w_1(x)\ w_3(y)\ r_2(y)\ w_2(t)\ w_1(t)\ r_4(t)\ w_1(z)\ r_3(x)\ r_4(y)\ w_4(t)\ r_1(z)$

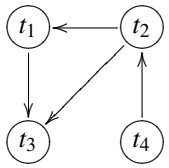
Si dica se gli schedule sono *VSR* e/o *CSR*, indicando (qualora esistano) *tutti* gli schedule seriali equivalenti. Si svolga l'esercizio illustrando dettagliatamente il processo/ragionamento seguito.

Schedule 1

1. Relazioni legge-da

LETTURA	LEGGE-DA	VINCOLI	ALTRI VINCOLI
$r_1(z)$	–		$1 \rightarrow 3$ (altrimenti introdurrebbe una leggi-da)
$r_2(z)$	–		$2 \rightarrow 3$ (altrimenti introdurrebbe una leggi-da)
$r_1(t)$	–		$1 \rightarrow 3$ (altrimenti introdurrebbe una leggi-da)
$r_4(y)$	–		$4 \rightarrow 2$ (altrimenti introdurrebbe una leggi-da)
$r_3(x)$	$w_3(x)$		
$r_1(y)$	$w_2(y)$	$2 \rightarrow 1$	$4 \rightarrow 2$ oppure $1 \rightarrow 4$ (siccome la prima è già verificata delle scritture finali, si scarta la seconda)

RISORSA	SCRITTURA FINALE	ALTRE SCRITTURE	VINCOLI
x	$w_3(x)$		
y	$w_2(y)$	$w_4(y)$	$4 \rightarrow 2$
z	$w_3(z)$		
t	$w_2(t)$		



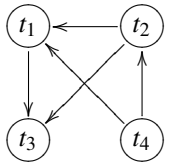
Nel grafo non è presente alcun ciclo e quindi lo schedule potrebbe essere *VSR*.

Esiste un solo possibile schedule seriale view-equivalente a quello dato: t_4, t_2, t_1, t_3 che presenta le stesse relazioni leggi-da e le stesse scritture finali, dello schedule analizzato, che risulta quindi *VSR*.

2. Valutiamo ora i conflitti presenti nello schedule per verificare se è anche *CSR*:

- $r_1(z), w_3(z)$
- $r_2(z), w_3(z)$
- $r_1(t), w_3(t)$
- $r_4(y), w_2(y)$
- $w_4(y), w_2(y)$
- $w_4(y), r_1(y)$
- $w_2(y), r_1(y)$

Il grafo dei conflitti riportato di seguito è aciclico.



Si conclude che lo schedule è *CSR*. Lo schedule seriale conflict equivalente a quello dato è t_4, t_2, t_1, t_3.

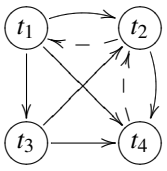
Schedule 2

1. Relazioni legge-da

LETTURA	LEGGE-DA	VINCOLI	ALTRI VINCOLI
$r_1(t)$	–		$1 \rightarrow 2$ (altrimenti introdurrebbe una leggi-da) $1 \rightarrow 4$ (altrimenti introdurrebbe una leggi-da)
$r_2(y)$	$w_3(y)$	$3 \rightarrow 2$	
$r_4(t)$	$w_1(t)$	$1 \rightarrow 4$	$2 \rightarrow 1$ oppure $4 \rightarrow 2$ (nessuna delle due può essere verificata, la prima per $r_1(t)$, la seconda per $w_4(t)$)
$r_3(x)$	$w_1(x)$	$1 \rightarrow 3$	
$r_4(y)$	$w_3(y)$	$3 \rightarrow 4$	
$r_1(z)$	$w_1(z)$		

Scritture finali

RISORSA	SCRITTURA FINALE	ALTRE SCRITTURE	VINCOLI
x	$w_1(x)$		
y	$w_3(y)$		
z	$w_1(z)$		
t	$w_4(t)$	$w_1(t)$, $w_2(t)$	$2 \rightarrow 4$ $1 \rightarrow 4$



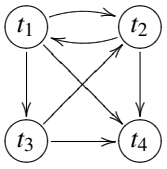
Considerando separatamente ciascuna delle frecce tratteggiate, i due grafi ottenuti sono entrambi ciclici. Quindi lo schedule non può essere *VSR*.

Non essendo *VSR* non è neppure *CSR*.

2. Valutiamo ora i conflitti presenti nello schedule per verificare che non sia effettivamente *CSR*:

   - $r_1(t), w_2(t)$
   - $r_1(t), w_4(t)$
   - $w_1(x), r_3(x)$
   - $w_3(y), r_2(y)$
   - $w_3(y), r_4(y)$
   - $w_2(t), w_1(t)$
   - $w_2(t), r_4(t)$
   - $w_2(t), w_4(t)$
   - $w_1(t), r_4(t)$
   - $w_1(t), w_4(t)$

   Il grafo dei conflitti riportato di seguito è ciclico.



   Si conclude che lo schedule non è *CSR*.

Esercizio 3)

Dato poi il seguente XML schema:

```
<?xml version="1.0"?>
<schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="catalogo">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="libro" maxOccurs="unbounded" minOccurs="0">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="titolo" type="xs:string" />
              <xs:element name="autore" type="xs:string" maxOccurs="5" minOccurs="0" />
              <xs:element name="editore" type="xs:string" />
              <xs:element name="prezzo" type="xs:decimal" />
            </xs:sequence>
            <xs:attribute name="disponibile" type="xs:boolean" />
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</schema>
```

Scrivere una interrogazione *XQuery* che restituisca tutti i libri disponibili editi da *Feltrinelli* in ordine crescente di prezzo.

```
for $lib in document("catalogo.xml")//libro
where $lib/editore="Feltrinelli"
       and $lib/@disponibile="true"
order by $lib/prezzo descending
return $lib
```

Esercizio 4)
Si consideri un controllo di concorrenza basato su timestamp. Si supponga, quando non diversamente indicato, l'inizializzazione di RTM e WTM al tempo 0.

1. Date le seguenti sequenze di richieste (e risposte del sistema) in un sistema che utilizza controllo di concorrenza basato su timestamp *mono-versione*, elencare tutti gli intervalli (valori *minimi* e *massimi*) di valori che potrebbe assumere *T1*.

Operazione	Risposta
write(x, 7)	OK
read(x, 11)	OK
write(x, 10)	NO
read(x, **T1**)	OK
write(x, 18)	NO
read(x, 21)	OK

2. Date le seguenti sequenze di richieste (e risposte del sistema) in un sistema che utilizza controllo di concorrenza basato su timestamp *mono-versione*, elencare tutti gli intervalli (valori *minimi* e *massimi*) di valori che potrebbe assumere *T2*.

Operazione	Risposta
write(x, 5)	OK
read(x, 10)	OK
read(x, 15)	OK
write(x, **T2**)	NO
read(x, 18)	OK

3. Date le seguenti sequenze di richieste (e risposte del sistema) in un sistema che utilizza controllo di concorrenza basato su timestamp *multi-versione*, elencare tutti gli intervalli (valori *minimi* e *massimi*) di valori che potrebbe assumere *T3*.

Operazione	Risposta
write(x, 10)	OK
read(x, 5)	OK
read(x, 15)	OK
write(x, **T3**)	OK

4. Date le seguenti sequenze di richieste (e risposte del sistema) in un sistema che utilizza controllo di concorrenza basato su timestamp *multi-versione*, elencare tutti gli intervalli (valori *minimi* e *massimi*) di valori che potrebbe assumere *T4*.

Operazione	Risposta
write(x, 7)	OK
read(x, 4)	OK
read(x, **T4**)	OK
write(x, 10)	NO
write(x, 17)	OK

1. $T1 \geq 19$
   Motivazioni:
   - $T1 \geq 8$ altrimenti l'operazione write(x,7) sarebbe stata accettata (sussunta dalla condizione seguente);
   - $T1 \geq 19$ altrimenti l'operazione write(x,18) sarebbe stata accettata;
2. $T2 \leq 14$
   Motivazioni:
   - altrimenti sarebbe stata accettata perché al momento della richiesta write(x,T2) i valori erano RTM(x)=15 e WTM(x)=5
3. $5 \leq T3 < 10$ e $T3 \geq 15$
   Motivazioni:
   Ci sono due versioni di *x*. Al momento della richiesta write(x,T3), i valori RTM e WTM delle diverse versioni sono:
   - RTM(x_0)=5 (supponendo inizializzazione al tempo 0)

- WTM(x_0)=0 (supponendo inizializzazione al tempo 0)
- RTM(x_{10})=15
- WTM(x_{10})=10

Una scrittura viene controllata rispetto al WTM della versione più recentemente creata e consentita se l'RTM di tale versione è minore del timestamp della transazione che richiede la scrittura:

- write(x,T3), con $T3 < 10$, controllata rispetto all'RTM di x_0, sarebbe stata concessa (risposta OK) solo per $T3 \geq 5$, mentre sarebbe stata rifiutata (risposta NO) per $T3 < 5$
- write(x,T3), con $10 \leq T3 < 15$, controllata rispetto a x_{10} sarebbe stata rifiutata (risposta NO)
- write(x,T3), con $15 \leq T3$, controllata rispetto a x_{10}, sarebbe stata concessa (risposta OK)

4. $10 < T4 \leq 17$
   Motivazioni: le operazioni di read vengono sempre accettate, le restrizioni sui valori di $T4$ devono essere controllate rispetto alle scritture che la lettura potrebbe influenzare. Al momento della richiesta read(x,T4) ci sono due versioni di x:

   - RTM(x_0)=4 (supponendo inizializzazione al tempo 0)
   - WTM(x_0)=0 (supponendo inizializzazione al tempo 0)
   - RTM(x_7)=7
   - WTM(x_7)=7

   La scrittura influenzata è quindi write(x,10), che deve essere negata, ne risulta che:

   - $T4 > 10$ altrimenti l'operazione write(x,10) sarebbe stata accettata;
   - $T4 \leq 17$ altrimenti l'operazione write(x,17) sarebbe stata rifiutata.