Basi di dati II

Esame — 20 settembre 2013 — Compito A

Rispondere su questo fascicolo.
Tempo a disposizione: due ore.

Cognome _____ **Nome** _____ **Matricola** _____

Domanda 1 (15%)
Per ciascuno degli schedule sotto riportati, indicare, scrivendo **sì** o **no** in ciascuna casella, a quali classi appartiene: S (seriale, rispetto a letture e scritture, non tenere conto dell'ordine degli avvii delle transazioni e dei commit), CSR (conflict-serializzabile), S2PL (cioè generabile da uno scheduler basato su 2PL stretto), MV (cioè generabile da uno scheduler multiversion con controllo di serializzabilità: "a serializable transaction cannot modify or lock rows changed by other transactions after the serializable transaction began"). Negli schedule, s_i indica l'inizio della transazione i e c_i il suo commit.

	S	CSR	S2PL	MV
$s_2, r_2(x), w_2(x), c_2, s_1, r_1(x), w_1(x), c_1$				
$s_2, r_2(x), w_2(x), s_1, r_1(x), w_1(x), c_2, c_1$				
$s_1, r_1(x), s_2, r_2(x), w_1(x), c_1, w_2(x), c_2$				
$s_1, r_1(x), s_2, r_2(x), w_2(x), r_2(y), w_2(y), c_2, r_1(y), c_1$				
$s_1, s_2, r_2(x), w_2(x), r_2(y), w_2(y), r_1(x), c_2, r_1(y), c_1$				

Domanda 2 (15%)
Si supponga si abbiano documenti che contengono informazioni su libri e autori, come il seguente:

```
<biblioteca>
   <libro>
      <titolo>Guerra e Pace</titolo> <autore>Lev Tolstoj</autore> <anno>1869</anno>
   </libro>
   <libro>
      <titolo>Anna Karenina</titolo> <autore>Lev Tolstoj</autore> <anno>1877</anno>
   </libro>
   <libro>
      <titolo>I Buddenbrook</titolo> <autore>Thomas Mann</autore> <anno>1901</anno>
   </libro>
   <libro>
      <titolo>La Montagna Incantata</titolo> <autore>Thomas Mann</autore> <anno>1924</anno>
   </libro>
   <libro>
      <titolo>Furore</titolo> <autore>John Steinbeck</autore> <anno>1939</anno>
   </libro>
</biblioteca>
```

Formulare le seguenti interrogazioni:

In XPath, trovare i titoli dei libri il cui autore è Lev Tolstoj e che sono stati scritti prima del 1875

In XQuery, produrre un documento che contiene, per ogni autore che abbia scritto almeno un romanzo prima del 1920, la lista di tali romanzi. Con riferimento al documento mostrato in precedenza, si vuole ottenere:

```
<autore>
   Lev Tolstoj
   <titolo>Guerra e Pace</titolo>
   <titolo>Anna Karenina</titolo>
</autore>
<autore>
   Thomas Mann
   <titolo>I Buddenbrook</titolo>
</autore>
```

Domanda 3 (15%)
Indicare il risultato di ciascuna delle seguenti interrogazioni XQuery:

```
let $i:= (3,2,1)
for $j in (1,2,3)
return <result>{$i} {$j}</result>
```

```
for $i in (1,2,3)
let $j:= (3,2,1)
return <result>{$i} {$j}</result>
```

```
for $i in (1,2,3)
for $j in (3,2,1)
return <result>{$i} {$j}</result>
```

```
let $i:= (1,2,3)
let $j:= (3,2,1)
return <result>{$i} {$j}</result>
```

Basi di dati II — 20 settembre 2013 — compito A

Domanda 4 (25%)
Considerare le operazioni di raggruppamento, specificate in SQL con clausole `GROUP BY` e funzioni aggregative sui gruppi. Ad esempio:

```
SELECT B, SUM(C) FROM R GROUP BY B
```

Queste operazioni vengono concettualmente eseguite in due passi (supponiamo per semplicità che, come nell'esempio, ci sia un solo attributo di raggruppamento e una sola funzione aggregativa)

1. partizionare le ennuple dell'operando sulla base dei valori dell'attributo di raggruppamento (`B` nell'esempio)
2. per ogni partizione, produrre una ennupla con (i) il valore dell'attributo di raggruppamento e (ii) il valore della funzione aggregativa (`SUM(C)` nell'esempio) applicata alle ennuple della partizione

Una possibile implementazione efficiente può essere ottenuta con una variante del mergesort a più vie, con porzioni ordinate e aggregazioni effettuate esaminando i valori affioranti di ciascuna partizione.

Considerare la relazione sotto schematizzata, sugli attributi `A`, `B` e `C`. Supponendo una disponibilità di buffer abbastanza ampia, ma non sufficiente a caricare in memoria l'intera relazione (supporre ad esempio una disponibilità di 4 buffer, con un fattore di blocco pari a 2 e quindi uno spazio occupato dalla relazione pari a 8 blocchi), considerare l'esecuzione dell'interrogazione sopra mostrata, in due passate (il che è possibile, visto che il numero di buffer è maggiore della radice del numero di blocchi).

Mostrare lo stato delle strutture in memoria centrale e secondaria dopo l'esecuzione di tre chiamate al metodo `next()` sullo scan che implementa l'interrogazione complessiva. In particolare, mostrare i "run" (cioè le porzioni di file ordinate durante prima passata — per comodità mostrare tutti gli attributi) memorizzati su disco e i buffer in memoria centrale, evidenziando per ciascun buffer il record corrente (il primo non ancora considerato). Mostrare anche i record prodotti dalle prime tre chiamate di `next()`.

Run su disco | Buffer | Record prodotti dalle prime tre `next()`

A	B	C
101	c	1
102	d	2
103	d	4
104	c	1
105	f	3
106	d	1
107	e	3
108	f	4
109	c	3
110	d	2
111	c	2
112	f	4
113	c	1
114	f	2
115	e	3
116	d	3

Basi di dati II — 20 settembre 2013 — compito A

Domanda 5 (15%)
Considerare una tabella `R` appena creata (e quindi vuota), con le seguenti ipotesi

- `R` è definita su due campi, `A` di lunghezza $a = 4$ byte e `B` di lunghezza $b = 14$ byte, senza vincoli espliciti di chiave (e quindi le operazioni si possono fare senza verifiche particolari);
- la struttura fisica utilizzata per `R` è heap, senza indici, con una memorizzazione a lunghezza fissa (in cui supponiamo che, oltre ai byte necessari per i campi, ne servano 2 ulteriori per la memorizzazione) e in cui si marcano come liberi gli spazi dei record eliminati, **senza riutilizzarli per successivi inserimenti, se non dopo una esplicita riorganizzazione**;
- il sistema utilizza blocchi di dimensione $D = 2$ Kbyte (approssimabili a 2000).

In tale contesto, supporre che vengano eseguite, nell'ordine, le seguenti operazioni

1. inserimento di $N = 200.000$ ennuple
2. eliminazione di $1/2 \times N = 100.000$ delle ennuple (sulla base di una condizione verificabile durante la scansione)
3. dopo la conclusione e la chiusura della scansione precedente, inserimento di altre N ennuple
4. riorganizzazione, con ricompattazione dei blocchi

Rispondere alle domande seguenti, indicando formule e valori numerici:

Fattore di blocco f per la relazione `R`:

Numero blocchi occupati da `R` dopo la prima serie di inserimenti (punto 1):

Numero blocchi occupati da `R` dopo le eliminazioni di cui al punto 2:

Numero blocchi occupati da `R` dopo la seconda serie di inserimenti (punto 3):

Numero blocchi occupati da `R` dopo la riorganizzazione (punto 4):

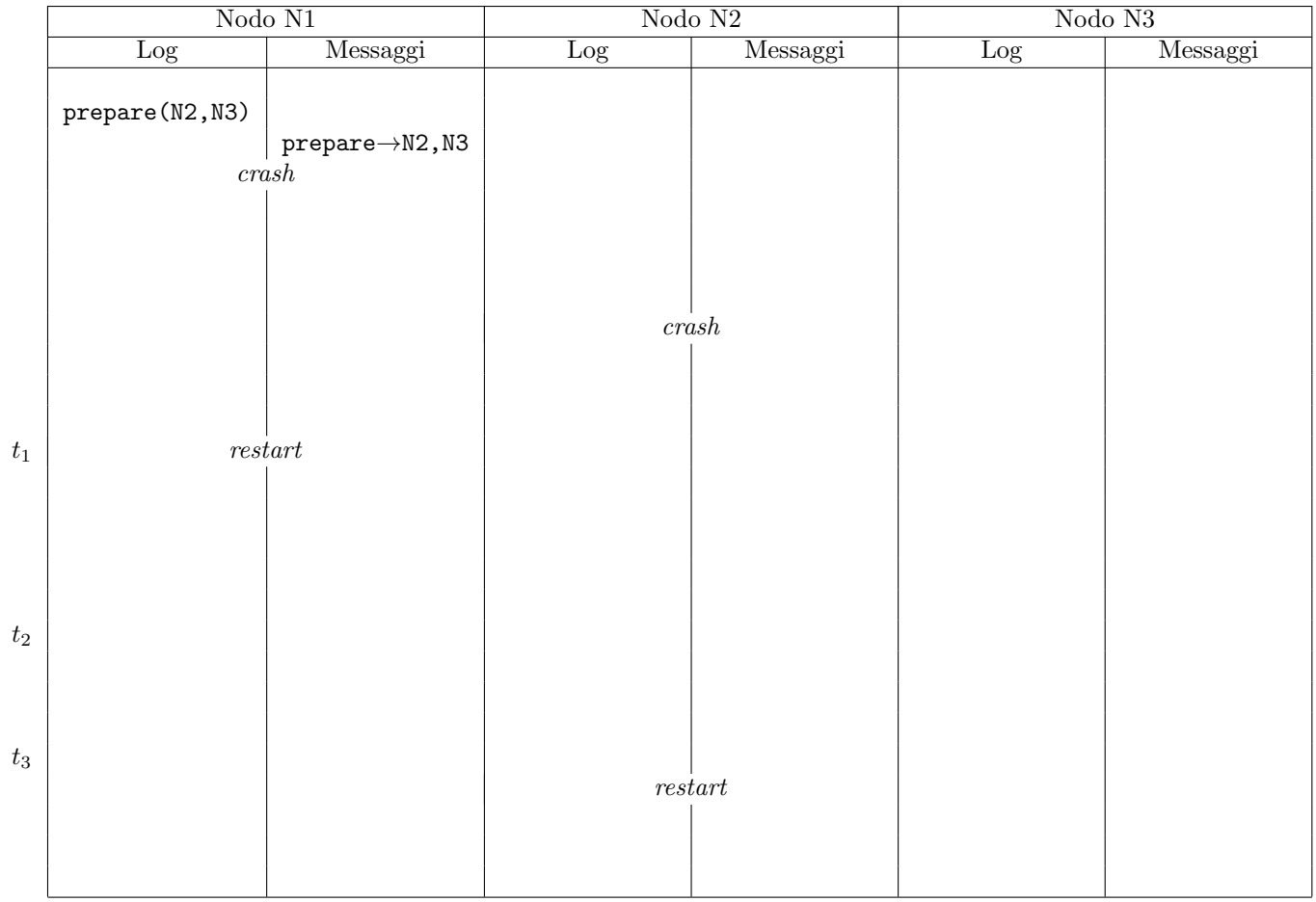
Numero di scritture in memoria secondaria (per le pagine dei dati e quelle del log) per la prima serie di inserimenti, supponendo che i record di log abbiano una lunghezza pari a circa il triplo di quella dei record del file, con riferimento ad un programma che utilizzi una transazione separata per ciascun inserimento

- numero di scritture di pagine di log:
- numero di scritture di pagine di dati:

Come nel caso precedente, ma con riferimento ad un programma che utilizzi un'unica transazione per tutti gli inserimenti (e supponendo che non vi siano altre transazioni attive)

- numero di scritture di pagine di log:
- numero di scritture di pagine di dati:

Domanda 6 (15%)
Considerare un sistema distribuito su cui viene eseguita una transazione che coinvolge tre nodi, un coordinatore N1 e due partecipanti N2 e N3. Dopo la richiesta di `prepare` da parte del coordinatore N1, il coordinatore stesso va in crash mentre i due partecipanti ricevono e rispondono correttamente, ma uno dei due, N2, va in crash subito dopo. Supporre che, dopo il crash, il coordinatore segua la strategia più semplice, abortendo la transazione. Indicare, nello schema sottostante, una possibile sequenza di scritture sui log e invio di messaggi, supponendo che entrambi i nodi siano ripristinati abbastanza presto. Per semplicità, si fa riferimento ad una sola transazione e quindi non c'è bisogno di indicarla. Per i messaggi si usi la notazione *tipo*→*destinatari* (come nell'esempio: `prepare`→`N2,N3`). Supporre che timeout per le varie fasi scattino all'incirca negli istanti indicati a sinistra della tabella e che il coordinatore, mentre è attivo, ricordi (senza scriverlo nel log) quali messaggi di ack ha ricevuto.



Eventuali commenti:

Basi di dati II

Esame — 20 settembre 2013 — Compito B

Rispondere su questo fascicolo.
Tempo a disposizione: due ore.

Cognome _____ **Nome** _____ **Matricola** _____

Domanda 1 (15%)
Per ciascuno degli schedule sotto riportati, indicare, scrivendo **sì** o **no** in ciascuna casella, a quali classi appartiene: S (seriale, rispetto a letture e scritture, non tenere conto dell'ordine degli avvii delle transazioni e dei commit), CSR (conflict-serializzabile), S2PL (cioè generabile da uno scheduler basato su 2PL stretto), MV (cioè generabile da uno scheduler multiversion con controllo di serializzabilità: "a serializable transaction cannot modify or lock rows changed by other transactions after the serializable transaction began"). Negli schedule, s_i indica l'inizio della transazione i e c_i il suo commit.

	S	CSR	S2PL	MV
$s_2, r_2(x), w_2(x), c_2, s_1, r_1(x), w_1(x), c_1$				
$s_1, r_1(x), s_2, r_2(x), w_1(x), c_1, w_2(x), c_2$				
$s_2, r_2(x), w_2(x), s_1, r_1(x), w_1(x), c_2, c_1$				
$s_1, s_2, r_2(x), w_2(x), r_2(y), w_2(y), r_1(x), c_2, r_1(y), c_1$				
$s_1, r_1(x), s_2, r_2(x), w_2(x), r_2(y), w_2(y), c_2, r_1(y), c_1$				

Basi di dati II — 20 settembre 2013 — compito B

Domanda 2 (15%)
Si supponga si abbiano documenti che contengono informazioni su libri e autori, come il seguente:

```
<biblioteca>
   <libro>
      <titolo>Guerra e Pace</titolo> <autore>Lev Tolstoj</autore> <anno>1869</anno>
   </libro>
   <libro>
      <titolo>Anna Karenina</titolo> <autore>Lev Tolstoj</autore> <anno>1877</anno>
   </libro>
   <libro>
      <titolo>I Buddenbrook</titolo> <autore>Thomas Mann</autore> <anno>1901</anno>
   </libro>
   <libro>
      <titolo>La Montagna Incantata</titolo> <autore>Thomas Mann</autore> <anno>1924</anno>
   </libro>
   <libro>
      <titolo>Furore</titolo> <autore>John Steinbeck</autore> <anno>1939</anno>
   </libro>
</biblioteca>
```

Formulare le seguenti interrogazioni:

In XPath, trovare i titoli dei libri il cui autore è Lev Tolstoj e che sono stati scritti prima del 1875

In XQuery, produrre un documento che contiene, per ogni autore che abbia scritto almeno un romanzo prima del 1920, la lista di tali romanzi. Con riferimento al documento mostrato in precedenza, si vuole ottenere:

```
<autore>
   Lev Tolstoj
   <titolo>Guerra e Pace</titolo>
   <titolo>Anna Karenina</titolo>
</autore>
<autore>
   Thomas Mann
   <titolo>I Buddenbrook</titolo>
</autore>
```

Domanda 3 (15%)
Indicare il risultato di ciascuna delle seguenti interrogazioni XQuery:

```
let $i:= (3,2,1)
for $j in (1,2,3)
return <result>{$i} {$j}</result>
```

```
for $i in (1,2,3)
let $j:= (3,2,1)
return <result>{$i} {$j}</result>
```

```
for $i in (1,2,3)
for $j in (3,2,1)
return <result>{$i} {$j}</result>
```

```
let $i:= (1,2,3)
let $j:= (3,2,1)
return <result>{$i} {$j}</result>
```

Basi di dati II — 20 settembre 2013 — compito B

Domanda 4 (25%)
Considerare le operazioni di raggruppamento, specificate in SQL con clausole `GROUP BY` e funzioni aggregative sui gruppi. Ad esempio:

```
SELECT B, SUM(C) FROM R GROUP BY B
```

Queste operazioni vengono concettualmente eseguite in due passi (supponiamo per semplicità che, come nell'esempio, ci sia un solo attributo di raggruppamento e una sola funzione aggregativa)

1. partizionare le ennuple dell'operando sulla base dei valori dell'attributo di raggruppamento (`B` nell'esempio)
2. per ogni partizione, produrre una ennupla con (i) il valore dell'attributo di raggruppamento e (ii) il valore della funzione aggregativa (`SUM(C)` nell'esempio) applicata alle ennuple della partizione

Una possibile implementazione efficiente può essere ottenuta con una variante del mergesort a più vie, con porzioni ordinate e aggregazioni effettuate esaminando i valori affioranti di ciascuna partizione.

Considerare la relazione sotto schematizzata, sugli attributi `A`, `B` e `C`. Supponendo una disponibilità di buffer abbastanza ampia, ma non sufficiente a caricare in memoria l'intera relazione (supporre ad esempio una disponibilità di 4 buffer, con un fattore di blocco pari a 2 e quindi uno spazio occupato dalla relazione pari a 8 blocchi), considerare l'esecuzione dell'interrogazione sopra mostrata, in due passate (il che è possibile, visto che il numero di buffer è maggiore della radice del numero di blocchi).

Mostrare lo stato delle strutture in memoria centrale e secondaria dopo l'esecuzione di tre chiamate al metodo `next()` sullo scan che implementa l'interrogazione complessiva. In particolare, mostrare i "run" (cioè le porzioni di file ordinate durante prima passata — per comodità mostrare tutti gli attributi) memorizzati su disco e i buffer in memoria centrale, evidenziando per ciascun buffer il record corrente (il primo non ancora considerato). Mostrare anche i record prodotti dalle prime tre chiamate di `next()`.

Run su disco | Buffer | Record prodotti dalle prime tre `next()`

A	B	C
101	c	2
102	d	2
103	d	4
104	c	2
105	f	3
106	d	1
107	e	3
108	f	4
109	c	3
110	d	2
111	c	2
112	f	4
113	c	1
114	f	2
115	e	3
116	d	3

Basi di dati II — 20 settembre 2013 — compito B

Domanda 5 (15%)
Considerare una tabella `T` appena creata (e quindi vuota), con le seguenti ipotesi

- `T` è definita su due campi, `A` di lunghezza $a = 6$ byte e `B` di lunghezza $b = 32$ byte, senza vincoli espliciti di chiave (e quindi le operazioni si possono fare senza verifiche particolari);
- la struttura fisica utilizzata per `T` è heap, senza indici, con una memorizzazione a lunghezza fissa (in cui supponiamo che, oltre ai byte necessari per i campi, ne servano 2 ulteriori per la memorizzazione) e in cui si marcano come liberi gli spazi dei record eliminati, **senza riutilizzarli per successivi inserimenti, se non dopo una esplicita riorganizzazione**;
- il sistema utilizza blocchi di dimensione $D = 4$ Kbyte (approssimabili a 4000).

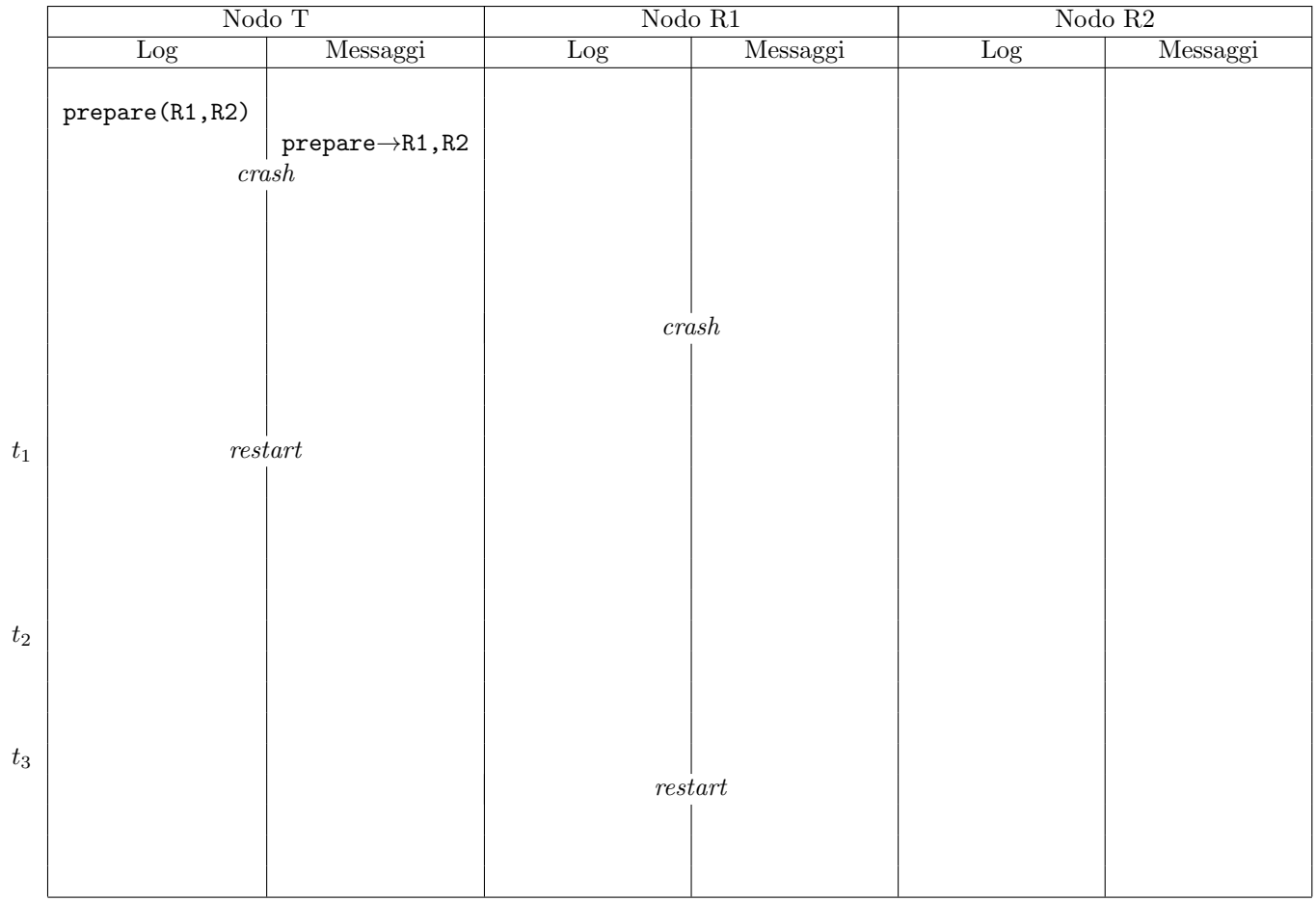
In tale contesto, supporre che vengano eseguite, nell'ordine, le seguenti operazioni

1. inserimento di $L = 200.000$ ennuple
2. eliminazione di $1/2 \times L = 100.000$ delle ennuple (sulla base di una condizione verificabile durante la scansione)
3. dopo la conclusione e la chiusura della scansione precedente, inserimento di altre L ennuple
4. riorganizzazione, con ricompattazione dei blocchi

Rispondere alle domande seguenti, indicando formule e valori numerici:

Fattore di blocco f per la relazione `T`:
Numero blocchi occupati da `T` dopo la prima serie di inserimenti (punto 1):
Numero blocchi occupati da `T` dopo le eliminazioni di cui al punto 2:
Numero blocchi occupati da `T` dopo la seconda serie di inserimenti (punto 3):
Numero blocchi occupati da `T` dopo la riorganizzazione (punto 4):
Numero di scritture in memoria secondaria (per le pagine dei dati e quelle del log) per la prima serie di inserimenti, supponendo che i record di log abbiano una lunghezza pari a circa il triplo di quella dei record del file, con riferimento ad un programma che utilizzi una transazione separata per ciascun inserimento • numero di scritture di pagine di log: • numero di scritture di pagine di dati:
Come nel caso precedente, ma con riferimento ad un programma che utilizzi un'unica transazione per tutti gli inserimenti (e supponendo che non vi siano altre transazioni attive) • numero di scritture di pagine di log: • numero di scritture di pagine di dati:

Domanda 6 (15%)
Considerare un sistema distribuito su cui viene eseguita una transazione che coinvolge tre nodi, un coordinatore T e due partecipanti R1 e R2. Dopo la richiesta di `prepare` da parte del coordinatore T, il coordinatore stesso va in crash mentre i due partecipanti ricevono e rispondono correttamente, ma uno dei due, R1, va in crash subito dopo. Supporre che, dopo il crash, il coordinatore segua la strategia più semplice, abortendo la transazione. Indicare, nello schema sottostante, una possibile sequenza di scritture sui log e invio di messaggi, supponendo che entrambi i nodi siano ripristinati abbastanza presto. Per semplicità, si fa riferimento ad una sola transazione e quindi non c'è bisogno di indicarla. Per i messaggi si usi la notazione *tipo*→*destinatari* (come nell'esempio: `prepare`→`R1,R2`). Supporre che timeout per le varie fasi scattino all'incirca negli istanti indicati a sinistra della tabella e che il coordinatore, mentre è attivo, ricordi (senza scriverlo nel log) quali messaggi di ack ha ricevuto.



Eventuali commenti:

Basi di dati II

Esame — 20 settembre 2013 — Compito A

Cenni sulle soluzioni

Rispondere su questo fascicolo.
Tempo a disposizione: due ore.

Cognome _____ **Nome** _____ **Matricola** _____

Domanda 1 (15%)
Per ciascuno degli schedule sotto riportati, indicare, scrivendo **sì** o **no** in ciascuna casella, a quali classi appartiene: S (seriale, rispetto a letture e scritture, non tenere conto dell'ordine degli avvii delle transazioni e dei commit), CSR (conflict-serializzabile), S2PL (cioè generabile da uno scheduler basato su 2PL stretto), MV (cioè generabile da uno scheduler multiversion con controllo di serializzabilità: "a serializable transaction cannot modify or lock rows changed by other transactions after the serializable transaction began"). Negli schedule, s_i indica l'inizio della transazione i e c_i il suo commit.

Soluzioni per il compito A

	S	CSR	S2PL	MV
$s_2, r_2(x), w_2(x), c_2, s_1, r_1(x), w_1(x), c_1$	sì	sì	sì	sì
$s_2, r_2(x), w_2(x), s_1, r_1(x), w_1(x), c_2, c_1$	sì	sì	no	no
$s_1, r_1(x), s_2, r_2(x), w_1(x), c_1, w_2(x), c_2$	no	no	no	no
$s_1, r_1(x), s_2, r_2(x), w_2(x), r_2(y), w_2(y), c_2, r_1(y), c_1$	no	no	no	sì *
$s_1, s_2, r_2(x), w_2(x), r_2(y), w_2(y), r_1(x), c_2, r_1(y), c_1$	sì	sì	no	sì *

(*) in questo caso $r_1(y)$ legge il valore di y all'inizio della transazione 1, cioË prima della scrittura $w_2(y)$

Basi di dati II — 20 settembre 2013 — compito A

Domanda 2 (15%)
Si supponga si abbiano documenti che contengono informazioni su libri e autori, come il seguente:

```
<biblioteca>
   <libro>
      <titolo>Guerra e Pace</titolo> <autore>Lev Tolstoj</autore> <anno>1869</anno>
   </libro>
   <libro>
      <titolo>Anna Karenina</titolo> <autore>Lev Tolstoj</autore> <anno>1877</anno>
   </libro>
   <libro>
      <titolo>I Buddenbrook</titolo> <autore>Thomas Mann</autore> <anno>1901</anno>
   </libro>
   <libro>
      <titolo>La Montagna Incantata</titolo> <autore>Thomas Mann</autore> <anno>1924</anno>
   </libro>
   <libro>
      <titolo>Furore</titolo> <autore>John Steinbeck</autore> <anno>1939</anno>
   </libro>
</biblioteca>
```

Formulare le seguenti interrogazioni:

In XPath, trovare i titoli dei libri il cui autore è Lev Tolstoj e che sono stati scritti prima del 1875

```
//libro[autore/text()="Lev Tolstoj"][anno<1870]
```

In XQuery, produrre un documento che contiene, per ogni autore che abbia scritto almeno un romanzo prima del 1920, la lista di tali romanzi. Con riferimento al documento mostrato in precedenza, si vuole ottenere:

```
<autore>
   Lev Tolstoj
   <titolo>Guerra e Pace</titolo>
   <titolo>Anna Karenina</titolo>
</autore>
<autore>
   Thomas Mann
   <titolo>I Buddenbrook</titolo>
</autore>
```

Possibile risposta:

```
let $nl := "&#10;"
for $a in distinct-values(//libro[anno<1920]/autore)
return
<autore>
   {$a}  {for $l in //libro[anno<1920]
          where $l/autore = $a
          return ($nl, " ", $l/titolo)
         }
</autore>
```

Domanda 3 (15%)
Indicare il risultato di ciascuna delle seguenti interrogazioni XQuery:

```
let $i:= (3,2,1)
for $j in (1,2,3)
return <result>{$i} {$j}</result>
```

Risultato

```
<result>3 2 1 1</result>
<result>3 2 1 2</result>
<result>3 2 1 3</result>
```

```
for $i in (1,2,3)
let $j:= (3,2,1)
return <result>{$i} {$j}</result>
```

Risultato

```
<result>1 3 2 1</result>
<result>2 3 2 1</result>
<result>3 3 2 1</result>
```

```
for $i in (1,2,3)
for $j in (3,2,1)
return <result>{$i} {$j}</result>
```

Risultato

```
<result>1 3</result>
<result>1 2</result>
<result>1 3</result>
<result>2 3</result>
<result>2 2</result>
<result>2 3</result>
<result>3 3</result>
<result>3 2</result>
<result>3 3</result>
```

```
let $i:= (1,2,3)
let $j:= (3,2,1)
return <result>{$i} {$j}</result>
```

Risultato

```
<result>1 2 3 3 2 1</result>
```

Basi di dati II — 20 settembre 2013 — compito A

Domanda 4 (25%)
Considerare le operazioni di raggruppamento, specificate in SQL con clausole `GROUP BY` e funzioni aggregative sui gruppi. Ad esempio:

```
SELECT B, SUM(C) FROM R GROUP BY B
```

Queste operazioni vengono concettualmente eseguite in due passi (supponiamo per semplicità che, come nell'esempio, ci sia un solo attributo di raggruppamento e una sola funzione aggregativa)

1. partizionare le ennuple dell'operando sulla base dei valori dell'attributo di raggruppamento (`B` nell'esempio)
2. per ogni partizione, produrre una ennupla con (i) il valore dell'attributo di raggruppamento e (ii) il valore della funzione aggregativa (`SUM(C)` nell'esempio) applicata alle ennuple della partizione

Una possibile implementazione efficiente può essere ottenuta con una variante del mergesort a più vie, con porzioni ordinate e aggregazioni effettuate esaminando i valori affioranti di ciascuna partizione.

Considerare la relazione sotto schematizzata, sugli attributi `A`, `B` e `C`. Supponendo una disponibilità di buffer abbastanza ampia, ma non sufficiente a caricare in memoria l'intera relazione (supporre ad esempio una disponibilità di 4 buffer, con un fattore di blocco pari a 2 e quindi uno spazio occupato dalla relazione pari a 8 blocchi), considerare l'esecuzione dell'interrogazione sopra mostrata, in due passate (il che è possibile, visto che il numero di buffer è maggiore della radice del numero di blocchi).

Mostrare lo stato delle strutture in memoria centrale e secondaria dopo l'esecuzione di tre chiamate al metodo `next()` sullo scan che implementa l'interrogazione complessiva. In particolare, mostrare i "run" (cioè le porzioni di file ordinate durante prima passata — per comodità mostrare tutti gli attributi) memorizzati su disco e i buffer in memoria centrale, evidenziando per ciascun buffer il record corrente (il primo non ancora considerato). Mostrare anche i record prodotti dalle prime tre chiamate di `next()`.

A	B	C
101	c	1
102	d	2
103	d	4
104	c	1
105	f	3
106	d	1
107	e	3
108	f	4
109	c	3
110	d	2
111	c	2
112	f	4
113	c	1
114	f	2
115	e	3
116	d	3

Run su disco

101	c	1
104	c	1
102	d	2
103	d	4
105	d	1
106	f	3

109	c	3
111	c	2
110	d	2
107	e	3
108	f	4
112	f	4

113	c	1
116	d	3
115	e	3
114	f	2

Buffer

105	d	1	←
106	f	3	

108	f	4	←
112	f	4	

115	e	3	←
114	f	2	

Record prodotti dalle prime 3 `next()`

c	8
d	13
e	9

Il numero ideale di buffer (da utilizzare sia nella prima sia nella seconda passata) è pari alla radice quadrata del numero dei blocchi e quindi a tre.
Nella prima passata, si costruiscono quindi tre run ordinati, ciascuno a partire da tre blocchi del file originario. L'ordinamento di ciascun run può essere effettuato in memoria centrale, usando tre buffer. Poiché il risultato della prima passata viene materializzato, vengono mostrati i run ordinati.
Nella seconda passata, si carica un blocco per ciascuno dei run e si fa calcola l'aggregazione usando nuovamente tre buffer e "consumando" i record via via. La seconda passata viene svolta in pipeline, e quindi i buffer sono mostrati con il contenuto che hanno quando è stato appena prodotto il terzo record.

Basi di dati II — 20 settembre 2013 — compito A

Domanda 5 (15%)
Considerare una tabella `R` appena creata (e quindi vuota), con le seguenti ipotesi

- `R` è definita su due campi, `A` di lunghezza $a = 4$ byte e `B` di lunghezza $b = 14$ byte, senza vincoli espliciti di chiave (e quindi le operazioni si possono fare senza verifiche particolari);
- la struttura fisica utilizzata per `R` è heap, senza indici, con una memorizzazione a lunghezza fissa (in cui supponiamo che, oltre ai byte necessari per i campi, ne servano 2 ulteriori per la memorizzazione) e in cui si marcano come liberi gli spazi dei record eliminati, **senza riutilizzarli per successivi inserimenti, se non dopo una esplicita riorganizzazione**;
- il sistema utilizza blocchi di dimensione $D = 2$ Kbyte (approssimabili a 2000).

In tale contesto, supporre che vengano eseguite, nell'ordine, le seguenti operazioni

1. inserimento di $N = 200.000$ ennuple
2. eliminazione di $1/2 \times N = 100.000$ delle ennuple (sulla base di una condizione verificabile durante la scansione)
3. dopo la conclusione e la chiusura della scansione precedente, inserimento di altre N ennuple
4. riorganizzazione, con ricompattazione dei blocchi

Rispondere alle domande seguenti, indicando formule e valori numerici:

Fattore di blocco f per la relazione `R`:

$$f = D/(a + b + 2) = 2000/20 = 100$$

Numero blocchi occupati da `R` dopo la prima serie di inserimenti (punto 1):

$$N/f = 200.000/100 = 2000$$

Numero blocchi occupati da `R` dopo le eliminazioni di cui al punto 2:

$$N/f = 200.000/100 = 2000$$

(lo spazio libero non viene riutilizzato)

Numero blocchi occupati da `R` dopo la seconda serie di inserimenti (punto 3):

$$2 \times N/f = 4000$$

(lo spazio libero non viene riutilizzato)

Numero blocchi occupati da `R` dopo la riorganizzazione (punto 4):

$$3/2 \times N/f = 3000$$

(lo spazio libero viene recuperato)

Numero di scritture in memoria secondaria (per le pagine dei dati e quelle del log) per la prima serie di inserimenti, supponendo che i record di log abbiano una lunghezza pari a circa il triplo di quella dei record del file, con riferimento ad un programma che utilizzi una transazione separata per ciascun inserimento

- numero di scritture di pagine di log:

  almeno $N = 200.000$, una per transazione

- numero di scritture di pagine di dati:

  con una strategia undo-only, una per transazione, $N = 200.000$; altrimenti, probabilmente di meno, anche solo $N/f = 2000$, una per blocco

Come nel caso precedente, ma con riferimento ad un programma che utilizzi un'unica transazione per tutti gli inserimenti (e supponendo che non vi siano altre transazioni attive)

- numero di scritture di pagine di log:

  dovrebbe essere possibile scrivere tutto il log al momento del commit e quindi (poiché i record sono grandi il triplo di quelli del file): $3 \times N/f = 6000$

- numero di scritture di pagine di dati:

  in ogni caso si dovrebbe avere $N/f = 2000$, una per blocco, o poco più

Domanda 6 (15%)
Considerare un sistema distribuito su cui viene eseguita una transazione che coinvolge tre nodi, un coordinatore N1 e due partecipanti N2 e N3. Dopo la richiesta di `prepare` da parte del coordinatore N1, il coordinatore stesso va in crash mentre i due partecipanti ricevono e rispondono correttamente, ma uno dei due, N2, va in crash subito dopo. Supporre che, dopo il crash, il coordinatore segua la strategia più semplice, abortendo la transazione. Indicare, nello schema sottostante, una possibile sequenza di scritture sui log e invio di messaggi, supponendo che entrambi i nodi siano ripristinati abbastanza presto. Per semplicità, si fa riferimento ad una sola transazione e quindi non c'è bisogno di indicarla. Per i messaggi si usi la notazione *tipo*→*destinatari* (come nell'esempio: `prepare`→N2,N3). Supporre che timeout per le varie fasi scattino all'incirca negli istanti indicati a sinistra della tabella e che il coordinatore, mentre è attivo, ricordi (senza scriverlo nel log) quali messaggi di ack ha ricevuto.

	Nodo N1		Nodo N2		Nodo N3	
	Log	Messaggi	Log	Messaggi	Log	Messaggi
	`prepare(N2,N3)`					
		`prepare`→N2,N3				
	crash					
			`ready`		`ready`	
				`ready`→N1		`ready`→N1
			crash			
t_1	*restart*					
	`abort`					
		`abort`→N2,N3				
					`abort`	
						`ack`→N1
t_2		`abort`→N2				
t_3		`abort`→N2				
			restart			
			`abort`			
				`ack`→N1		
	`complete`					

Eventuali commenti: