Basi di dati II
Esame — 25 settembre 2012 — Compito A

Rispondere su questo fascicolo. Tempo a disposizione: due ore e trenta minuti.

Cognome _____ **Nome** _____ **Matricola** _____ **Ordin.** _____

Domanda 1 (15%)
Si consideri una relazione *R(A B C D E)*, in cui gli attributi hanno tutti la stessa dimensione $a = 4$ Byte, molto più piccola della dimensione del blocco pari a P = 4 KByte. Si supponga che la relazione contenga $N = 2.000.000$ ennuple e che le operazioni più frequenti su di essa siano le seguenti:

o_1 `SELECT * FROM R ORDER BY A`, con frequenza $f_1 = 10$ operazioni nell'unità di tempo

o_2 `SELECT A, B, C FROM R ORDER BY A`, con frequenza $f_2 = 200$ operazioni nell'unità di tempo

Valutare le due seguenti alternative di memorizzazione, calcolando il costo complessivo (riportare la formula che indica il numero di accessi nell'unità di tempo e il relativo valore numerico):

(i) memorizzazione della relazione *R(A B C D E)* ordinata su A

costo unitario di o_1:

costo unitario di o_2:

costo complessivo:

(ii) memorizzazione delle proiezioni *R1(A B C)* e *R2(A D E)*, entrambe ordinate su A (supporre che il join possa essere eseguito con il metodo merge-join e che quindi il costo del join stesso sia trascurabile rispetto a quello delle due scansioni).

costo unitario di o_1:

costo unitario di o_2:

costo complessivo:

Domanda 2 (10%)
Considerare il documento XML qui sotto e definire uno schema XSD per il quale esso sia valido.

```
<?xml version="1.0" encoding="UTF-8"?>
<students>
   <student>
      <firstName> Paolo </firstName>
      <lastName> Neri </lastName>
      <id> 281283 </id>
      <courses>
            <course>
                        <name> Programmazione Orientata agli Oggetti </name>
                        <shortName> POO </shortName>
                        <record>
                              <grade> 28 </grade>
                              <date> 13/06/11 </date>
                        </record>
            </course>
            <course>
                        <name> Analisi e progettazione del software </name>
                        <shortName> APS </shortName>
            </course>
                  ...
      </courses>
   </student>
   <student>
      ...
   </student>
<students>
```

Domanda 3 (15%)
Con riferimento a documenti come quello mostrato nella Domanda 2 (supponendolo memorizzato nel file `esame.xml`), rispondere alle seguenti interrogazioni

1. In XPath, trovare gli studenti che hanno superato POO con voto superiore a 24

2. In XPath, trovare gli studenti che hanno una media superiore a 27

3. In XQuery, trovare gli studenti che hanno superato due esami con uno stesso voto

Domanda 4 (20%)
Una agenzia di viaggio on-line vende ai clienti pacchetti di viaggio proposti dai tour operator e desidera un sistema di supporto alle proprie attività che permetta di effettuare analisi sui pacchetti venduti in passato, al fine di comprendere sia le scelte dei clienti sulla base delle loro caratteristiche sia i profitti che derivano dalle vendite dei vari pacchetti. In particolare, si vuole realizzare un data mart con uno schema dimensionale che permetta di conoscere il numero di viaggi venduti, con il prezzo di vendita e il profitto per l'agenzia, rispetto a varie dimensioni, quali il tour operator, il cliente, con le sue caratteristiche fondamentali (fascia di età, professione, città e regione di partenza; si noti che alcune possono cambiare nel tempo), la destinazione (regione, località e sue caratteristiche), il tipo di sistemazione (categoria di albergo, appartamento, etc.).
Mostrare un possibile schema dimensionale, commentando brevemente (nello spazio disponibile) le eventuali scelte significative e chiarendo bene quale sia la grana dei fatti e il significato di ciascuna dimensione.

Domanda 5 (10%)
Considerare il seguente scenario in cui due client diversi inviano richieste ad un gestore del controllo di concorrenza. Ciascun client può inviare una richiesta solo dopo che è stata eseguita o rifiutata la precedente (se invece una richiesta viene bloccata da un lock, allora il client rimane inattivo fino alla concessione o allo scadere del timeout). Si supponga che, in caso di stallo, abortisca la transazione che ha avanzato la richiesta per prima. In caso di abort, si supponga che il client rilanci immediatamente la stessa transazione.

client 1	client 2
read(x)	
	read(x)
x = x + 10	
write(x)	
	x = x + 20
	write(x)
commit	
	commit

Considerare uno scheduler che utilizzi il controllo di concorrenza basato su multiversioni e livelli di isolamento SERIALIZABLE e READ COMMITTED. Assumiamo che (come avviene di solito) multiversioni preveda

- SERIALIZABLE: le letture fanno riferimento allo stato della base di dati all'inizio della transazione e le scritture di una transazione T sono soggette ad un lock a due fasi stretto (solo per le scritture) e sono ammesse solo se il dato non è stato modificato, dopo l'inizio di T, da altre transazioni.
- READ COMMITTED: le letture fanno riferimento allo stato della base di dati all'inizio della specifica lettura e le scritture sono soggette ad un lock a due fasi stretto (solo per le scritture).

Mostrare il comportamento dello scheduler nei due casi seguenti, supponendo che il valore iniziale dell'oggetto x sia 200. Indicare le operazioni che vengono eseguite nell'ordine con, per ciascuna, il valore che viene letto o scritto. In conclusione, per ciascun caso, dire se si verificano o meno anomalie.

READ COMMITTED	SERIALIZABLE

Domanda 6 (10%)
Considerare il seguente scenario in cui due client diversi inviano richieste ad un gestore del controllo di concorrenza. Ciascun client può inviare una richiesta solo dopo che è stata eseguita o rifiutata la precedente (se invece una richiesta viene bloccata da un lock, allora il client rimane inattivo fino alla concessione o allo scadere del timeout). Si supponga che, in caso di stallo, abortisca la transazione che ha avanzato la richiesta per prima. In caso di abort, si supponga che il client rilanci immediatamente la stessa transazione.

client 1	client 2
read(x)	
x = x + 10	
write(x)	
	read(x)
	x = x + 20
	write(x)
commit	
	commit

Considerare uno scheduler che utilizzi il controllo di concorrenza basato su 2PL e livelli di isolamento SERIALIZABLE e READ COMMITTED. Assumiamo che (come avviene di solito) 2PL preveda

- SERIALIZABLE: lock a due fasi stretto, con lock condivisi per letture e esclusivi per scritture.
- READ COMMITTED: lock condivisi per la lettura senza 2PL (possono essere rilasciati prima della acquisizione di altri lock) ed esclusivi per la scrittura con 2PL stretto (mantenuti fino a commit o abort).

Mostrare il comportamento dello scheduler nei due casi seguenti, supponendo che il valore iniziale dell'oggetto x sia 200. Indicare le operazioni che vengono eseguite nell'ordine con, per ciascuna, il valore che viene letto o scritto. In conclusione, per ciascun caso, dire se si verificano o meno anomalie.

READ COMMITTED	SERIALIZABLE

Domanda 7 (20%)
Considerare le relazioni sotto schematizzate, su cui si deve effettuare un hash join (sulla base del campo numerico). Si supponga che il fattore di blocco sia pari a 2 per entrambe le relazioni (e quindi che esse, come mostrato dalle divisioni fra le celle, occupino rispettivamente 7 e 10 blocchi) e che i buffer disponibili siano 4. Come noto, l'hash join (in questo caso, come in molti altri) si può eseguire in due passate, suddividendo prima entrambe le relazioni (per mezzo di una stessa funzione hash) in un numero di porzioni (dette *bucket*) il cui quadrato sia superiore al numero di blocchi della più piccola delle due relazioni, (in questo caso quindi 3) e poi confrontando i record nei bucket omologhi.

Mostrare (1) i bucket che si otterrebbero per le due relazioni (con una funzione hash che calcola il resto della divisione per 3, che nell'esempio corrisponde sempre al valore dell'ultima cifra); (2) il contenuto dei buffer all'inizio della seconda fase; (3) il contenuto dei buffer dopo l'esecuzione di sette chiamate al metodo `next()` sullo scan che implementa l'hash join; (4) i record prodotti dalle prime sette chiamate di `next()`.

Bucket per R1

Buffer all'inizio della seconda fase

Record prodotti dalle prime sette chiamate di next()

R1

A	901
B	931
C	330
D	660
E	362
F	362
G	900
H	390
I	390
L	900
M	362
N	391
O	392
P	601

Bucket per R2

Buffer dopo 7 chiamate di next()

R2

901	...
931	...
932	...
902	...
330	...
900	...
660	...
301	...
302	...
630	...
631	...
360	...
361	...
362	...
390	...
391	...
392	...
600	...
601	...
602	...

Basi di dati II
Esame — 25 settembre 2012 — Compito B

Rispondere su questo fascicolo. Tempo a disposizione: due ore e trenta minuti.

Cognome _____ **Nome** _____ **Matricola** _____ **Ordin.** _____

Domanda 1 (15%)
Si consideri una relazione *R(A B C D E)*, in cui gli attributi hanno tutti la stessa dimensione $L = 4$ Byte, molto più piccola della dimensione del blocco pari a P = 4 KByte. Si supponga che la relazione contenga $R = 1.000.000$ ennuple e che le operazioni più frequenti su di essa siano le seguenti:

o_1 `SELECT * FROM R ORDER BY A`, con frequenza $f_1 = 10$ operazioni nell'unità di tempo

o_2 `SELECT A, B, C FROM R ORDER BY A`, con frequenza $f_2 = 200$ operazioni nell'unità di tempo

Valutare le due seguenti alternative di memorizzazione, calcolando il costo complessivo (riportare la formula che indica il numero di accessi nell'unità di tempo e il relativo valore numerico):

(i) memorizzazione della relazione *R(A B C D E)* ordinata su A

costo unitario di o_1:

costo unitario di o_2:

costo complessivo:

(ii) memorizzazione delle proiezioni *R1(A B C)* e *R2(A D E)*, entrambe ordinate su A (supporre che il join possa essere eseguito con il metodo merge-join e che quindi il costo del join stesso sia trascurabile rispetto a quello delle due scansioni).

costo unitario di o_1:

costo unitario di o_2:

costo complessivo:

Domanda 2 (10%)
Considerare il documento XML qui sotto e definire uno schema XSD per il quale esso sia valido.

```
<?xml version="1.0" encoding="UTF-8"?>
<students>
   <student>
      <firstName> Paolo </firstName>
      <lastName> Neri </lastName>
      <id> 281283 </id>
      <courses>
            <course>
                        <name> Programmazione Orientata agli Oggetti </name>
                        <shortName> POO </shortName>
                        <record>
                              <grade> 28 </grade>
                              <date> 13/06/11 </date>
                        </record>
            </course>
            <course>
                        <name> Analisi e progettazione del software </name>
                        <shortName> APS </shortName>
            </course>
                  ...
      </courses>
   </student>
   <student>
      ...
   </student>
<students>
```

Domanda 3 (15%)
Con riferimento a documenti come quello mostrato nella Domanda 2 (supponendolo memorizzato nel file `esame.xml`), rispondere alle seguenti interrogazioni

1. In XPath, trovare gli studenti che hanno superato POO con voto superiore a 24

2. In XPath, trovare gli studenti che hanno una media superiore a 27

3. In XQuery, trovare gli studenti che hanno superato due esami con uno stesso voto

Domanda 4 (20%)
Una agenzia di viaggio on-line vende ai clienti pacchetti di viaggio proposti dai tour operator e desidera un sistema di supporto alle proprie attività che permetta di effettuare analisi sui pacchetti venduti in passato, al fine di comprendere sia le scelte dei clienti sulla base delle loro caratteristiche sia i profitti che derivano dalle vendite dei vari pacchetti. In particolare, si vuole realizzare un data mart con uno schema dimensionale che permetta di conoscere il numero di viaggi venduti, con il prezzo di vendita e il profitto per l'agenzia, rispetto a varie dimensioni, quali il tour operator, il cliente, con le sue caratteristiche fondamentali (fascia di età, professione, città e regione di partenza; si noti che alcune possono cambiare nel tempo), la destinazione (regione, località e sue caratteristiche), il tipo di sistemazione (categoria di albergo, appartamento, etc.).
Mostrare un possibile schema dimensionale, commentando brevemente (nello spazio disponibile) le eventuali scelte significative e chiarendo bene quale sia la grana dei fatti e il significato di ciascuna dimensione.

Domanda 5 (10%)
Considerare il seguente scenario in cui due client diversi inviano richieste ad un gestore del controllo di concorrenza. Ciascun client può inviare una richiesta solo dopo che è stata eseguita o rifiutata la precedente (se invece una richiesta viene bloccata da un lock, allora il client rimane inattivo fino alla concessione o allo scadere del timeout). Si supponga che, in caso di stallo, abortisca la transazione che ha avanzato la richiesta per prima. In caso di abort, si supponga che il client rilanci immediatamente la stessa transazione.

client 1	client 2
read(x)	
	read(x)
x = x + 10	
write(x)	
	x = x + 20
	write(x)
commit	
	commit

Considerare uno scheduler che utilizzi il controllo di concorrenza basato su 2PL e livelli di isolamento SERIALIZABLE e READ COMMITTED. Assumiamo che (come avviene di solito) 2PL preveda

- SERIALIZABLE: lock a due fasi stretto, con lock condivisi per letture e esclusivi per scritture.
- READ COMMITTED: lock condivisi per la lettura senza 2PL (possono essere rilasciati prima della acquisizione di altri lock) ed esclusivi per la scrittura con 2PL stretto (mantenuti fino a commit o abort).

Mostrare il comportamento dello scheduler nei due casi seguenti, supponendo che il valore iniziale dell'oggetto x sia 200. Indicare le operazioni che vengono eseguite nell'ordine con, per ciascuna, il valore che viene letto o scritto. In conclusione, per ciascun caso, dire se si verificano o meno anomalie.

SERIALIZABLE	READ COMMITTED

Domanda 6 (10%)
Considerare il seguente scenario in cui due client diversi inviano richieste ad un gestore del controllo di concorrenza. Ciascun client può inviare una richiesta solo dopo che è stata eseguita o rifiutata la precedente (se invece una richiesta viene bloccata da un lock, allora il client rimane inattivo fino alla concessione o allo scadere del timeout). Si supponga che, in caso di stallo, abortisca la transazione che ha avanzato la richiesta per prima. In caso di abort, si supponga che il client rilanci immediatamente la stessa transazione.

client 1	client 2
read(x)	
x = x + 10	
write(x)	
	read(x)
	x = x + 20
	write(x)
commit	
	commit

Considerare uno scheduler che utilizzi il controllo di concorrenza basato su multiversioni e livelli di isolamento SERIALIZABLE e READ COMMITTED. Assumiamo che (come avviene di solito) multiversioni preveda

- SERIALIZABLE: le letture fanno riferimento allo stato della base di dati all'inizio della transazione e le scritture di una transazione T sono soggette ad un lock a due fasi stretto (solo per le scritture) e sono ammesse solo se il dato non è stato modificato, dopo l'inizio di T, da altre transazioni.
- READ COMMITTED: le letture fanno riferimento allo stato della base di dati all'inizio della specifica lettura e le scritture sono soggette ad un lock a due fasi stretto (solo per le scritture).

Mostrare il comportamento dello scheduler nei due casi seguenti, supponendo che il valore iniziale dell'oggetto x sia 200. Indicare le operazioni che vengono eseguite nell'ordine con, per ciascuna, il valore che viene letto o scritto. In conclusione, per ciascun caso, dire se si verificano o meno anomalie.

READ COMMITTED	SERIALIZABLE

Domanda 7 (20%)
Considerare le relazioni sotto schematizzate, su cui si deve effettuare un hash join (sulla base del campo numerico). Si supponga che il fattore di blocco sia pari a 2 per entrambe le relazioni (e quindi che esse, come mostrato dalle divisioni fra le celle, occupino rispettivamente 7 e 10 blocchi) e che i buffer disponibili siano 4. Come noto, l'hash join (in questo caso, come in molti altri) si può eseguire in due passate, suddividendo prima entrambe le relazioni (per mezzo di una stessa funzione hash) in un numero di porzioni (dette *bucket*) il cui quadrato sia superiore al numero di blocchi della più piccola delle due relazioni, (in questo caso quindi 3) e poi confrontando i record nei bucket omologhi.

Mostrare (1) i bucket che si otterrebbero per le due relazioni (con una funzione hash che calcola il resto della divisione per 3, che nell'esempio corrisponde sempre al valore dell'ultima cifra); (2) il contenuto dei buffer all'inizio della seconda fase; (3) il contenuto dei buffer dopo l'esecuzione di sette chiamate al metodo `next()` sullo scan che implementa l'hash join; (4) i record prodotti dalle prime sette chiamate di `next()`.

Bucket per R1

Buffer all'inizio della seconda fase

Record prodotti dalle prime sette chiamate di next()

R1

A	931
B	901
C	300
D	660
E	362
F	362
G	930
H	390
I	390
L	930
M	362
N	391
O	392
P	601

Bucket per R2

Buffer dopo 7 chiamate di next()

R2

931	...
901	...
902	...
932	...
300	...
930	...
660	...
301	...
302	...
630	...
631	...
360	...
361	...
362	...
390	...
391	...
392	...
600	...
601	...
602	...