

Basi di dati II

Prova parziale — 22 aprile 2013 — Compito A

Rispondere su questo fascicolo.

Tempo a disposizione: un'ora e quindici minuti.

Cognome _____ Nome _____ Matricola _____

Domanda 1 (25%)

Si consideri una base di dati su cui una applicazione effettua moltissimi inserimenti e aggiornamenti, con operazioni tutte molto semplici ma estremamente numerose. Considerare le due situazioni seguenti:

- A. concorrentemente alle operazioni sopra citate vengono eseguite molte interrogazioni (e nessun altro aggiornamento)
- B. concorrentemente alle operazioni sopra citate non viene eseguita nessun'altra operazione

Commentare brevemente per ciascuna delle due situazioni quali potrebbero essere i vantaggi e gli svantaggi (con riferimento alle prestazioni in termini di tempo di risposta sia per la normale operatività sia in caso di guasto e conseguente necessità di recovery) delle seguenti tre scelte per l'applicazione che esegue inserimenti e aggiornamenti:

- i. riunire le operazioni in transazioni di medie dimensioni (alcune decine di operazioni per ciascuna)
- ii. eseguire ciascuna operazione in una transazione separata
- iii. riunire tutte le operazioni in un'unica transazione

Nota bene: non è detto che esista una soluzione ideale, ciò che si deve fornire sono riflessioni critiche, che, sinteticamente, illustrino spunti interessanti.

	A	B
i.		
ii.		
iii.		

Domanda 2 (25%)

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, riutilizzandoli per successivi inserimenti (come avviene in SimpleDB);
- 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 $3/4 \times N = 150.000$ delle ennuple prima inserite (sulla base di una condizione verificabile durante la scansione)
3. dopo la conclusione e la chiusura della scansione precedente, inserimento di altre N ennuple

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 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:

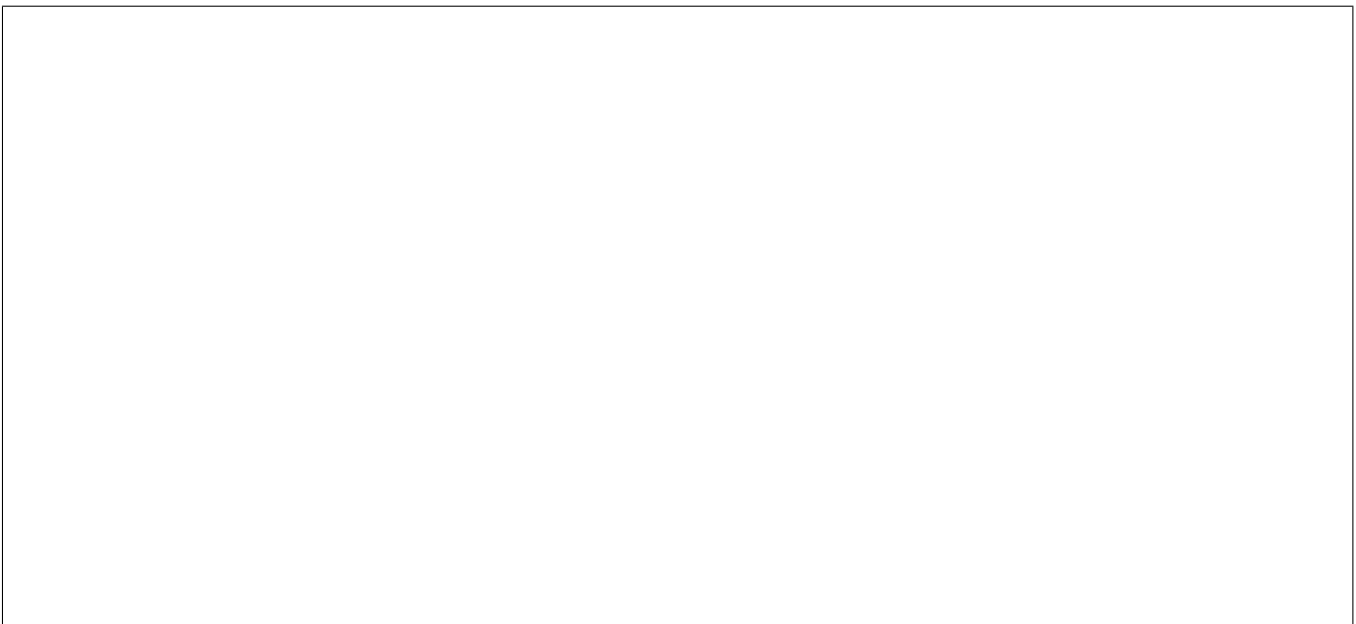
- numero di scritture di pagine di dati:

Domanda 3 (20%)

Si consideri un B+-tree con nodi intermedi che contengono tre chiavi e quattro puntatori e foglie con tre chiavi, in cui vengano inserite chiavi (a partire dall'albero vuoto) nel seguente ordine: 13, 20, 32, 41, 51, 61, 71, 85, 21, 22. Mostrare l'albero dopo l'inserimento di cinque chiavi, di otto chiavi e alla fine. (Si ricorda che nel B+-tree ogni chiave compare in una foglia)



Mostrare poi l'albero dopo l'eliminazione della chiave 51 dall'ultimo albero ottenuto in risposta alla domanda precedente.



Domanda 4 (15%)

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	read(x)
write(x)	x = x + 20
	write(x)
commit	commit

Considerare scheduler che utilizzino i due metodi di controllo di concorrenza basati su 2PL stretto e multiversioni e, in entrambi i casi, livello di isolamento **SERIALIZABLE**.

Mostrare il comportamento dei due scheduler, supponendo che il valore iniziale dell'oggetto x sia 400. 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.

2PL	multiversioni

Domanda 5 (15%)

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)	
$x = x + 10$	read(x)
write(x)	
commit	
	$x = x + 20$
	write(x)
	commit

Considerare scheduler che utilizzino i due metodi di controllo di concorrenza basati su 2PL stretto e multiversioni e, in entrambi i casi, livello di isolamento **SERIALIZABLE**.

Mostrare il comportamento dei due scheduler, supponendo che il valore iniziale dell'oggetto x sia 400. 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.

multiversioni	2PL

Basi di dati II

Prova parziale — 22 aprile 2013 — Compito B

Rispondere su questo fascicolo.

Tempo a disposizione: un'ora e quindici minuti.

Cognome _____ Nome _____ Matricola _____

Domanda 1 (25%)

Si consideri una base di dati su cui una applicazione effettua moltissimi inserimenti e aggiornamenti, con operazioni tutte molto semplici ma estremamente numerose. Considerare le due situazioni seguenti:

- A. concorrentemente alle operazioni sopra citate non viene eseguita nessun'altra operazione
- B. concorrentemente alle operazioni sopra citate vengono eseguite molte interrogazioni (e nessun altro aggiornamento)

Commentare brevemente per ciascuna delle due situazioni quali potrebbero essere i vantaggi e gli svantaggi (con riferimento alle prestazioni in termini di tempo di risposta sia per la normale operatività sia in caso di guasto e conseguente necessità di recovery) delle seguenti tre scelte per l'applicazione che esegue inserimenti e aggiornamenti:

- i. eseguire ciascuna operazione in una transazione separata
- ii. riunire le operazioni in transazioni di medie dimensioni (alcune decine di operazioni per ciascuna)
- iii. riunire tutte le operazioni in un'unica transazione

Nota bene: non è detto che esista una soluzione ideale, ciò che si deve fornire sono riflessioni critiche, che, sinteticamente, illustrino spunti interessanti.

	A	B
i.		
ii.		
iii.		

Domanda 2 (25%)

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, riutilizzandoli per successivi inserimenti (come avviene in SimpleDB);
- 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 $3/4 \times L = 150.000$ delle ennuple prima inserite (sulla base di una condizione verificabile durante la scansione)
3. dopo la conclusione e la chiusura della scansione precedente, inserimento di altre L ennuple

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 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:

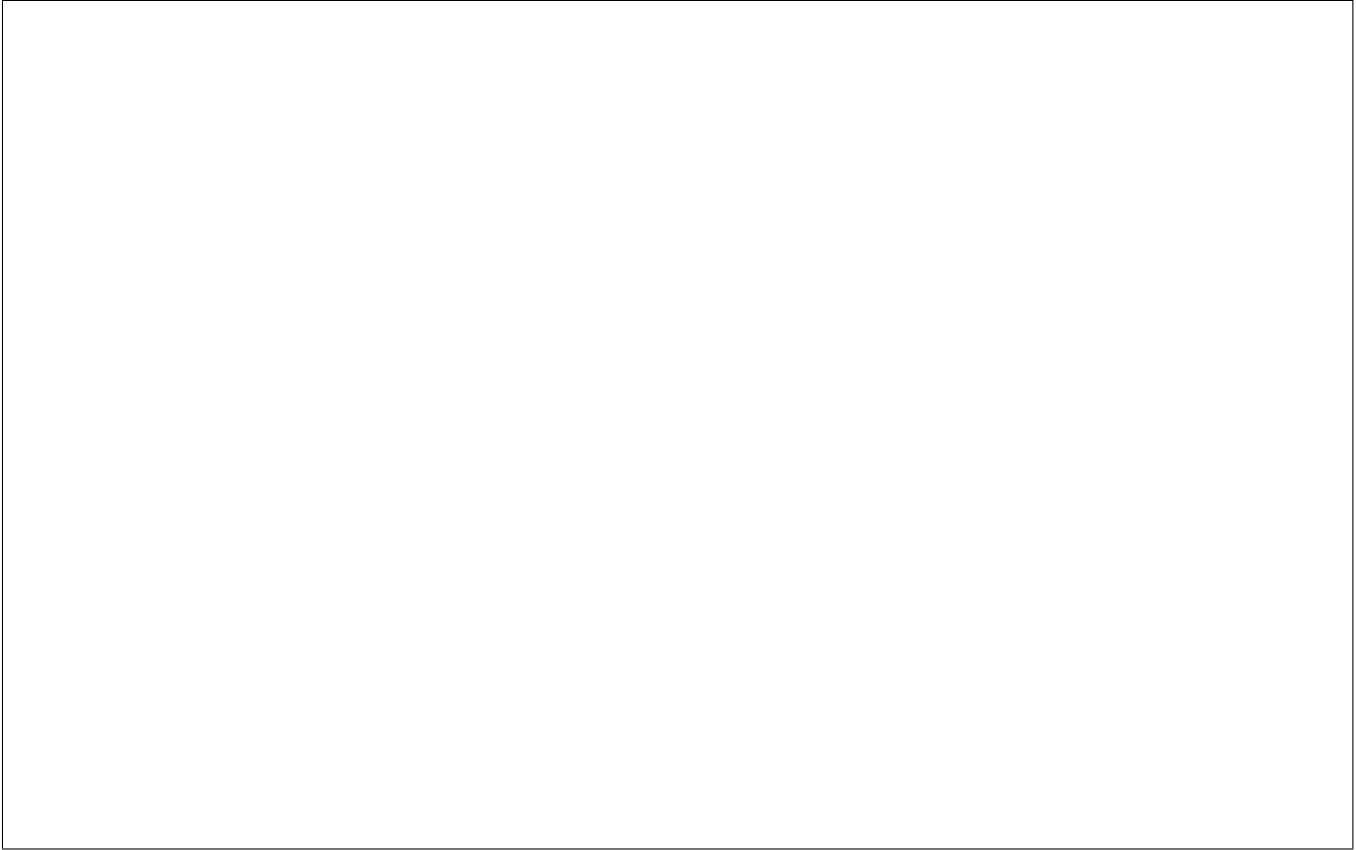
Come nel caso precedente, ma con riferimento ad un programma che utilizzi un'unica transazione per tutti gli inserimenti (e supponendo che non vi siano altre transazioni attive)

- numero di scritture di pagine di log:

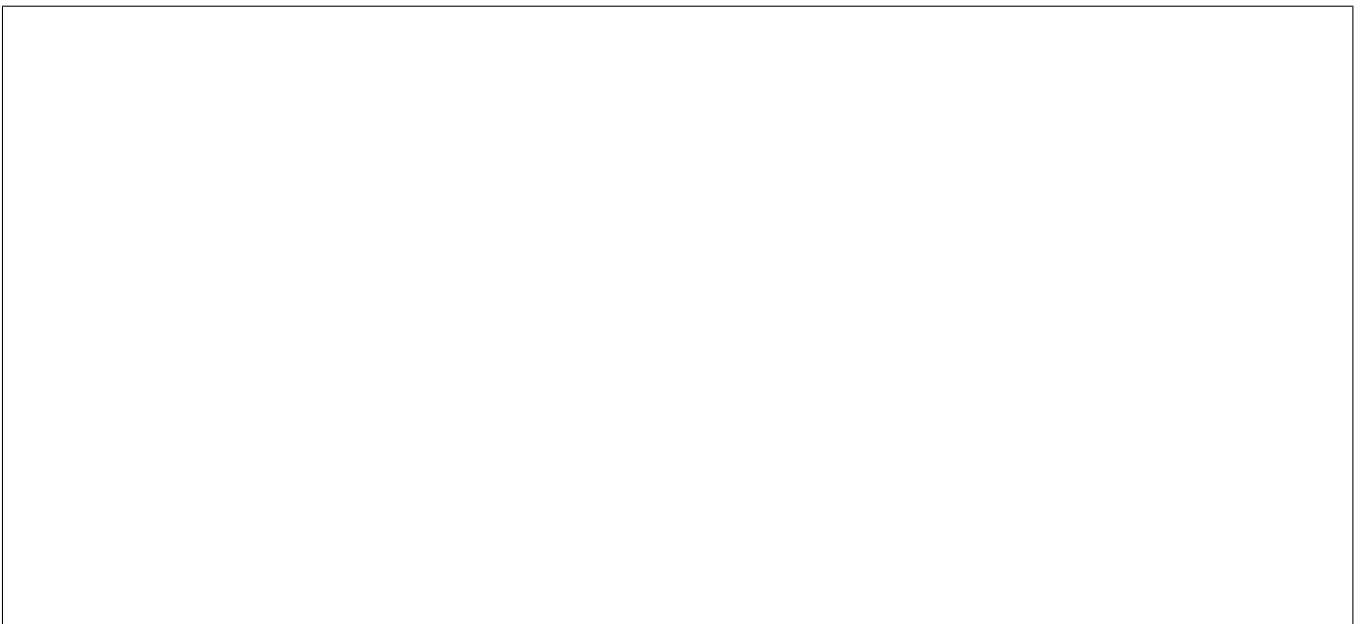
- numero di scritture di pagine di dati:

Domanda 3 (20%)

Si consideri un B+-tree con nodi intermedi che contengono tre chiavi e quattro puntatori e foglie con tre chiavi, in cui vengano inserite chiavi (a partire dall'albero vuoto) nel seguente ordine: 12, 20, 31, 43, 52, 63, 74, 84, 21, 22. Mostrare l'albero dopo l'inserimento di cinque chiavi, di otto chiavi e alla fine. (Si ricorda che nel B+-tree ogni chiave compare in una foglia)



Mostrare poi l'albero dopo l'eliminazione della chiave 52 dall'ultimo albero ottenuto in risposta alla domanda precedente.



Domanda 4 (15%)

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)	
$x = x + 10$	read(x)
write(x)	
commit	
	$x = x + 20$
	write(x)
	commit

Considerare scheduler che utilizzino i due metodi di controllo di concorrenza basati su 2PL stretto e multiversioni e, in entrambi i casi, livello di isolamento **SERIALIZABLE**.

Mostrare il comportamento dei due scheduler, supponendo che il valore iniziale dell'oggetto x sia 600. 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.

multiversioni	2PL

Domanda 5 (15%)

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 scheduler che utilizzino i due metodi di controllo di concorrenza basati su 2PL stretto e multiversioni e, in entrambi i casi, livello di isolamento **SERIALIZABLE**.

Mostrare il comportamento dei due scheduler, supponendo che il valore iniziale dell'oggetto x sia 300. 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.

2PL	multiversioni

Basi di dati II

Prova parziale — 22 aprile 2013 — Compito C

Rispondere su questo fascicolo.

Tempo a disposizione: un'ora e quindici minuti.

Cognome _____ Nome _____ Matricola _____

Domanda 1 (25%)

Si consideri una base di dati su cui una applicazione effettua moltissimi inserimenti e aggiornamenti, con operazioni tutte molto semplici ma estremamente numerose. Considerare le due situazioni seguenti:

- A. concorrentemente alle operazioni sopra citate vengono eseguite molte interrogazioni (e nessun altro aggiornamento)
- B. concorrentemente alle operazioni sopra citate non viene eseguita nessun'altra operazione

Commentare brevemente per ciascuna delle due situazioni quali potrebbero essere i vantaggi e gli svantaggi (con riferimento alle prestazioni in termini di tempo di risposta sia per la normale operatività sia in caso di guasto e conseguente necessità di recovery) delle seguenti tre scelte per l'applicazione che esegue inserimenti e aggiornamenti:

- i. eseguire ciascuna operazione in una transazione separata
- ii. riunire tutte le operazioni in un'unica transazione
- iii. riunire le operazioni in transazioni di medie dimensioni (alcune decine di operazioni per ciascuna)

Nota bene: non è detto che esista una soluzione ideale, ciò che si deve fornire sono riflessioni critiche, che, sinteticamente, illustrino spunti interessanti.

	A	B
i.		
ii.		
iii.		

Domanda 2 (25%)

Considerare una tabella T appena creata (e quindi vuota), con le seguenti ipotesi

- T 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 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, riutilizzandoli per successivi inserimenti (come avviene in SimpleDB);
- 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 $3/4 \times N = 150.000$ delle ennuple prima inserite (sulla base di una condizione verificabile durante la scansione)
3. dopo la conclusione e la chiusura della scansione precedente, inserimento di altre N ennuple

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 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:

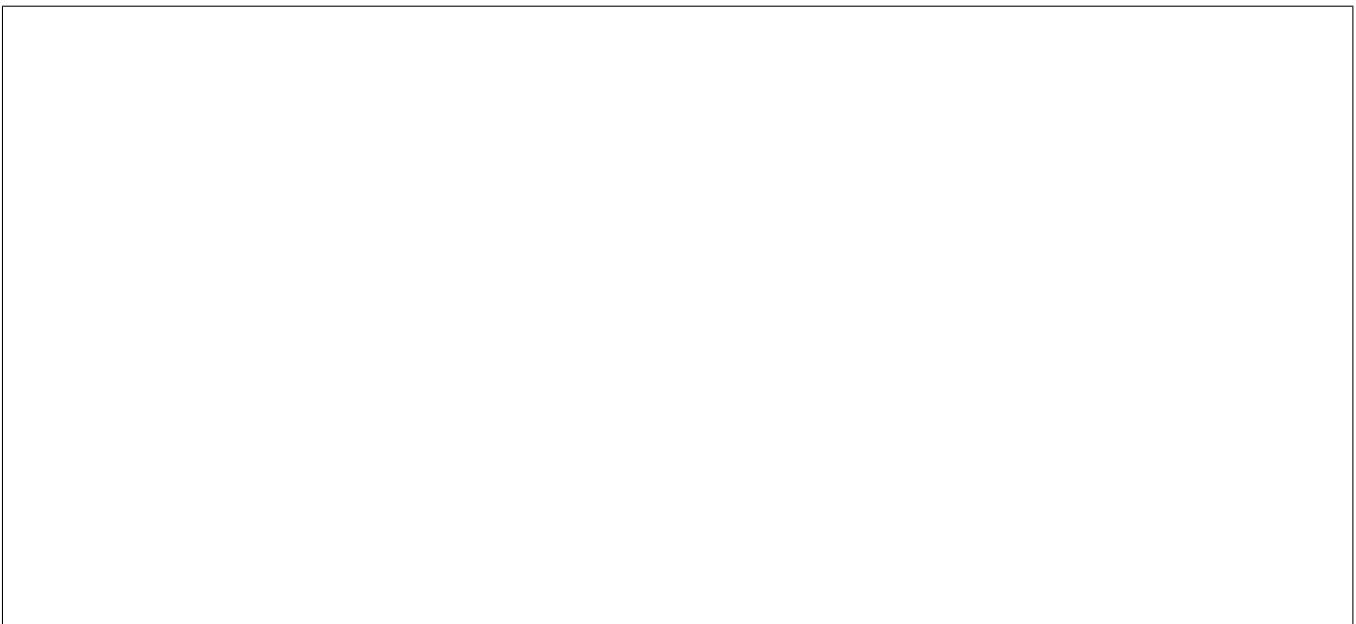
- numero di scritture di pagine di dati:

Domanda 3 (20%)

Si consideri un B+-tree con nodi intermedi che contengono tre chiavi e quattro puntatori e foglie con tre chiavi, in cui vengano inserite chiavi (a partire dall'albero vuoto) nel seguente ordine: 11, 20, 32, 44, 53, 62, 72, 83, 21, 22. Mostrare l'albero dopo l'inserimento di cinque chiavi, di otto chiavi e alla fine. (Si ricorda che nel B+-tree ogni chiave compare in una foglia)



Mostrare poi l'albero dopo l'eliminazione della chiave 53 dall'ultimo albero ottenuto in risposta alla domanda precedente.



Domanda 4 (15%)

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	read(x)
write(x)	x = x + 20
	write(x)
commit	commit

Considerare scheduler che utilizzino i due metodi di controllo di concorrenza basati su 2PL stretto e multiversioni e, in entrambi i casi, livello di isolamento **SERIALIZABLE**.

Mostrare il comportamento dei due scheduler, supponendo che il valore iniziale dell'oggetto x sia 500. 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.

2PL	multiversioni

Domanda 5 (15%)

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)	
$x = x + 10$	read(x)
write(x)	
commit	
	$x = x + 20$
	write(x)
	commit

Considerare scheduler che utilizzino i due metodi di controllo di concorrenza basati su 2PL stretto e multiversioni e, in entrambi i casi, livello di isolamento **SERIALIZABLE**.

Mostrare il comportamento dei due scheduler, supponendo che il valore iniziale dell'oggetto x sia 400. 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.

multiversioni	2PL

Basi di dati II

Prova parziale — 22 aprile 2013 — Compito D

Rispondere su questo fascicolo.

Tempo a disposizione: un'ora e quindici minuti.

Cognome _____ Nome _____ Matricola _____

Domanda 1 (25%)

Si consideri una base di dati su cui una applicazione effettua moltissimi inserimenti e aggiornamenti, con operazioni tutte molto semplici ma estremamente numerose. Considerare le due situazioni seguenti:

- A. concorrentemente alle operazioni sopra citate non viene eseguita nessun'altra operazione
- B. concorrentemente alle operazioni sopra citate vengono eseguite molte interrogazioni (e nessun altro aggiornamento)

Commentare brevemente per ciascuna delle due situazioni quali potrebbero essere i vantaggi e gli svantaggi (con riferimento alle prestazioni in termini di tempo di risposta sia per la normale operatività sia in caso di guasto e conseguente necessità di recovery) delle seguenti tre scelte per l'applicazione che esegue inserimenti e aggiornamenti:

- i. riunire tutte le operazioni in un'unica transazione
- ii. eseguire ciascuna operazione in una transazione separata
- iii. riunire le operazioni in transazioni di medie dimensioni (alcune decine di operazioni per ciascuna)

Nota bene: non è detto che esista una soluzione ideale, ciò che si deve fornire sono riflessioni critiche, che, sinteticamente, illustrino spunti interessanti.

	A	B
i.		
ii.		
iii.		

Domanda 2 (25%)

Considerare una tabella R appena creata (e quindi vuota), con le seguenti ipotesi

- R 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 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, riutilizzandoli per successivi inserimenti (come avviene in SimpleDB);
- 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 $3/4 \times L = 150.000$ delle ennuple prima inserite (sulla base di una condizione verificabile durante la scansione)
3. dopo la conclusione e la chiusura della scansione precedente, inserimento di altre L ennuple

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 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:

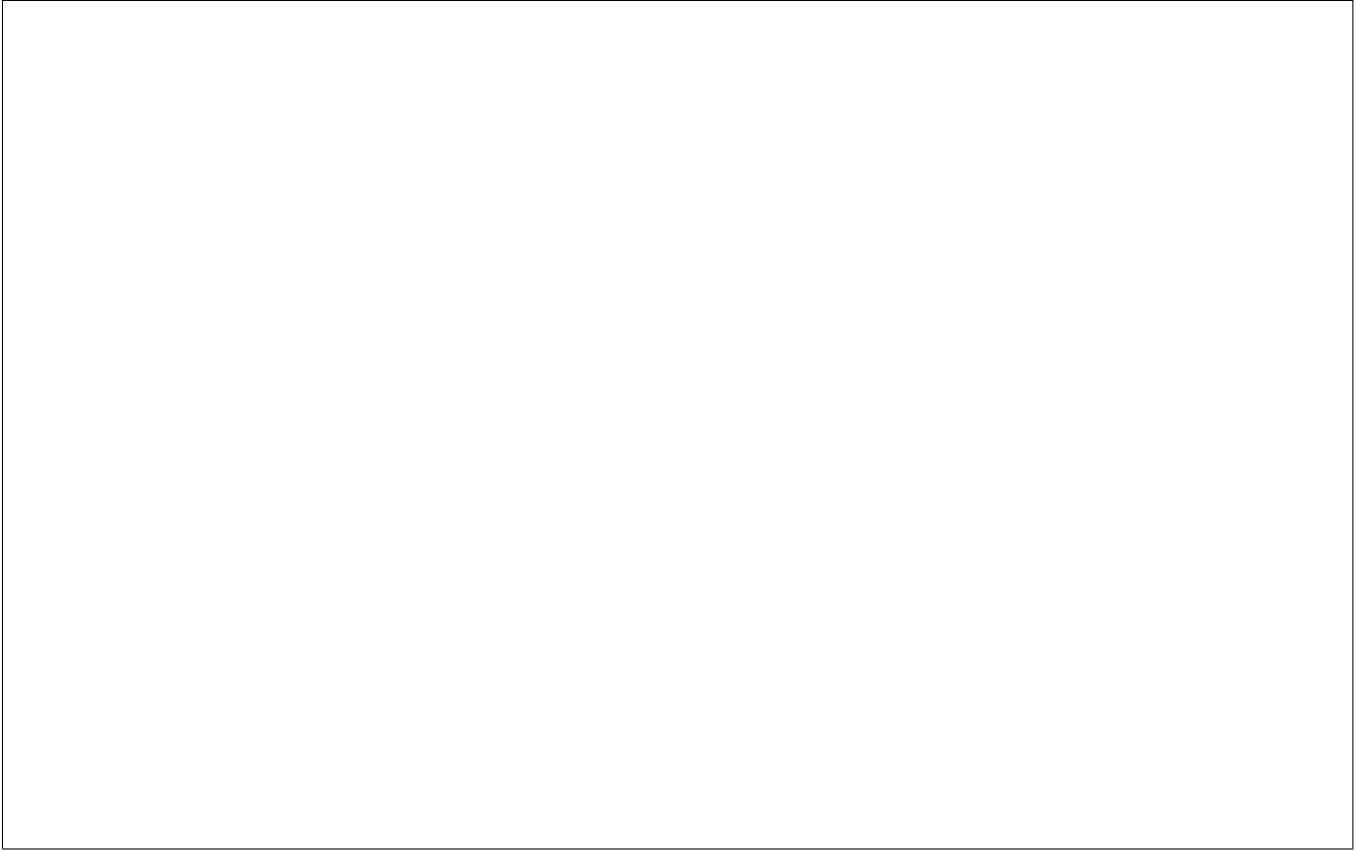
Come nel caso precedente, ma con riferimento ad un programma che utilizzi un'unica transazione per tutti gli inserimenti (e supponendo che non vi siano altre transazioni attive)

- numero di scritture di pagine di log:

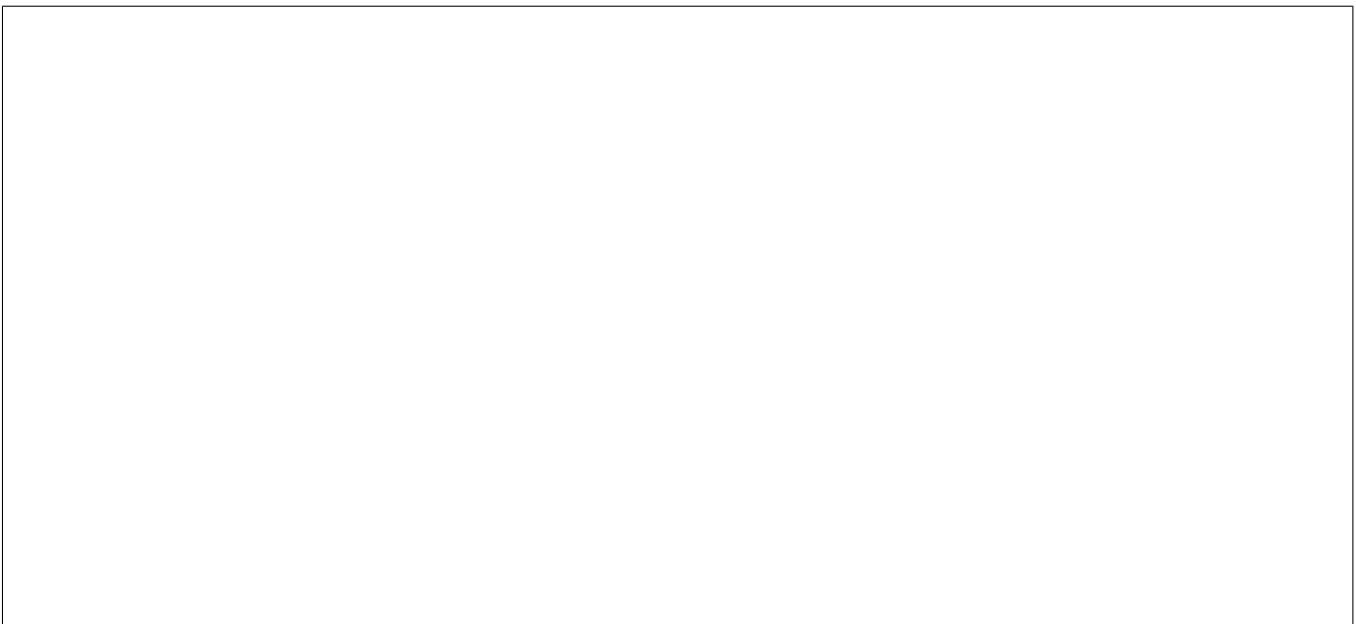
- numero di scritture di pagine di dati:

Domanda 3 (20%)

Si consideri un B+-tree con nodi intermedi che contengono tre chiavi e quattro puntatori e foglie con tre chiavi, in cui vengano inserite chiavi (a partire dall'albero vuoto) nel seguente ordine: 12, 20, 34, 43, 52, 64, 73, 83, 21, 22. Mostrare l'albero dopo l'inserimento di cinque chiavi, di otto chiavi e alla fine. (Si ricorda che nel B+-tree ogni chiave compare in una foglia)



Mostrare poi l'albero dopo l'eliminazione della chiave 52 dall'ultimo albero ottenuto in risposta alla domanda precedente.



Domanda 4 (15%)

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)	
$x = x + 10$	read(x)
write(x)	
commit	$x = x + 20$
	write(x)
	commit

Considerare scheduler che utilizzino i due metodi di controllo di concorrenza basati su 2PL stretto e multiversioni e, in entrambi i casi, livello di isolamento **SERIALIZABLE**.

Mostrare il comportamento dei due scheduler, 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.

multiversioni	2PL

Domanda 5 (15%)

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 scheduler che utilizzino i due metodi di controllo di concorrenza basati su 2PL stretto e multiversioni e, in entrambi i casi, livello di isolamento SERIALIZABLE.

Mostrare il comportamento dei due scheduler, supponendo che il valore iniziale dell'oggetto x sia 500. 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.

2PL	multiversioni

Basi di dati II

Prova parziale — 22 aprile 2013 — Compito A

Cenni sulle soluzioni

(per il compito A, gli altri sono simili, le varianti del testo sono in rosso)

Rispondere su questo fascicolo.

Tempo a disposizione: un'ora e quindici minuti.

Cognome _____ Nome _____ Matricola _____

Domanda 1 (25%)

Si consideri una base di dati su cui una applicazione effettua moltissimi inserimenti e aggiornamenti, con operazioni tutte molto semplici ma estremamente numerose. Considerare le due situazioni seguenti:

- A. concorrentemente alle operazioni sopra citate **vengono eseguite molte interrogazioni (e nessun altro aggiornamento)**
- B. concorrentemente alle operazioni sopra citate **non viene eseguita nessun'altra operazione**

Commentare brevemente per ciascuna delle due situazioni quali potrebbero essere i vantaggi e gli svantaggi (con riferimento alle prestazioni in termini di tempo di risposta sia per la normale operatività sia in caso di guasto e conseguente necessità di recovery) delle seguenti tre scelte per l'applicazione che esegue inserimenti e aggiornamenti:

- i. **riunire le operazioni in transazioni di medie dimensioni (alcune decine di operazioni per ciascuna)**
- ii. **eseguire ciascuna operazione in una transazione separata**
- iii. **riunire tutte le operazioni in un'unica transazione**

Nota bene: non è detto che esista una soluzione ideale, ciò che si deve fornire sono riflessioni critiche, che, sinteticamente, illustrino spunti interessanti.

Commenti sulle soluzioni

Considerazioni generali, da combinare poi per rispondere alle domande:

- rispetto alla concorrenza (e quindi nel caso in cui ci siano di esecuzione le interrogazioni, che sono molte), la singola transazione molto probabilmente penalizza, perché blocca e/o viene bloccata; la soluzione estrema opposta è molto più efficiente, quella intermedia è probabilmente pure efficiente (se la base di dati è grande, bloccherà solo una piccola parte dei dati)
- più sono le transazioni, maggiore è l'onere per la gestione della affidabilità (con inizio e fine di transazioni e scritture, anche forzate, sul log)
- in caso di guasto, la soluzione con transazione unica porta all'annullamento di tutte le azioni svolte (il commit è alla fine e quindi un eventuale guasto sarà prima di esso), con la necessità quindi di ripetere il lavoro

Domanda 2 (25%)

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, riutilizzandoli per successivi inserimenti (come avviene in SimpleDB);
- 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 $3/4 \times N = 150.000$ delle ennuple prima inserite (sulla base di una condizione verificabile durante la scansione)
3. dopo la conclusione e la chiusura della scansione precedente, inserimento di altre N ennuple

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):

$$5/4 \times N/f = 2500$$

(lo spazio libero viene riutilizzato)

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

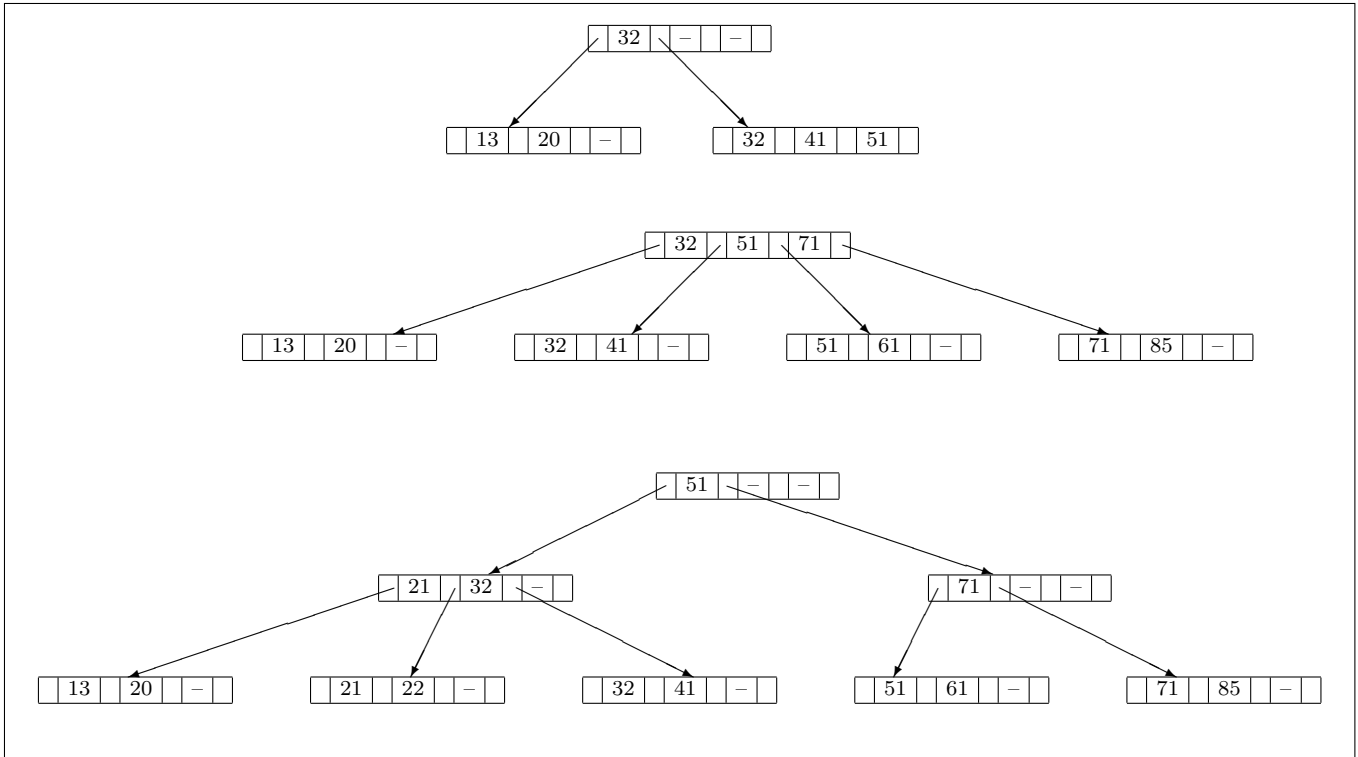
- numero di scritture di pagine di log:
almeno $N = 200.000$, una per transazione
- numero di scritture di pagine di dati:
con una strategia undo-only, una per transazione, $N = 200.000$; altrimenti, probabilmente di meno, anche solo $N/f = 2000$, una per blocco

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)

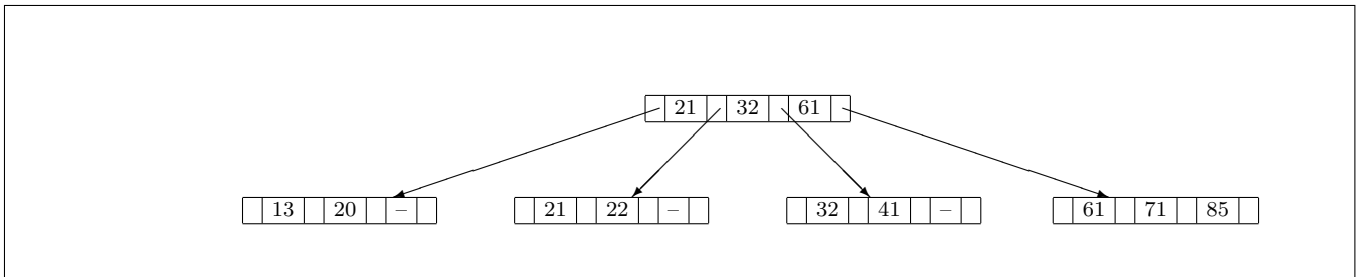
- numero di scritture di pagine di log:
dovrebbe essere possibile scrivere tutto il log al momento del commit e quindi (poiché i record sono grandi il triplo di quelli del file): $3 \times N/f = 6000$
- numero di scritture di pagine di dati:
in ogni caso si dovrebbe avere $N/f = 2000$, una per blocco, o poco più

Domanda 3 (20%)

Si consideri un B+-tree con nodi intermedi che contengono tre chiavi e quattro puntatori e foglie con tre chiavi, in cui vengano inserite chiavi (a partire dall'albero vuoto) nel seguente ordine: 13, 20, 32, 41, 51, 61, 71, 85, 21, 22. Mostrare l'albero dopo l'inserimento di cinque chiavi, di otto chiavi e alla fine. (Si ricorda che nel B+-tree ogni chiave compare in una foglia)



Mostrare poi l'albero dopo l'eliminazione della chiave 51 dall'ultimo albero ottenuto in risposta alla domanda precedente.



Domanda 4 (15%)

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 scheduler che utilizzino i due metodi di controllo di concorrenza basati su 2PL stretto e multiversioni e, in entrambi i casi, livello di isolamento SERIALIZABLE.

Mostrare il comportamento dei due scheduler, supponendo che il valore iniziale dell'oggetto x sia 400. 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.

2PL				multiversioni			
client 1		client 2		client 1		client 2	
read(x)	legge 400	read(x)	legge 400	read(x)	legge 400	read(x)	legge 400
x = x + 10 xlock(x)	x vale 410 bloccata	x = x + 20 xlock(x)	x vale 420 bloccata	x = x + 10 write(x)	x vale 410 scrive 410	x = x + 20 xlock(x)	x vale 420 bloccata
abort slock(x)	bloccata	stallo		commit		abort read(x)	x vale 410 legge 410
read(x)	legge 420	write(x)	scrive 420	x = x + 20 write(x)	x vale 430 scrive 430	x = x + 20 write(x)	x vale 430 scrive 430
x = x + 10 write(x)	x vale 430 scrive 430	commit					
commit							
non c'è anomalia				non c'è anomalia; l'abort è dovuto al fatto che x è stato modificato dopo l'avvio della transazione che sta scrivendo			

Domanda 5 (15%)

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) commit	x = x + 20 write(x) commit

Considerare scheduler che utilizzino i due metodi di controllo di concorrenza basati su 2PL stretto e multiversioni e, in entrambi i casi, livello di isolamento SERIALIZABLE.

Mostrare il comportamento dei due scheduler, supponendo che il valore iniziale dell'oggetto x sia 400. 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.

multiversioni				2PL			
client 1		client 2		client 1		client 2	
read(x)	legge 400	read(x)	legge 400	read(x)	legge 400	read(x)	legge 400
x = x + 10	x vale 410			x = x + 10	x vale 410	x = x + 20	x vale 420
write(x)	scrive 410			xlock(x)	bloccata	xlock(x)	bloccata
commit						stallo	
		x = x + 20	x vale 420	abort			
		abort		slock(x)	bloccata	write(x)	scrive 420
		read(x)	legge 410			commit	
		x = x + 20	x vale 430	read(x)	legge 420		
		write(x)	scrive 430	x = x + 10	x vale 430		
				write(x)	scrive 430		
				commit			
non c'è anomalia; l'abort è dovuto al fatto che x è stato modificato dopo l'avvio della transazione che sta scrivendo				non c'è anomalia			