Tecnologie e architetture per la gestione dei dati — 5 luglio 2024

Tempo a disposizione: due ore.

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

Tecnologie e architetture per la gestione dei dati — 5 luglio 2024

Domanda 1 (15%)
Si considerino un sistema con blocchi di dimensione $B = 8000$ byte e una relazione *R(A,C,....)* di cardinalità pari circa a $N = 10.000.000$, con ennuple di $l = 80$ byte e campo A chiave di tipo stringa (ad esempio, il codice fiscale) e campo C di tipo intero anch'esso chiave (ad esempio, un numero di matricola) — quindi la relazione ha due chiavi, ognuna delle quali, da sola, identifica univocamente le ennuple. Supporre che il sistema offra

- strutture primarie di tre tipi: (a) disordinate, (b) ordinate rispetto ad un campo su cui è definito un indice, (c) hash
- indici di tipo B-tree (anche più di uno sulla stessa relazione, primari o secondari)

Considerare un carico applicativo con le seguenti operazioni **tutte di lettura**

1. ricerca di una ennupla sulla base del valore parziale (una sottostringa iniziale) della chiave A, con frequenza giornaliera $f_1 = 100.000$; supporre che il valore parziale sia abbastanza selettivo e porti alla identificazione, in media, di $n = 10$ ennuple;
2. ricerca di una ennupla sulla base del valore (completo) della chiave C, con frequenza giornaliera $f_2 = 100.000$;
3. scansione dell'intera relazione, ordinata sulla base del valore di A, con frequenza giornaliera $f_3 = 1$;
4. scansione dell'intera relazione, ordinata sulla base del valore di C, con frequenza giornaliera $f_4 = 1$;

Progettare l'organizzazione fisica della relazione, (i) scegliendo la struttura primaria fra le varie possibilità e (ii) individuando gli eventuali indici. Ragionare in termini di numero di accessi a memoria secondaria, assumendo che (1) gli indici primari abbiano profondità $p_1 = 3$ e gli indici secondari profondità $p_2 = 4$, (2) il buffer disponibile abbia (2a) dimensione minore del numero di blocchi del file e maggiore della radice quadrata di tale numero e (2b) permetta di mantenere stabilmente in memoria due livelli di indice, sia per i primari sia per i secondari; (3) l'hash, per gli accessi puntuali, abbia costo unitario. Proporre almeno due alternative (quelle che intuitivamente si ritengono migliori) e valutarne il costo. Rispondere negli spazi sottostanti, in forma sia simbolica sia numerica.

	Alternativa 1	Alternativa 2	Alternativa 3 (eventuale)
Descrizione struttura			
Costo operazione 1			
Costo operazione 2			
Costo operazione 3			
Costo operazione 4			
Costo totale			

Tecnologie e architetture per la gestione dei dati — 5 luglio 2024

Domanda 2 (15%)
Considerare ancora il caso illustrato nella domanda precedente, ma con riferimento ad una realtà con le seguenti frequenze per le operazioni:

1. $f_1 = 10.000$
2. $f_2 = 1.000.000$
3. $f_3 = 1$
4. $f_4 = 10$

Considerare ancora almeno due alternative (quelle che intuitivamente si ritengono migliori in questo caso).

	Alternativa 1	Alternativa 2	Alternativa 3 (eventuale)
Descrizione struttura			
Costo operazione 1			
Costo operazione 2			
Costo operazione 3			
Costo operazione 4			
Costo totale			

Tecnologie e architetture per la gestione dei dati — 5 luglio 2024

Domanda 3 (30%)
Come noto, in molti sistemi (ad esempio in PostgreSQL) è possibile ordinare fisicamente le relazioni, senza che però l'ordinamento venga mantenuto. Allo scopo, l'ordinamento viene rieseguito periodicamente, su dati che sono in parte ordinati e in parte no. Una possibile strategia seguita (molto comune) è la seguente:

- le eliminazioni vengono realizzate semplicemente "marcando" i record e lasciando quindi lo spazio inutilizzato
- gli inserimenti vengono effettuati in coda, nell'ordine in cui vengono richiesto
- le modifiche vengono trattate ciascuna come un'eliminazione e un inserimento

Di conseguenza, si possono presentare situazioni come quella mostrata in sotto, in cui la relazione è in parte ordinata e in parte disordinata (l'asterisco "*" indica i record cancellati). Si noti che la relazione presenta molti blocchi ordinati (dodici nell'esempio) e alcuni disordinati (tre).

Supponendo di avere a disposizione **sei** pagine buffer, descrivere brevemente nel riquadro l'algoritmo da utilizzare per riordinare la relazione, mostrando anche il contenuto dei buffer in occasione del primo caricamento e quello alla fine dell'ordinamento. Rispondere anche alle altre domande che vengono poste.

111	...
120	...
141	...
181	...
200	...
221	...
232	...
251	...
345	...
*	
443	...
501	...
502	...
525	...
686	...
*	
735	...
774	...
783	...
801	...
805	...
839	...
842	...
900	...

535	...
171	...
484	...
838	...
262	...
646	...

- Descrivere sinteticamente l'algoritmo (bastano poche righe informali, non serve pseudocodice) demira

- Con riferimento all'esempio e all'algoritmo proposto, quante letture fisiche di blocchi sono necessarie? [ ] E quante scritture? [ ]

- Mostrare il contenuto delle pagine di buffer al primo caricamento e il contenuto delle stesse alla fine dell'esecuzione dell'algoritmo

Primo caricamento delle pagine del buffer	Contenuto finale delle pagine del buffer

Tecnologie e architetture per la gestione dei dati — 5 luglio 2024

Domanda 4 (20%)
Considerare un sistema che utilizzi blocchi di lunghezza $D = 4$ KB (approssimabili a 4000 byte) e una tabella R con una struttura fisica heap con record a lunghezza fissa che occupano $L = 20$ byte ciascuno, in cui vengono inserite $M = 100.000$ ennuple, con valori della chiave tutti diversi fra loro e da quelli già nella relazione (quindi il sistema verifica il soddisfacimento del vincolo di chiave e ammette tutte le operazioni).
Rispondere alle domande seguenti, indicando formule e valori numerici:

Indicare il numero di scritture in memoria secondaria necessarie per realizzare i 100.000 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 della relazione, nei tre casi seguenti:
  - strategia undo-redo senza vincoli particolari
  - strategia redo-only (no-undo)
  - strategia undo-only (no-redo)

Come nel caso precedente, ma con riferimento ad un programma che, per realizzare i 100.000 inserimenti, utilizzi complessivamente $k = 10$ transazioni, ognuna con 10.000 inserimenti (e supponendo che non vi siano altre transazioni attive)

- numero di scritture di pagine di log:

- numero di scritture di pagine della relazione, nei tre casi seguenti:
  - strategia undo-redo senza vincoli particolari
  - strategia redo-only (no-undo)
  - strategia undo-only (no-redo)

Tecnologie e architetture per la gestione dei dati — 5 luglio 2024

Domanda 5 (40%)
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 e che, in caso di abort, il client rilanci la stessa transazione (subito dopo l'esecuzione delle altre azioni in attesa sullo stesso dato).

`start transaction isolation level .... ;`
`read(x) ;`
 	
`read(y);`
`y = y + x + 1`
`write(y);`
`commit;`
	
	`start transaction isolation level ....;`
	`read(y);`
	
	`read(x);`
	`x = x + y + 100;`
	`write(x)`
	`commit;`

A. Considerare uno scheduler con controllo di concorrenza basato su **Multiversioni** (come in Postgres) e livello di isolamento **SERIALIZABLE** su entrambe le transazioni.
Mostrare il comportamento dello scheduler, supponendo che i valori iniziali degli oggetti x e y siano entrambi **1000**. Indicare, nell'ordine, le operazioni che vengono eseguite da ciascun client, specificando, per ciascuna, il valore che viene letto o scritto. Per le transazioni rilanciate dopo abort, non è necessario scrivere tutti i dettagli, ma è sufficiente indicare sinteticamente le azioni e il risultato delle scritture.
In conclusione, dire se si verificano o meno anomalie.

client 1	client 2
Si verificano anomalie?(sì o no e in caso affermativo quale)	

Considerare nuovamente lo scenario della domanda precedente, con riferimento ai contesti diversi mostrati nella pagina seguente e rispondere a quanto richiesto sopra.

Tecnologie e architetture per la gestione dei dati — 5 luglio 2024

B. Controllo di concorrenza **Multiversioni** (come in Postgres) e livello **REPEATABLE READ**

client 1	client 2
Si verificano anomalie?(sì o no e in caso affermativo quale)	

C. Controllo di concorrenza basato su **2PL** e livello di isolamento **SERIALIZABLE**

client 1	client 2
Si verificano anomalie?(sì o no e in caso affermativo quale)	

D.Controllo di concorrenza basato su **2PL** e livello di isolamento **REPEATABLE READ**

client 1	client 2
Si verificano anomalie?(sì o no e in caso affermativo quale)	

Tecnologie e architetture per la gestione dei dati — 5 luglio 2024

Cenni sulle soluzioni

Tempo a disposizione: due ore.

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

Tecnologie e architetture per la gestione dei dati — 5 luglio 2024

Domanda 1 (15%)
Si considerino un sistema con blocchi di dimensione $B = 8000$ byte e una relazione *R(A,C,....)* di cardinalità pari circa a $N = 10.000.000$, con ennuple di $l = 80$ byte e campo A chiave di tipo stringa (ad esempio, il codice fiscale) e campo C di tipo intero anch'esso chiave (ad esempio, un numero di matricola) — quindi la relazione ha due chiavi, ognuna delle quali, da sola, identifica univocamente le ennuple. Supporre che il sistema offra

- strutture primarie di tre tipi: (a) disordinate, (b) ordinate rispetto ad un campo su cui è definito un indice, (c) hash
- indici di tipo B-tree (anche più di uno sulla stessa relazione, primari o secondari)

Considerare un carico applicativo con le seguenti operazioni **tutte di lettura**

1. ricerca di una ennupla sulla base del valore parziale (una sottostringa iniziale) della chiave A, con frequenza giornaliera $f_1 = 100.000$; supporre che il valore parziale sia abbastanza selettivo e porti alla identificazione, in media, di $n = 10$ ennuple;
2. ricerca di una ennupla sulla base del valore (completo) della chiave C, con frequenza giornaliera $f_2 = 100.000$;
3. scansione dell'intera relazione, ordinata sulla base del valore di A, con frequenza giornaliera $f_3 = 1$;
4. scansione dell'intera relazione, ordinata sulla base del valore di C, con frequenza giornaliera $f_4 = 1$;

Progettare l'organizzazione fisica della relazione, (i) scegliendo la struttura primaria fra le varie possibilità e (ii) individuando gli eventuali indici. Ragionare in termini di numero di accessi a memoria secondaria, assumendo che (1) gli indici primari abbiano profondità $p_1 = 3$ e gli indici secondari profondità $p_2 = 4$, (2) il buffer disponibile abbia (2a) dimensione minore del numero di blocchi del file e maggiore della radice quadrata di tale numero e (2b) permetta di mantenere stabilmente in memoria due livelli di indice, sia per i primari sia per i secondari; (3) l'hash, per gli accessi puntuali, abbia costo unitario. Proporre almeno due alternative (quelle che intuitivamente si ritengono migliori) e valutarne il costo. Rispondere negli spazi sottostanti, in forma sia simbolica sia numerica.

	Alternativa 1	Alternativa 2	Alternativa 3 (eventuale)
Descr. strutt.	Struttura ordinata su A con indice primario e indice secondario su C	Struttura ordinata su C con indice primario e indice secondario su A	Struttura hash su C e indice secondario su A
Costo Op. 1	p_1-2+1=ca.2: visita indice (p_1-2) e 1 blocco con i 10 record	p_2-2+10=ca.12: visita indice (p_2-2) e 10 record	p_2-2+10=ca.12: visita indice (p_2-2) e 10 record
Costo Op. 2	p_2-2+1=ca.3: visita indice (p_2-2) e 1 record	p_1-2+1=ca.2: visita indice (p_1-2) e 1 record	1
Costo Op. 3	Essendo il file ordinato, basta la scansione: L/(B/c)=100.000	È necessario ordinare il file, servono due passate: 3 × L/(B/c)=300.000	Serve ordinamento: 300.000
Costo Op. 3	Serve ordinamento: 300.000	Basta scansione: 100.000	Serve ordinamento: 300.000
Tot	2 × 100.000 + 3 × 100.000 + 100.000 × 1 + 300.000 × 1 = ca 900.000	12 × 100.000 + 2 × 100.000 + 300.000 × 1 + 100.000 × 1 = ca 1.800.000	12 × 100.000 + 1 × 100.000 + 300.000 × 1 + 300.000 × 1 = ca 1.900.000

Tecnologie e architetture per la gestione dei dati — 5 luglio 2024

Domanda 2 (15%)
Considerare ancora il caso illustrato nella domanda precedente, ma con riferimento ad una realtà con le seguenti frequenze per le operazioni:

1. $f_1 = 10.000$
2. $f_2 = 1.000.000$
3. $f_3 = 1$
4. $f_4 = 10$

Considerare ancora almeno due alternative (quelle che intuitivamente si ritengono migliori in questo caso).

	Alternativa 1	Alternativa 2	Alternativa 3 (eventuale)
Descr. strutt.	Struttura ordinata su A con indice primario e indice secondario su C	Struttura ordinata su C con indice primario e indice secondario su A	Struttura hash su C e indice secondario su A
Costo Op. 1	2	12	12
Costo Op. 2	3	2	1
Costo Op. 3	100.000	300.000	300.000
Costo Op. 3	300.000	100.000	300.000
Tot	$2 \times 10.000 + 3 \times 1.000.000 + 100.000 \times 1 + 300.000 \times 10$ = ca 6.100.000	$12 \times 10.000 + 2 \times 1.000.000 + 300.000 \times 1 + 100.000 \times 10$ = ca 3.400.000	$12 \times 10.000 + 1 \times 1.000.000 + 300.000 \times 1 + 300.000 \times 10$ = ca 4.400.000

Tecnologie e architetture per la gestione dei dati — 5 luglio 2024

Domanda 3 (30%)
Come noto, in molti sistemi (ad esempio in PostgreSQL) è possibile ordinare fisicamente le relazioni, senza che però l'ordinamento venga mantenuto. Allo scopo, l'ordinamento viene rieseguito periodicamente, su dati che sono in parte ordinati e in parte no. Una possibile strategia seguita (molto comune) è la seguente:

- le eliminazioni vengono realizzate semplicemente "marcando" i record e lasciando quindi lo spazio inutilizzato
- gli inserimenti vengono effettuati in coda, nell'ordine in cui vengono richiesto
- le modifiche vengono trattate ciascuna come un'eliminazione e un inserimento

Di conseguenza, si possono presentare situazioni come quella mostrata in sotto, in cui la relazione è in parte ordinata e in parte disordinata (l'asterisco "*" indica i record cancellati). Si noti che la relazione presenta molti blocchi ordinati (dodici nell'esempio) e alcuni disordinati (tre).

Supponendo di avere a disposizione **sei** pagine buffer, descrivere brevemente nel riquadro l'algoritmo da utilizzare per riordinare la relazione, mostrando anche il contenuto dei buffer in occasione del primo caricamento e quello alla fine dell'ordinamento. Rispondere anche alle altre domande che vengono poste.

111	...
120	...
141	...
181	...
200	...
221	...
232	...
251	...
345	...
*	
443	...
501	...
502	...
525	...
686	...
*	
735	...
774	...
783	...
801	...
805	...
839	...
842	...
900	...

535	...
171	...
484	...
838	...
262	...
646	...

- Descrivere sinteticamente l'algoritmo (bastano poche righe informali, non serve pseudocodice)
  Risposta: Si può utilizzare una variante del merge sort che tenga conto del fatto che una frazione significativa del file è ordinata. Si può procedere nel modo seguente:

  1. caricare in tre pagine del buffer i tre blocchi disordinati
  2. ordinare i record in queste tre pagine
  3. eseguire un merge fra queste tre pagine e i (dodici) blocchi ordinati, che vengono caricati uno alla volta in una quarta pagina del buffer; una quinta pagina viene utilizzata per accumulare via via il risultato, con scrittura quando essa è piena; durante la fusione vengono anche trascurati i record cancellati, e così il numero di blocchi scritti è 14

  In questo modo, ciascuno dei blocchi originari viene letto una sola volta e ciascuno dei blocchi del file riordinato viene scritto una sola volta.

  demira

- Con riferimento all'esempio e all'algoritmo proposto, quante letture fisiche di blocchi sono necessarie? | 15 | E quante scritture? | 14 |

- Mostrare il contenuto delle pagine di buffer al primo caricamento e il contenuto delle stesse alla fine dell'esecuzione dell'algoritmo

Primo caricamento delle pagine del buffer	Contenuto finale delle pagine del buffer
535 ... 171 ...	171 ... 262 ...
484 ... 838 ...	484 ... 535 ...
262 ... 646 ...	646 ... 838 ...
	842 ... 900 ...
	842 ... 900 ...

Tecnologie e architetture per la gestione dei dati — 5 luglio 2024

Domanda 4 (20%)
Considerare un sistema che utilizzi blocchi di lunghezza $D = 4$ KB (approssimabili a 4000 byte) e una tabella R con una struttura fisica heap con record a lunghezza fissa che occupano $L = 20$ byte ciascuno, in cui vengono inserite $M = 100.000$ ennuple, con valori della chiave tutti diversi fra loro e da quelli già nella relazione (quindi il sistema verifica il soddisfacimento del vincolo di chiave e ammette tutte le operazioni).
Rispondere alle domande seguenti, indicando formule e valori numerici:

Indicare il numero di scritture in memoria secondaria necessarie per realizzare i 100.000 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 $M = 100.000$, una per transazione
- numero di scritture di pagine della relazione, nei tre casi seguenti:
  - strategia undo-redo senza vincoli particolari : al massimo una per transazione, $M = 100.000$; probabilmente di meno, anche solo $M/f = 500$, una per blocco (dove f indica il fattore di blocco $f = D/L = 200$)
  - strategia redo-only (no-undo) : come per la strategia undo-redo
  - strategia undo-only (no-redo) : in questo caso certamente una per transazione, $M = 100.000$, perché è l'unico modo per evitare il redo: le modifiche debbono essere scritte prima del commit

Come nel caso precedente, ma con riferimento ad un programma che, per realizzare i 100.000 inserimenti, utilizzi complessivamente $k = 10$ transazioni, ognuna con 10.000 inserimenti (e supponendo che non vi siano altre transazioni attive)

- numero di scritture di pagine di log:
  per ogni transazione si debbono scrivere le pagine di log corrispondenti. Ogni transazione scrive M/k ennuple e quindi le relative scritture su log occupano $M/k \times L \times 3 = 600$KB e cioè $M/k \times L \times 3 * 1/D =$ ca.150 blocchi. In totale, per $k = 10$ transazioni, circa $M \times L \times 3 * 1/D =$ ca.1500 scritture
- numero di scritture di pagine della relazione, nei tre casi seguenti:
  - strategia undo-redo senza vincoli particolari non si può dire con precisione, anche solo $M/f = 500$, una per blocco
  - strategia redo-only (no-undo) per ogni transazione, si debbono scrivere i blocchi coinvolti, se le scritture sono sequenziali, ogni blocco si scrive una volta sola (a parte un po' di sfrido) quindi $M/f = 500$
  - strategia undo-only (no-redo) anche in questo caso, si può pensare che ogni blocco si scriva una volta sola, quindi ancora $M/f = 500$

Tecnologie e architetture per la gestione dei dati — 5 luglio 2024

Domanda 5 (40%)
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 e che, in caso di abort, il client rilanci la stessa transazione (subito dopo l'esecuzione delle altre azioni in attesa sullo stesso dato).

`start transaction isolation level ....  ;`
`read(x) ;`
 	
`read(y);`
`y = y + x + 1`
`write(y);`
`commit;`
	
	`start transaction isolation level ....;`
	`read(y);`
	
	`read(x);`
	`x = x + y + 100;`
	`write(x)`
	`commit;`

A. Considerare uno scheduler con controllo di concorrenza basato su **Multiversioni** (come in Postgres) e livello di isolamento **SERIALIZABLE** su entrambe le transazioni.
Mostrare il comportamento dello scheduler, supponendo che i valori iniziali degli oggetti x e y siano entrambi **1000**. Indicare, nell'ordine, le operazioni che vengono eseguite da ciascun client, specificando, per ciascuna, il valore che viene letto o scritto. Per le transazioni rilanciate dopo abort, non è necessario scrivere tutti i dettagli, ma è sufficiente indicare sinteticamente le azioni e il risultato delle scritture.
In conclusione, dire se si verificano o meno anomalie.

Risposta/commento:
Le operazioni vengono tutte eseguite nell'ordine in cui vengono richieste, fino alla scrittura da parte del secondo client, che viene rifiutata, per la presenza del ciclo lettura scrittura (la transazione 2 scrive x dopo che la 1 lo ha letto e la transazione 1 scrive y dopo che la 2 lo ha letto). Il sistema, alla richiesta di scrittura, mostra il messaggio:

```
ERROR: could not serialize access due to read/write dependencies among transactions
DETAIL: Reason code:  Canceled on identification as a pivot, during write.
```

Se poi la transazione viene rilanciata, procede regolarmente

Considerare nuovamente lo scenario della domanda precedente, con riferimento ai contesti diversi mostrati nella pagina seguente e rispondere a quanto richiesto sopra.

B. Controllo di concorrenza **Multiversioni** (come in Postgres) e livello **REPEATABLE READ**

Risposta/commento:
Le operazioni vengono tutte eseguite nell'ordine in cui vengono richieste. Si verifica una anomalia, perché in effetti il risultato che si ottiene non corrisponde ad alcune esecuzione seriale. Il valore di y che si ottiene è 2001 il che è corretto, mentre per x si ottiene 2100. In una esecuzione seriale, il secondo valore aggiornato dovrebbe essere 3101.

C. Controllo di concorrenza basato su **2PL** e livello di isolamento **SERIALIZABLE**

Risposta/commento:
Si genera uno stallo per le richieste di lock in scrittura, viene abortita la seconda transazione, che poi riparte e tutto si conclude senza anomalie

D.Controllo di concorrenza basato su **2PL** e livello di isolamento **REPEATABLE READ**

Risposta/commento:
Si genera uno stallo per le richieste di lock in scrittura, viene abortita la seconda transazione, che poi riparte e tutto si conclude senza anomalie