Tecnologie e architetture per la gestione dei dati — 5 luglio 2023

Tempo a disposizione: due ore.

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

Tecnologie e architetture per la gestione dei dati — 5 luglio 2023

Domanda 1 (20%)
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 $n = 20$ 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 $f_4 = 10$;

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 2023

Domanda 2 (15%)
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	...
472	...

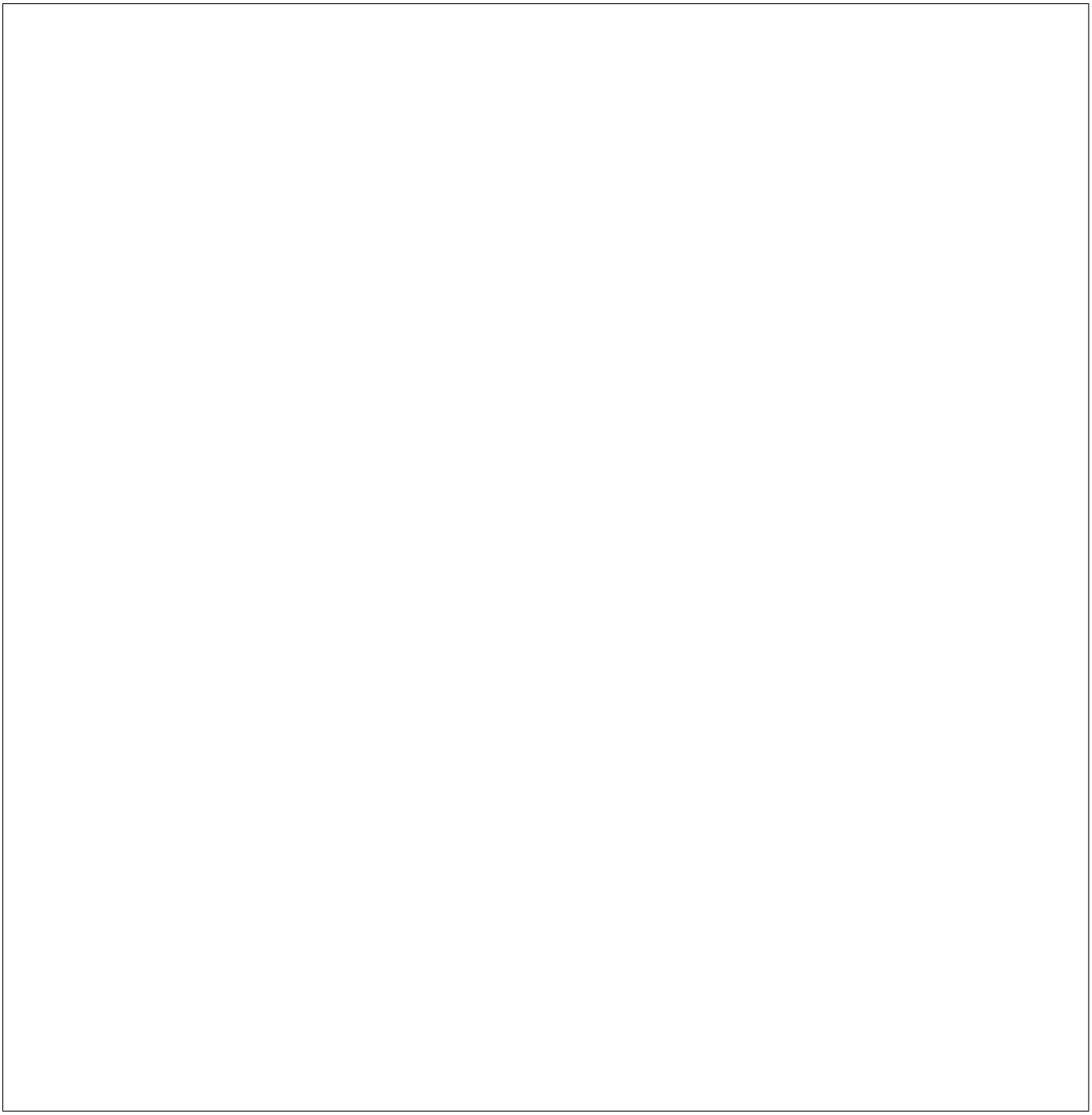
- Descrivere sinteticamente l'algoritmo (bastano poche righe informali, non serve pseudocodice)

- Con riferimento all'esempio e all'algoritmo proposto, quante letture fisiche di blocchi vengono eseguite? [ ] 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

Domanda 3 (15%)
Si consideri un B-tree con nodi intermedi che contengono due chiavi e tre puntatori e foglie con due chiavi, in cui vengano inserite chiavi (a partire dall'albero vuoto) nel seguente ordine: 42, 55, 14, 70, 15, 23, 28, 31, 34, 35, 36. Mostrare l'albero dopo l'inserimento di cinque chiavi, di otto chiavi e alla fine.



Tecnologie e architetture per la gestione dei dati — 5 luglio 2023

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, assumendo che il sistema utilizzi una strategia undo-redo senza vincoli particolari

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, sempre assumendo una strategia undo-redo senza vincoli particolari

Domanda 5 (15%)

Considerare lo scenario a fianco in cui tre client diversi inviano richieste ad un gestore della 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 al 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 la stessa transazione (subito dopo l'esecuzione delle altre azioni in attesa sullo stesso dato). Considerare uno scheduler con controllo di concorrenza basato su **Multversioni** (come in Postgres) e livello di isolamento **READ COMMITTED** sulle prime due transazioni e **SERIALIZABLE** sulla terza. Mostrare il comportamento dello scheduler, supponendo che il valore iniziale dell'oggetto x sia **500** e quello dell'oggetto y sia **100**. Indicare, nell'ordine, le operazioni che vengono eseguite da ciascun client, specificando, per ciascuna, il valore che viene letto o scritto. In conclusione, dire se si verificano o meno anomalie.

client 1	client 2	client 3
		begin read(x)
begin read(x) x = x + 10 write(x) read(y)		
	begin read(y) y = y + 20 write(y) read(x) x = x – 20 write(x) commit	
y = y – 10 write(y) commit		
		(dopo molto tempo) read(x) commit

client 1	client 2	client 3

Si verificano anomalie?

Domanda 6 (15%)

Considerare nuovamente lo scenario della domanda precedente, ripetuto qui a fianco per comodità. Considerare uno scheduler con controllo di concorrenza basato su **2PL stretto** con livello di isolamento **SERIALIZABLE** sulle prime due transazioni e **READ COMMITTED** sulla terza. Mostrare il comportamento dello scheduler, supponendo ancora che il valore iniziale dell'oggetto x sia **500** e quello dell'oggetto y sia **100**.

client 1	client 2	client 3
		begin read(x)
begin read(x) x = x + 10 write(x) read(y)		
	begin read(y) y = y + 20 write(y) read(x) x = x – 20 write(x) commit	
y = y – 10 write(y) commit		
		(dopo molto tempo) read(x) commit

client 1	client 2	client 3

Si verificano anomalie?