Basi di dati II

Prova parziale — 11 aprile 2012 — 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** _____ **Ordin.** _____

Domanda 1 (20%)
Considerare un sistema con dischi con $N = 400$ blocchi per traccia

- tempo medio di posizionamento della testina (tempo di seek) $t_S = 5$ msec
- tempo medio di latenza (attesa dovuta alla rotazione) $t_L = 3$ msec
- tempo minimo di lettura di un blocco $t_B = 15$ μsec

Rispondere alle seguenti domande mostrando formula e valore numerico (N.B. non servono calcolatrici, i calcoli sono semplici; **se si ritengono le informazioni imprecise, dare risposte approssimate, usando il buon senso**).

1. Qual è il tempo medio necessario per leggere un blocco del quale sia dato l'indirizzo fisico?

La somma del tempo medio di seek, del tempo medio di latenza e del tempo minimo di lettura di un blocco

$$t_{tot} = t_S + t_L + t_B = 5 + 3 + 0,015 \text{ msec } = \text{ ca } 8 \text{ msec}$$

2. Qual è il tempo medio necessario per la scansione sequenziale di un file costituito da $F = 100$ blocchi contigui, non letti di recente?

Il tempo medio necessario per leggere un blocco (il primo) più il tempo minimo di lettura per ciascuno degli altri

$$t_{tot} + (F - 1) \times t_B = \text{ ca } 9,5 \text{ msec}$$

3. Qual è il tempo che si può ipotizzare necessario per eseguire un accesso diretto ad un record di un file attraverso un indice che abbia profondità $p = 4$ e fan-out (fattore di blocco dell'indice) $f_I = 100$, usato di recente, ma in modo non molto intenso?

Si può immaginare che al massimo la radice dell'indice si trovi nel buffer, ma non gli altri nodi. Quindi, tre accessi per l'indice e uno per il record del file, in posizioni non prevedibili:

$$4 \times t_{tot} = \text{ ca } 32 \text{ msec}$$

4. Qual è il tempo che si può ipotizzare necessario per eseguire $m = 1000$ accessi diretti in tempi ravvicinati a record di un file (molto grande) attraverso un indice che abbia profondità $p = 4$, fan-out $f_I = 100$, con disponibilità di circa $P = 150$ pagine di buffer?

Si può immaginare che la radice e un altro livello dell'indice restino nel buffer, dopo il primo caricamento, quindi, per ogni record, due accessi all'indice e uno al file, in posizioni non prevedibili (qualche accesso in più per il caricamento iniziale e qualcuno in meno per blocchi acceduti più volte):

$$3 \times m \times t_{tot} = \text{ ca } 24 \text{ sec}$$

Domanda 2 (20%)
Si consideri una base di dati con le relazioni

- R1(A,B,C,D) con
  - N_1=2.500.000 ennuple di lunghezza $l_1 = 20$ Byte
  - b=5 valori diversi sull'attributo B (tutti gli interi compresi fra 1 e b)
  - c=25.000 valori diversi sull'attributo C (tutti gli interi compresi fra 1 e c)
  - una struttura disordinata, un indice sulla chiave primaria A e un altro sull'attributo C;
- R2(E,F,G) con
  - N_2=1.000.000 ennuple di lunghezza $l_2 = 100$ Byte
  - una struttura disordinata, un indice sulla chiave primaria E

Supporre che:

- i blocchi abbiano dimensione $P = 2$ KByte (approssimabile a 2000 Byte);
- ogni operazione possa contare su un numero di pagine di buffer pari a circa q=300;
- gli indici abbiano tutti p=4 livelli (contando anche radice e foglie) e fattore di blocco massimo f_i=100;
- il sistema esegua i join come nested loop oppure come hash-join (si ricorda che questi ultimi hanno un costo pari a circa tre volte la somma dei blocchi dei due file, a condizione che il quadrato del numero di pagine di buffer disponibili sia maggiore del numero di blocchi del più piccolo dei due file).

Valutare il costo, indicandolo in modo sia simbolico sia numerico e specificando quale algoritmo si utilizza per il join di ciascuna delle interrogazioni seguenti (NB: **al di là del dettaglio, la cui precisione non è forse nemmeno possibile, è essenziale mostrare una comprensione degli ordini di grandezza**):

```
select *
from R1 join R2 on D=E
```

Conviene l'hash-join; costo:

$$3 \times (\frac{N_1}{P/l_1} + \frac{N_2}{P/l_2}) = 225.000 \text{ (nei compiti B e D: ca 345.000)}$$

Il nested-loop costerebbe molto di più

$$\frac{N_1}{P/l_1} + N_1(p - 2 + 1) = \text{ ca 7.500.000 (nei compiti B e D: ca 4.500.000)}$$

```
select *
from R1 join R2 on D=E
where B=5
```

Conviene ancora l'hash-join, con una piccola riduzione di costo (perché dopo la prima scansione di R1 si possono scartare le ennuple non selezionate):

$$\frac{N_1}{P/l_1} + 2 \times \frac{N_1}{P/l_1} \times \frac{1}{b} + 3 \times \frac{N_2}{P/l_2} = \text{ ca 117.000 (nei compiti B e D: ca 317.000)}$$

Il nested-loop costerebbe di più anche in questo caso

$$\frac{N_1}{P/l_1} + \frac{N_1}{b}(p - 2 + 1) = \text{ ca 1.500.000 (nei compiti B e D: ca 900.000)}$$

```
select *
from R1 join R2 on D=E
where C=502
```

Conviene sicuramente un nested loop in cui come tabella esterna c'è un accesso diretto a R1 per la selezione su B (costo p per l'indice e $\frac{N_1}{c}$) e sulla tabella interna si fa pure l'accesso diretto ($\frac{N_1}{c}$ accessi, ognuno dei quali costa $p - 1 + 1$, perché si può ipotizzare la radice nel buffer, ma non gli altri livelli):

$$p + \frac{N_1}{c} + \frac{N_1}{c}(p - 1 + 1) = \text{ ca. } 500$$

L'hash-join richiederebbe scansioni, con un costo sicuramente maggiore.

Domanda 3 (20%)
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, ignorare commit e abort), 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
$s_2, s_1, r_2(x), w_2(x), c_2, r_1(x), w_1(x), c_1$	sì	sì	sì	no
$s_2, r_2(x), w_2(x), c_2, s_1, r_1(x), w_1(x), c_1$	sì	sì	sì	sì
$s_1, r_1(x), s_2, r_2(x), w_1(x), c_1, w_2(x), c_2$	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 *

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

Domanda 4 (20%)
Una catena di supermercati ha una base di dati dei propri clienti che dispongono di una "tessera fedeltà," con varie informazioni su ciascun cliente, fra cui (a) il totale dei punti acquisiti attraverso l'uso della tessera e (b) il negozio della catena cui fa riferimento (ad esempio, quello presso cui ha inizialmente richiesto la tessera). Si vuole eseguire su di essa l'interrogazione che calcola, per ciascun negozio, il numero dei clienti, la somma dei punti fedeltà dei clienti e la relativa media per cliente. Indicare quale livello di isolamento (`READ UNCOMMITTED`, `READ COMMITTED`, `REPEATABLE READ` o `SERIALIZABLE`) si potrebbe scegliere in ciascuno dei seguenti casi (si supponga che, in generale, sia stato rilevato che, nel corso degli inserimenti e delle modifiche, vengono inseriti valori sbagliati anche di vari ordini di grandezza, che sono poi corretti prima del commit):

1. L'operazione è eseguita mentre vengono inseriti alcuni nuovi clienti (per ciascun negozio pochi rispetto a quelli già presenti), con la finalità di acquisire informazioni approssimate ma ragionevolmente indicative sugli andamenti complessivi.
2. L'operazione è eseguita mentre vengono inseriti molti nuovi clienti, con la finalità di acquisire informazioni approssimate ma ragionevolmente indicative sugli andamenti complessivi.
3. L'operazione è eseguita mentre vengono inseriti alcuni nuovi clienti (per ciascun negozio pochi rispetto a quelli già presenti), con la finalità di individuare i primi tre negozi da premiare in una campagna promozionale sulla base dei punti acquisiti dai rispettivi clienti.
4. L'operazione è eseguita mentre vengono modificati i valori dei punti fedeltà di tutti i clienti (a seguito di una ridefinizione dei criteri di assegnazione dei punti stessi), con la finalità di individuare i primi tre negozi da premiare in una campagna promozionale sulla base dei punti acquisiti dai rispettivi clienti.
5. L'operazione è eseguita in un momento in cui non ci sono aggiornamenti di alcun genere, con la finalità di acquisire informazioni approssimate ma ragionevolmente indicative sugli andamenti complessivi.

Risposte				
1.	2.	3.	4.	5.
RC	S	S	RR	RU

Domanda 5 (20%)
Si considerino due relazioni $R_1(\underline{A}, C)$, $R_2(\underline{A}, D, E, F)$, in cui gli attributi hanno tutti la stessa dimensione $L = 4$Byte, molto più piccola della dimensione del blocco pari a $B = 4000$ Byte. Si supponga che le relazioni abbiano entrambe $N = 1.000.000$ ennuple, con gli stessi valori su A, e che le operazioni più frequenti su di essa siano le seguenti:

- o_1 selezione di una ennupla del join di R_1 e R_2 (sulla base del valore di A), con frequenza $f_1 = 10.000$;
- o_2 scansione dell'intera relazione R_1, con frequenza $f_2 = 1$

Valutare le due seguenti alternative di memorizzazione, calcolando il costo complessivo (riportare la formula che indica il numero di accessi nell'unità di tempo, in base alle variabili sopra citate):

(i) memorizzazione separata delle due relazioni, entrambe ordinate su A e con indice primario su A con profondità $p = 4$, con 2 livelli mediamente disponibili nel buffer.

costo unitario di o_1:	costo unitario di o_2:
$c_1 = 3 + 3 = 6$	$c_2 = \frac{N}{B/(2 \times L)} = \frac{2 \times N \times L}{B} = 2000$
costo complessivo: $c_1 \times f_1 + c_2 \times f_2 = 6 \times 10.000 + 2000 \times 1 = 62.000$	

(ii) memorizzazione in un cluster delle due relazioni pure entrambe ordinate su A e con indice primario su A con profondità sempre $p = 4$, con 2 livelli mediamente disponibili nel buffer.

costo unitario di o_1:	costo unitario di o_2:
$c_1 = 3$	$c_2 = \frac{N}{B/(6 \times L)} = \frac{6 \times N \times L}{B} = 6000$
costo complessivo: $c_1 \times f_1 + c_2 \times f_2 = 3 \times 10.000 + 6000 \times 1 = 36.000$	

In conclusione, conviene quindi la memorizzazione in un cluster? (Sì o No) Sì