Basi di dati II, primo modulo Prova parziale — 28 marzo 2011— Compito A
Cenni sulle soluzioni (per il compito A, gli altri sono simili)

Rispondere su questo fascicolo.
Tempo a disposizione: un'ora e quindici minuti.

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

Domanda 1 (15%)
Considerare un disco con una velocità di rotazione di v = 15.000 giri al minuto e un tempo medio di posizionamento della testina (tempo di seek) $t_S = 5$ msec. Ogni traccia contiene N = 400 blocchi e ogni blocco contiene B = 4 KB. Rispondere alle seguenti domande mostrando formula e valore numerico (N.B. non servono calcolatrici, i risultati sono semplici, approssimare 1 MB = 1000 KB).

1. Qual è tempo medio di latenza (attesa dovuta alla rotazione) t_L?

Il tempo necessario per mezzo giro:

$$t_L = \frac{1}{2} \times \frac{1}{v} = \frac{1}{2} \times \frac{1}{15.000} \text{ min} = \frac{1}{2} \times \frac{1}{15.000} \times 60 \text{ sec} = 0,002 \text{ sec} = 2 \text{ msec}$$

2. Qual è la massima velocità di trasferimento (in MB al secondo)?

Velocità di rotazione moltiplicato capacità di una traccia

$$v \times N \times B = 15.000 \times 400 \times 4 \text{ KB/min} = 15.000 \times 400 \times 4 \times \frac{1}{60} \text{ KB/sec} = 400.000 \text{ KB/sec} = 400 \text{ MB/sec}$$

3. Qual è il tempo minimo t_B necessario per leggere un blocco?

Il tempo necessario per fare un giro diviso il numero di blocchi di una traccia

$$t_B = \frac{1}{v} \times 60 \times \frac{1}{N} = \frac{1}{15.000} \times \frac{1}{400} \text{min} = \frac{1}{15.000} \times 60 \times \frac{1}{400} \times 60 \text{sec} = 10 \mu\text{sec}$$

4. Qual è il tempo medio necessario per leggere un blocco?

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

$$t_S + t_L + t_B = 5 + 2 + 0,010 \text{ msec} \ = \ \text{ca } 7 \text{ msec}$$

5. Qual è il tempo medio necessario per leggere una traccia?

Il tempo medio di seek più il tempo necessario per un giro

$$t_S + \frac{1}{v} = 5 \text{ msec} \ + \frac{1}{15.000} \text{ min} \ = 5 \text{ msec} \ + \frac{1}{15.000} \times 60 \times 1000 \text{ msec} \ = 9 \text{ msec}$$

Domanda 2 (25%)
Si consideri una base di dati con le relazioni (entrambe con indice sulla chiave primaria)

$$T1(\underline{A},B,C),\ T2(\underline{D},E,F)$$

Eseguendo le interrogazioni seguenti, su Postgres (ma il comportamento su altri sistemi è probabilmente lo stesso), si rilevano le seguenti scelte per l'operatore di join:

1.	`select *` `from T1 join T2 on C=D`	Hash join

2.	`select *`
	`from T1 join T2 on C=D`
	`where A>2 and A<23`
		Nested loop join

3.	`select A, B, C`
	`from T1 join T2 on C=D`
	`where A>2 and A<23`
		Nested loop join

Motivare ciò, valutando, per ciascuna delle tre interrogazioni, il costo di un piano di esecuzione con hash join e uno con nested loop, e supponendo che

- le relazioni abbiano rispettivamente N_1=2.000.000 ed N_2=1.000.000 ennuple, (con fattore di blocco rispettivamente f_1=20 e f_2=10)
- gli indici abbiano entrambi p=4 livelli (contando anche radice e foglie) e il fattore di blocco massimo dell'indice sia, in entrambi i casi f_i=90
- l'operazione possa contare su un numero di pagine di buffer pari a circa q=150.

Rispondere riempiendo la tabella sottostante, indicando il costo in modo sia simbolico sia numerico (considerare, ovviamente, anche l'eventuale selezione)

	Hash join	Nested loop join
1.	senza uso dei buffer, ciascuna delle relazioni viene letta (un accesso per blocco) e rimemorizzata (un accesso per ogni record), poi si leggono in parallelo le due rimemorizzazioni (un accesso per blocco); trascuriamo per semplicità il costo della generazione del risultato, perché dipende dalla selettività; il costo è circa $2 \times \frac{N_1}{f_1} + N_1 + 2 \times \frac{N_2}{f_2} + N_2 = 3.600.000$ Con l'uso dei buffer il costo si può ridurre, anche di molto; in particolare, con un numero di buffer pari a circa la radice del numero di blocchi del file più piccolo (che però non abbiamo in questo caso), si ha un costo: $3 \times (\frac{N_1}{f_1} + \frac{N_2}{f_2}) = 600.000$ L'algoritmo costruisce rimemorizzazioni hash con tante liste di blocchi quante sono le pagine di buffer disponibili e poi costruisce le ennuple del join esaminendo le liste omologhe; se le liste entrano in memoria, basta una rilettura per ciascun file (e il costo è quello sopra), altrimenti è un po' maggiore, perché la rimemorizzazione della relazione più piccola va riletta più volte	$\frac{N_1}{f_1} + N_1 \times (p - 2 + 1) = 6.100.000$ (p-2+1): i livelli dell'indice, meno quelli nel buffer (mediamente 2), più 1 per l'accesso al record
2.	$a = 20$, numero di valori diversi di A selezionati dalla `where` $p + a + \frac{N_2}{f_2} =$ ca 100.000 Alla relazione esterna si accede con l'indice: profondità più scansione, perché un intervallo; le poche ennuple vengono organizzate in memoria secondo la struttua hash e quindi basta poi una scansione della seconda relazione	$(p + a) + a \times (p - 1 + 1) =$ ca 100 Alla relazione esterna si accede con l'indice: $(p + a)$ per accedere ai record di interesse di R1; poi a accessi diretti, con la sola radice nel buffer
3.	idem	$(p + a) + a \times (p - 1) =$ ca 80 Per la relazione interna non serve l'accesso ai record

Domanda 3 (20%)
Si consideri una base di dati con le relazioni

R1(A,B,C,D), R2(E,F,G), R3(H,I,L), R4(M,N,P)

con un vincolo di integrità referenziale fra l'attributo B di R1 e la chiave E di R2 e con la vista:

- ```
  create view V as
  select *
  from R1 join R2 on B=E
  left join R3 on C=H
  join R4 on D=M
  ```

Per ciascuna delle seguenti interrogazioni, indicare quali join debbono essere eseguiti per rispondere all'interrogazione stessa

`select E,F` `from V`	$R_1 \bowtie_{B=E} R_2 \bowtie_{D=M} R_4$
`select H,I,L` `from V`	$R_1 \bowtie_{C=H} R_3 \bowtie_{D=M} R_4$
`select A,B` `from V`	$R_1 \bowtie_{D=M} R_4$
`select A,M,N,P` `from V`	$R_1 \bowtie_{D=M} R_4$

Osservazione alcuni DBMS (ad esempio Postgres) non sfruttano il vincolo di integrità referenziale e quindi eseguono sempre anche il join con R_2

Domanda 4 (20%)
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, 1, 4, 8, 10, 12, 14, 16, 20, 24, 5, 6, 13. Mostrare l'albero dopo l'inserimento di quattro, sette, dieci chiavi e alla fine.

provare con l'applet suggerita a lezione: `http://slady.net/java/bt/view.php?w=700`
  ```

Domanda 5 (20%)
Si supponga di avere un recovery manager che utilizzi un checkpoint non quiescente e che scriva record di update ("SetStringRecord" secondo la terminologia di SimpleDB) aventi la forma seguente:

<SETSTRING, TxID, TableName, BlkNo, Offset, BeforeValue, AfterValue >

Si noti che, rispetto alla notazione usata sul libro, l'oggetto dell'operazione viene identificato da TableName (nome della relazione o meglio del file che la memorizza), BlkNo (numero del blocco nel file), Offset (posizione del valore di interesse nel blocco).

In tale contesto, supporre che il recovery manager, al riavvio dopo un crash, trovi i seguenti record nel log:

```
<START, 2>
<SETSTRING, 2, Impiegati, 33, 0, xxxx, Rossi>
<START, 1>
<SETSTRING, 1, Impiegati, 44, 0, xxxx, Neri>
<START, 3>
<COMMIT, 2>
<SETSTRING, 3, Impiegati, 33, 0, Rossi, Verdi>
<START, 4>
<SETSTRING, 4, Impiegati, 55, 0, xxxx, Bruni>
<NQCKPT, 1, 3, 4>
<SETSTRING, 4, Impiegati, 55, 0, Bruni, Neri>
<SETSTRING, 4, Impiegati, 66, 0, xxxx, Bianchi>
<START, 5>
<COMMIT, 4>
```

1. Fino a quale record del log arriva la scansione a ritroso?

<START, 1>

2. Quali modifiche sulla base di dati debbono essere eseguite durante un recovery di tipo undo-redo?

undo(SETSTRING,3,...), undo(SETSTRING,1,...), redo(SETSTRING, 4, ...,55, Bruni), redo(SETSTRING, 4, ...,55, Neri), redo(SETSTRING, 4, ...,55, Bianchi)
In effetti, la prima delle redo non serve, se il checkpoint, come succede di solito (nele strategie diverse dalla redo-only), salva tutte le pagine sporche del buffer

3. Quali modifiche sulla base di dati debbono essere eseguite durante un recovery di tipo undo-only?

undo(SETSTRING,3,...), undo(SETSTRING,1,...)

4. È possibile che la transazione T_1 abbia modificato il buffer contenente il blocco 70 della relazione Impiegati? Spiegare perché e in caso affermativo spiegarne le conseguenze.

Sì, perché la modifica può essere avvenuta con il record corrispondente pure scritto nel log, ma senza che il log sia stato scritto su disco

5. È possibile che la transazione T_1 abbia modificato su disco il blocco 70 della relazione Impiegati? Spiegare

No, perché la regola "write-ahead log" richiede che la modifica su disco venga fatta solo dopo la scrittura su disco del corrispondente record di log

6. È possibile che la transazione T_1 abbia modificato su disco il blocco 44 della relazione Impiegati?

In caso di strategia redo-only, la modifica sicuramente non è avvenuta (si scrive solo dopo il commit). Negli altri casi, il checkpoint viene di solito implementato copiando su disco tutte le pagine sporche del buffer e quindi la modifica è avvenuta.